МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет

Навчально-науковий інститут новітніх освітніх технологій

Кафедра інформаційно-обчислювальних систем і управління

ХАРХАЛІС Валерія Юріївна

Веб-застосунок для інтервального запам'ятовування / Spaced repetition web-application

спеціальність: 122 - Комп'ютерні науки

освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконала студентка групи КНз-41

В.Ю. Хархаліс

_____

Науковий керівник:

к.т.н., доцент, І.В. Турченко

_____

Кваліфікаційну роботу

допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри

_____Н. В. Дзюбановська

ТЕРНОПІЛЬ – 2026

Навчально-науковий інститут новітніх освітніх технологій

Кафедра інформаційно-обчислювальних систем і управління

Освітній ступінь «бакалавр»

спеціальність: 122 – Комп’ютерні науки

освітньо-професійна програма – Комп’ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Н.В. Дзюбановська

«____» _____20__р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ХАРХАЛІС Валерії Юріївні

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Веб-застосунок для інтервального запам'ятовування / Spaced repetition web-application

керівник роботи к.т.н., доцент, І.В. Турченко

затверджений наказом по університету від 01 грудня 2025 року № 894.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити

– описати предметну область розробки веб-застосунку інтервального запам’ятовування;

– оглянути та проаналізувати існуючі рішення;

– постановити задачі проєктування;

– описати загальну структуру проєктованого веб-застосунку;

– описати алгоритмічне забезпечення веб-застосунку;

– описати інформаційне забезпечення веб-застосунку;

– реалізувати програмне забезпечення;

– розробити інтерфейс користувача;

– протестувати розроблену програму.

5. Перелік графічного матеріалу у роботі

– Схема взаємодії складових шаблону “Модель-Вигляд-Контролер”

– Схема взаємодії окремих модулів програми

– Схема алгоритму SuperMemo-2

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Назва етапів кваліфікаційної роботи	до 01.01.2026 р.	
2	Огляд літературних джерел відповідно до предметної області та складання плану роботи	до 01.02.2026 р.	
3	Написання 1 розділу кваліфікаційної роботи	до 05.02.2026 р.	
4	Написання 2 розділу кваліфікаційної роботи	до 01.05.2026 р.	
5	Написання 3 розділу кваліфікаційної роботи	до 15.05.2026 р.	
6	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 25.05.2026 р.	
7	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 30.05.2026 р.	
8	Перевірка кваліфікаційної роботи на оригінальність тексту	до 10.06.2026 р.	
9	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	

Студент_____В.Ю. Хархаліс
підпис

Керівник роботи _____к.т.н., доцент І.В. Турченко
підпис

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-застосунок для інтервального запам'ятовування» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 61 сторінку і містить 24 ілюстрації, 3 таблиці, 6 додатків та 26 використаних джерел.

Метою роботи є дослідження підходів до розроблення систем для інтервального запам’ятовування та створення програмного веб-застосунку для оптимізації процесу засвоєння інформації на базі алгоритму SM-2.

Методами розроблення обрано метод системного аналізу (для вивчення наявних програмних рішень та специфікацій формату стиснених даних), метод об’єктно-орієнтованого проектування (для побудови масштабованої архітектури додатка на базі програмної платформи Laravel), методи моделювання баз даних (для проектування надійної реляційної структури збереження навчального матеріалу та історії повторень), а також методи алгоритмічної оптимізації.

Унаслідок виконання роботи обґрунтовано вибір технологічних та архітектурних рішень для створення освітніх платформ та розроблено програмний додаток, який дає змогу централізовано створювати, імпортувати та вивчати картки, а також аналізувати успішність користувача за допомогою візуалізованого аналізу даних у межах єдиного багатомовного користувацького середовища.

Результати дослідження можуть бути використані для подальшого вдосконалення механізмів адаптивного тестування, розширення програмної системи (зокрема, розроблення автоматизованої програми-помічника для швидкого доступу до навчальних занять), а також слугувати програмною основою для розроблення спеціалізованих комерційних освітніх продуктів.

Ключові слова: ІНТЕРВАЛЬНЕ ЗАПАМ’ЯТОВУВАННЯ, АЛГОРИТМ SUPERMEMO, ВЕБ-ЗАСТОСУНОК, ВІЗУАЛІЗАЦІЯ СТАТИСТИКИ, ОСВІТНІ ТЕХНОЛОГІЇ

ANNOTATION

The bachelor's thesis on the topic "Spaced repetition web-application" for the degree of Bachelor in specialty 122 "Computer Science", educational program "Computer Science", consists of 61 pages and contains 24 figures, 3 tables, 6 appendices, and 26 references.

The purpose of the study is to research approaches to the development of spaced repetition web-application and to create a software application for optimizing the information assimilation process based on the SM-2 algorithm. .

The chosen development methods include the method of systems analysis (to study existing software solutions and compressed data format specifications), the method of object-oriented design (to build a scalable application architecture based on the Laravel software framework), database modeling methods (to design a reliable relational structure for storing educational material and repetition history), as well as algorithmic optimization methods.

As a result of the work, the choice of technological and architectural solutions for creating educational platforms was substantiated, and a software application was developed that allows to centrally create, import, and study flashcards, as well as analyze user performance through visualized data analysis within a unified multilingual user environment.

The research results can be used to further improve adaptive testing mechanisms, expand the software system (in particular, develop an automated assistant program for quick access to study sessions), and serve as a software foundation for the development of specialized commercial educational products.

Keywords: SPACED REPETITION, SUPERMEMO ALGORITHM, WEB APPLICATION, STATISTICS VISUALIZATION, EDUCATIONAL TECHNOLOGIES.

ЗМІСТ

ВСТУП

У сучасному світі, коли потреба у високій продуктивності стрімко зростає, виникає необхідність запам'ятовувати нову інформацію за мінімальну кількість часу та з найбільшим об'ємом. На протязі всього розвитку людства виникають нові психологічні підходи для спрощення цієї потреби, і також, разом з цим, розвиваються й технології, які полегшують виконання подібної задачі.

Традиційні підходи до навчання, котрі базуються на механічному зазубрюванні в короткі терміни, мають вкрай низьку ефективність в довгий термін. Це явище дослідив німецький психолог Герман Еббінгауз в кінці 21 століття, математично описавши його як “крива забування” [1]. Це обгрунтування стверджує, що без активного повторення людина забуває до 60% вивченого матеріалу вже на протязі кількох днів і вже через деякий час у пам’яті залишається лише 20% від усього обсягу інформації. Щоб цьому запобігти, когнітивна психологія запропонувала підхід методом інтервального повторення або інтервального запам’ятовування (англ. Spaced Repetition) [2], який базується на повторенні матеріалу через певні поступово зростаючі проміжки часу. Кожне таке повторення змінює “криву забування”, роблячи її більш стійкою до різкого падіння, що означає формування стійких нейронних зв’язків.

Втім, реалізація цього підходу справді покращилась лише з розвитком обчислювальної техніки. Першопроходцем в цій сфері щодо питання інтервального запам’ятовування став польський дослідник Пьотр Возняк. У 1980-х роках він розпочав комп’ютеризація процесу планування повторень і розробив родину алгоритмів SuperMemo, найвідомішим з яких є алгоритм SM-2 [3]. Особливість цього алгоритму полягає в “факторі спрощення” для кожної окремої одиниці інформації (картки). Алгоритм динамічно розраховує ідеальний день для наступного показу картки, враховуючи кожну відповідь користувача на неї, саме тоді коли ймовірність забування стає критичною.

Незважаючи на доведену математичну ефективність алгоритму SuperMemo, сучасна проблематика застосування систем інтервального запам’ятовування змістилася у площину архітектурних рішень та користувацького досвіду (UX). Більшість популярних рішень (наприклад, класичний Anki) створювалися переважно як десктопні програми. Вони часто мають перевантажений і застарілий інтерфейс, високий поріг входження для новачків, а також вимагають встановлення локального програмного забезпечення на кожен пристрій. Локальне зберігання баз даних ускладнює безшовну синхронізацію навчального прогресу між робочим комп'ютером, ноутбуком та смартфоном користувача, що є критично важливим критерієм для сучасного програмного продукту.

Метою роботи є розроблення веб-застосунку для інтервального запам’ятовування.

Для досягнення поставленої мети передбачається виконання наступних завдань:

– описати предметну область розробки веб-застосунку інтервального запам’ятовування;

– оглянути та проаналізувати існуючі рішення;

– постановити задачі проєктування;

– описати загальну структуру проєктованого веб-застосунку;

– описати алгоритмічне забезпечення веб-застосунку;

– описати інформаційне забезпечення веб-застосунку;

– реалізувати програмне забезпечення;

– розробити інтерфейс користувача;

– протестувати розроблену програму.

Результати роботи опубліковано в матеріалах міжнародної науково-практичної конференції студентів і молодих вчених «ICSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами», м. Тернопіль, 21–22 травня 2026 р. (додаток Е).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

1.1 Опис предметної області розробки веб-застосунку інтервального запам’ятовування

Предметною областю даного дослідження є програмні веб-застосунки для інтервального повторення (англ. Spaced Repetition Systems, SRS). У сучасному швидкоплинному інформаційному середовищі проблема ефективного засвоєння, систематизації та довготривалого збереження великих обсягів даних набуває особливої актуальності. Фундаментальною фізіологічною та когнітивною перешкодою, яку покликані вирішувати інформаційні системи даного класу, є природний процес втрати інформації пам'яттю людини, що описується «кривою забування» Германа Еббінгауза (рисунок 1.1) [1].

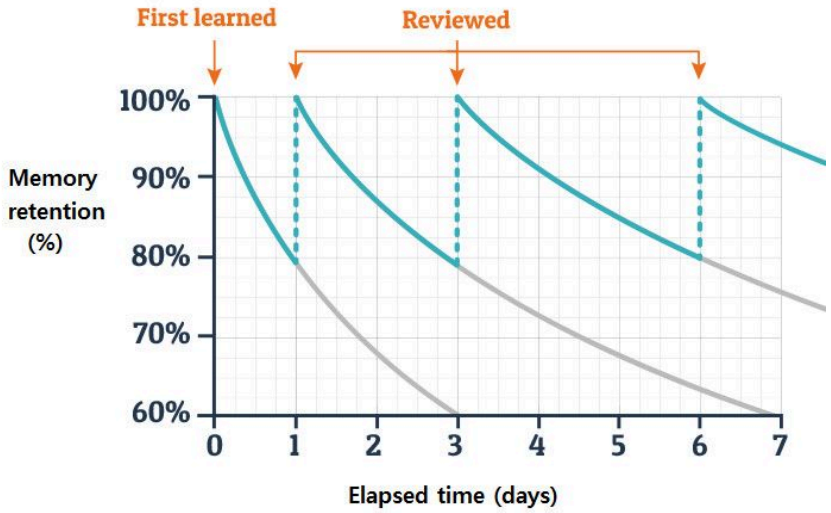


Рисунок 1.1 – “Крива забування” Германа Еббінгауза

Для оптимізації процесу запам'ятовування застосовується метод інтервального повторення – науково обґрунтована мнемонічна техніка, яка базується на повторенні навчального матеріалу із поступовим збільшенням часових проміжків між сеансами. Програмна автоматизація цього методу вимагає

застосування спеціалізованих математичних моделей для точного прогнозування моменту, коли інформація близька до забування, і саме тоді потребує повторення.

У контексті даної роботи ключовим елементом предметної області є алгоритмічна база планування повторень, зокрема математична логіка сімейства алгоритмів SuperMemo, а саме його другої ітерації (SM-2). Алгоритм SM-2 є класичним та одним із найбільш перевірених підходів, який обчислює наступний інтервал повторення (I) та модифікує коефіцієнт легкості (E-Factor) для кожної одиниці інформації (картки) на основі суб'єктивної оцінки користувачем якості пригадування під час поточного сеансу.

Попри високу ефективність математичної моделі SM-2, аналіз сучасного стану предметної області показує наявність проблеми утримання уваги користувачів та забезпечення систематичності навчання. У зв'язку з цим, стійку зацікавленість аудиторії доцільно забезпечувати за рахунок проектування інтерактивного користувацького веб-інтерфейсу з легкими методами доповнення користувацької бази знань (карток) та засобів наочного відображення успішності. Створення розширеної інформаційної панелі з елементами візуалізації щоденного прогресу, графіками розподілу навчального матеріалу та прогнозуванням майбутнього навантаження дає змогу підвищити внутрішню мотивацію. Використання сучасних веб-технологій забезпечує безперешкодний і швидкий доступ до системи через інструменти браузера на будь-яких пристроях, що мінімізує бар'єр початку роботи та забезпечує безшовну взаємодію із програмним комплексом у межах єдиного веб-простору.

Таким чином, для реалізації програмного продукту для цієї предметної області необхідно визначити такий склад функцій:

– реєстрація та авторизація користувачів за допомогою захищених облікових записів;

– формування, редагування та структурування бази навчальних матеріалів у вигляді карток та формуванням їх у колоди;

Математичне планування розкладу та обчислення оптимальних інтервалів повторень за алгоритмом SM-2.

1) Ініціалізація навчальних сесій та забезпечення інтерактивної взаємодії з інтерфейсом у режимі реального часу.

2) Збирання, оброблення та статистичний аналіз оцінок якості пригадування інформації для подальшого коригування графіку навчання та відображення аналітики.

Для наочного відображення ієрархії та взаємозв'язку цих етапів розроблено діаграму дерева функцій, яка представлена на рисунку 1.2.

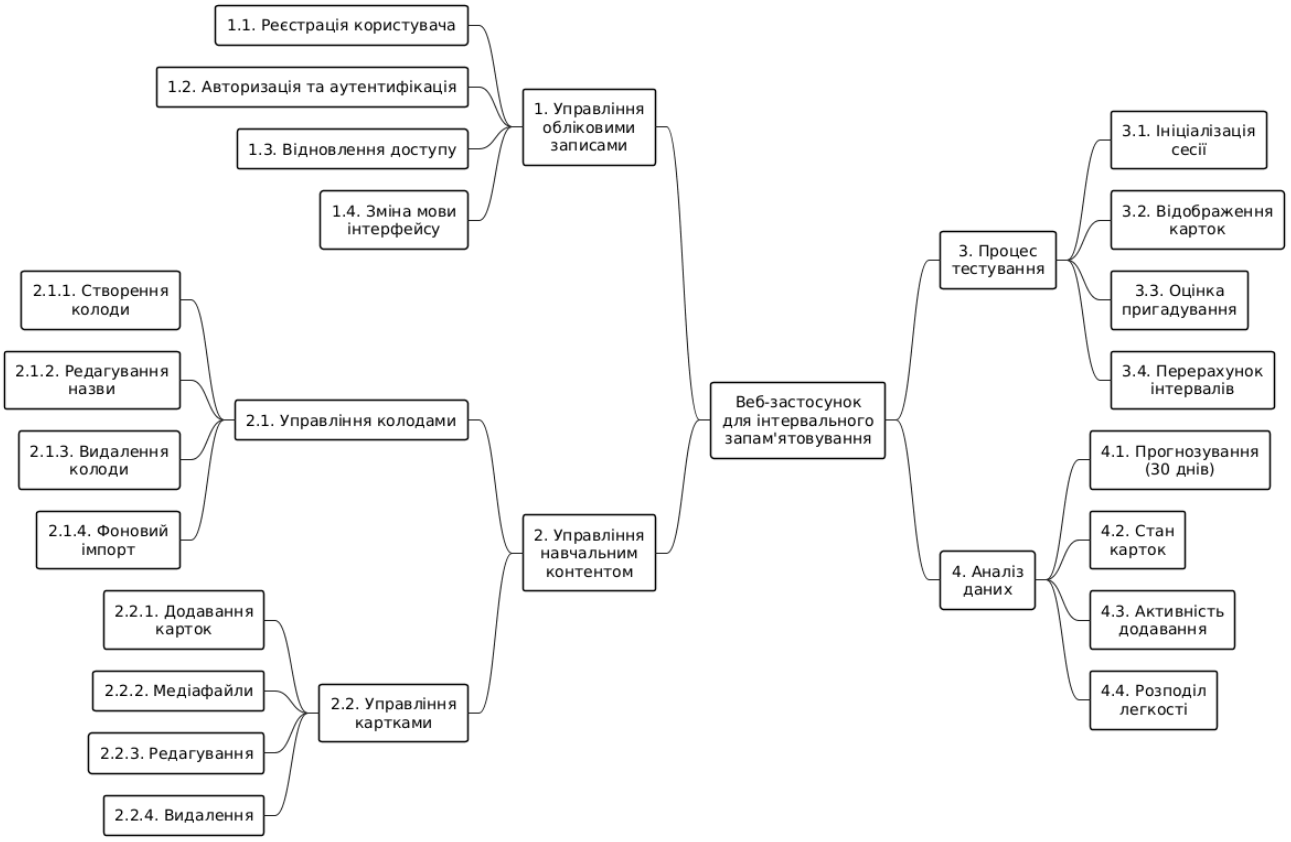


Рисунок 1.2 – Діаграма дерева функцій веб-застосунку інтервального запам’ятовування

Транзакційна складова процесу забезпечує збір, накопичення та оброблення кількісних даних про матеріал вивчення. До неї належить безперервна фіксація взаємодій користувачів із веб-застосунком: запис поточних оцінок якості пригадування, динамічний перерахунок і збереження нових значень E-Factor та інтервалів для кожної картки.

Аналітична складова бізнес-процесу забезпечує аналіз кількісних показників, сформованих у транзакційній складовій. Вона має забезпечити дослідження кількісних показників у різних розрізах і вимірах:

– за періодами часу: аналіз динаміки активності користувачів (щоденна / щотижнева кількість повторень) та виявлення оптимальних годин для нагадувань;

– за матеріалами: аналіз ефективності окремих навчальних модулів чи наборів карток (виявлення матеріалів з найнижчим відсотком успішного запам'ятовування);

– за користувачами: оцінка індивідуального прогресу, рівня залученості та відтоку користувачів.

1.2 Огляд і аналіз існуючих рішень

Для формування оптимальних вимог до розроблюваної системи виконано аналіз функціональності й інтерфейсу провідних програмних продуктів, призначених для автоматизації процесів інтервального повторення:

– Anki [4];

– SuperMemo [5];

– SuperMemo.com [6];

– Quizlet [7].

Anki [4] – класична система, що базується на модифікованому алгоритмі SM-2:

а) екранні форми: інтерфейс додавання карток (підтримка HTML/CSS), вікно сеансу повторення з кнопками оцінки складності («Знову», «Важко», «Добре», «Легко»), екран статистичних графіків;

б) підтримувані платформи: десктоп (Windows, Linux, macOS), мобільна (iOS, Android);

в) аналіз: продукт має еталонну математичну модель та має середній ступінь входження.

Головні екранні форми Anki зображено на рисунках 1.3 – 1.4.

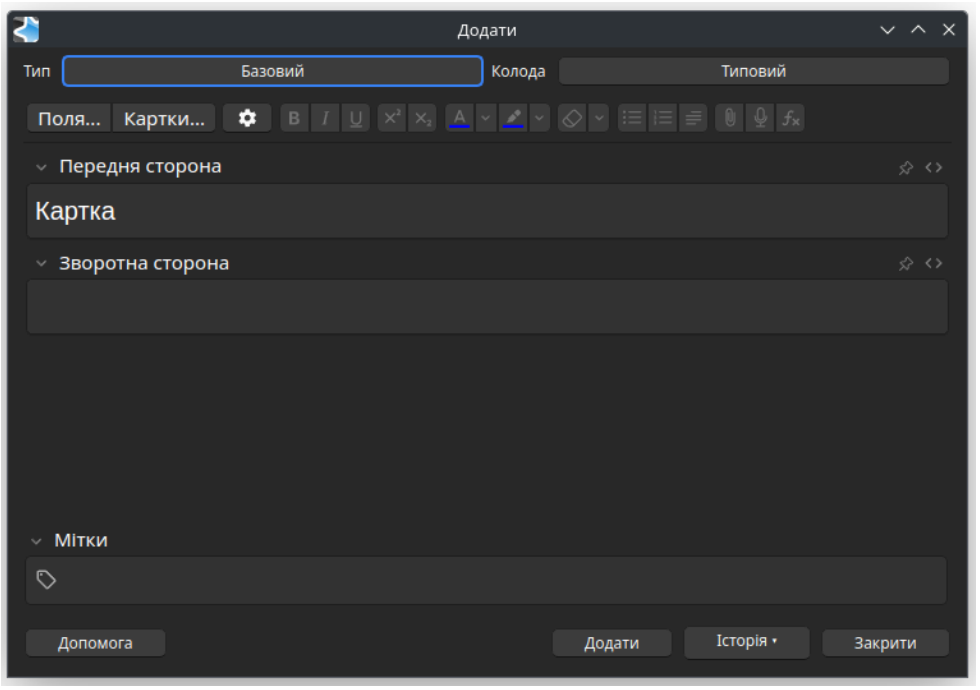


Рисунок 1.3 – Екранна форма додавання картки в Anki

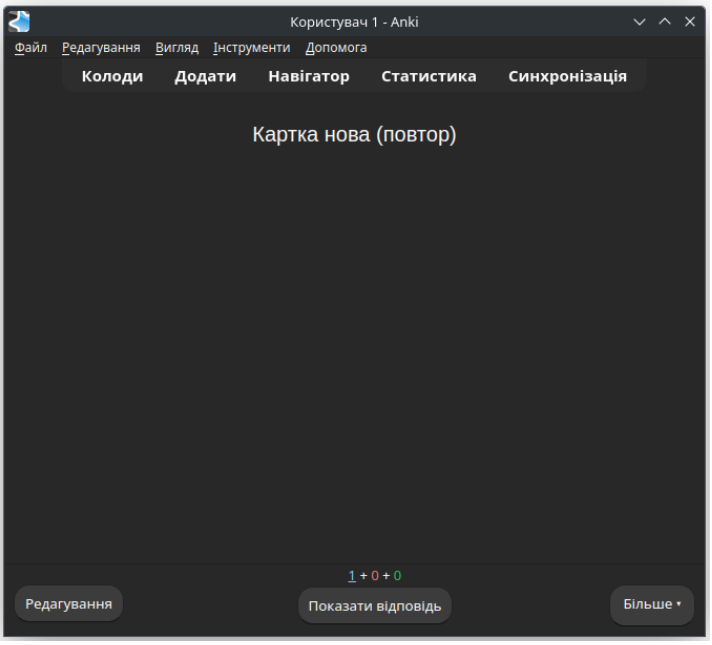


Рисунок 1.4 – Екранна форма повторення картки в Anki

SuperMemo [5] – оригінальне програмне забезпечення, що є першоджерелом алгоритмів сімейства SM (рисунок 1.5):

а) екранні форми: деревоподібна структура курсів, складні вікна налаштувань параметрів алгоритму, інтерфейс читання та поступового формування карток (Incremental reading);

б) підтримувані платформи: Windows;

в) аналіз: надає максимальну функціональність, але має перевантажений інтерфейс, що створює високий бар'єр входу для нових клієнтів, підтримує лише одну платформу і потребує купівлі (з версії 17)

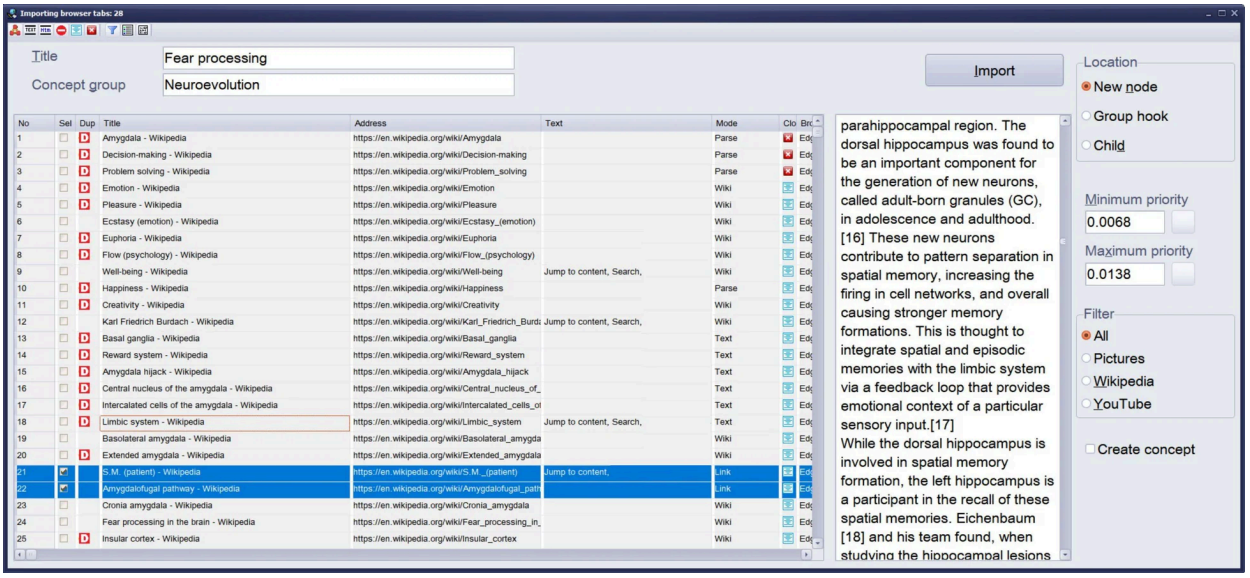


Рисунок 1.5 – Інтерфейс SuperMemo 19

SuperMemo.com (рисунок 1.6) [6] – відгалуження від оригінальної розробки SuperMemo, орієнтоване на масовість:

а) екранні форми: сучасний мінімалістичний та адаптивний інтерфейс, каталог готових курсів, спрощені вікна сеансів повторення (із підтримкою аудіо та зображень), інтерактивні дашборди відстеження прогресу, екрани діалогів із вбудованим ШІ (MemoChat);

б) підтримувані платформи: Web, iOS, Android;

в) аналіз: вирішує проблему високого порогу входу завдяки сучасному дизайну та прихованій складності алгоритмів. Проте основним недоліком є закрита екосистема з платною підпискою та фокус лише на готовому лінгвістичному контенті, що обмежує гнучкість користувача.

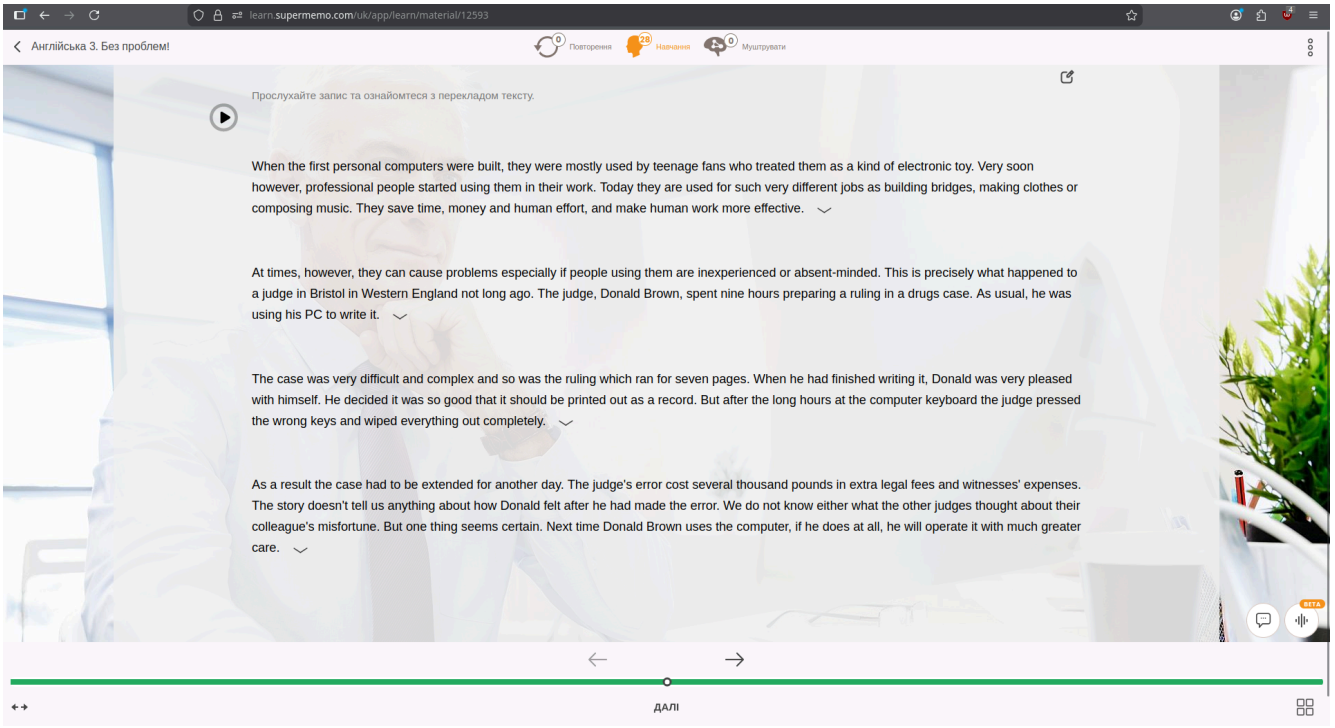


Рисунок 1.6 – Вікно навчання на https://learn.supermemo.com

Quizlet (рисунок 1.7) [7] – популярна платформа для вивчення матеріалу:

а) екранні форми: інтерфейси інтерактивних ігрових режимів («Заучування», «Тест», «Підбір»), візуально привабливий класичний режим перегортання двосторонніх карток, а також зручний редактор для швидкого масового введення навчальних термінів та їх визначень;

б) підтримувані платформи: Web, iOS, Android;

в) аналіз: продукт вирізняється високим рівнем залучення завдяки гейміфікованим механікам, набором вже готових наборів карток для підготовки до міжнародних екзаменів, можливості легко обмінюватися готовими наборами карток з іншими користувачами. Головним недоліком є те, що платформа історично орієнтована на короткострокове заучування, а повноцінні алгоритми довготривалого інтервального повторення обмежені та приховані за платною підпискою. Отже, незважаючи на привабливий користувацький інтерфейс та наявність елементів гейміфікації, використання Quizlet як основного інструменту для формування довготривалої бази знань є неефективним.

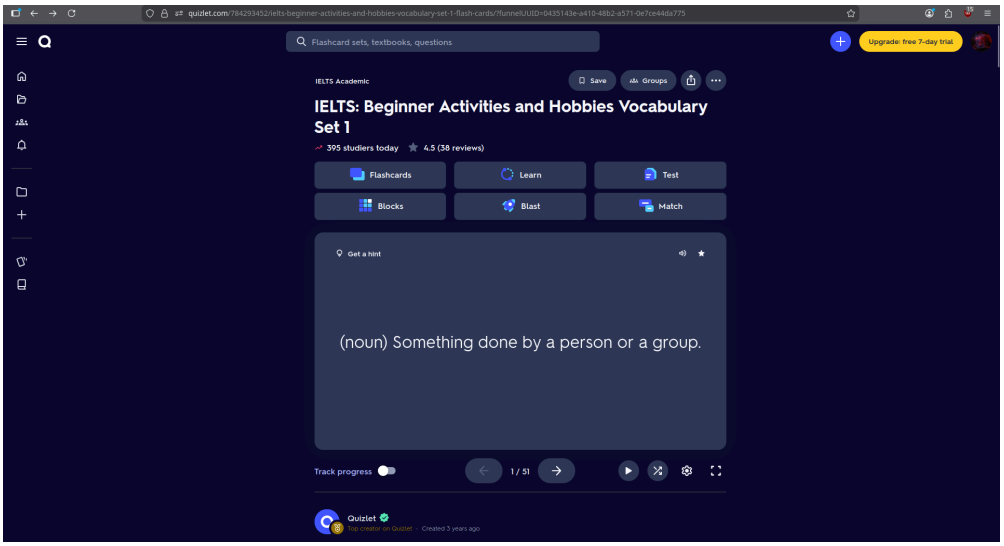


Рисунок 1.7 – Екранна форма вивчення в Quizlet

Проведений огляд та аналіз існуючих програмних рішень для інтервального повторення, зокрема Anki, SuperMemo та Quizlet, дозволив виявити їхні функціональні можливості та обмеження. Аналіз продемонстрував, що попри високу ефективність класичних алгоритмів (SM-2, SM-18), багато сучасних систем є або перевантаженими надлишковим функціоналом, що ускладнює процес навчання для початківців, або, навпаки, занадто спрощеними, обмежуючи можливості налаштування навчальних колод.

1.3 Постановка задачі проєктування

Метою роботи є розроблення веб-застосунку з використанням методики інтервального повторення, що забезпечує адаптивне навчання шляхом обчислення найсприятливішого інтервалу показу навчальної одиниці (картки) користувачеві.

Концепція полягає у розробленні автоматизованого програмного додатка, який повноцінно реалізує логіку алгоритму інтервального повторення SM-2 та надає користувачам інтерактивне середовище для створення та взаємодії з навчальним матеріалом. Такий підхід має на меті створити зручний, кросплатформний інструмент, доступний з будь-якого пристрою через веб-оглядач, що поєднує можливості гнучкого управління контентом,

імпортування готових баз знань, легкого доповнення баз знань користувацькими даними та глибокого аналізу успішності навчання.

Для однозначного розуміння принципів роботи розроблюваного програмного додатка та формалізації процесів засвоєння інформації необхідно визначити базові поняття, що використовуються в СІЗ. Ці поняття наведено у таблиці 1.1.

Таблиця 1.1 – Базові поняття властиві системам інтервального запам’ятовування

Поняття	Визначення	
Картка	Базова неподільна одиниця інформації в системі навчання; карткою може бути як одне слово, так і цілий абзац – головне щоб картка характеризувала атомарну частку в визначеній предметній області. Складається з двох взаємопов'язаних частин: лицьової сторони (містить поняття або запитання) та зворотної сторони (містить правильну еталонну відповідь-визначення), які також можуть доповнюватися супровідними мультимедійними даними.	
Колода	Логічне об'єднання навчальних карток, згрупованих за певною тематикою, навчальним курсом або рівнем складності; слугує основним контейнером для структурування інформації, імпортування та організації навчального процесу.	
Статус картки	Нова картка	Навчальна одиниця, яка була додана до бази даних системи, але ще жодного разу не демонструвалася користувачеві в процесі проходження тестування.
	Незріла картка	навчальна картка, яка перебуває на початковому етапі вивчення. Інтервал її повторення є коротким. Цей статус свідчить про те, що інформація ще не закріпилася у довгостроковій пам'яті та потребує регулярного і частого повторення
	Зріла картка	навчальна картка, яку користувач успішно пригадав кілька разів поспіль, унаслідок чого її інтервал повторення значно зріс. Досягнення карткою цього статусу є індикатором успішного перенесення інформації до довгострокової пам'яті

Продовження Таблиці 1.1

Поняття	Визначення	
Оцінка картки	Оцінка якості пригадування	суб'єктивний показник, який користувач самостійно визначає після відкриття зворотної сторони картки. Цей показник відображає ступінь легкості, з якою вдалося відтворити правильну відповідь
	Коефіцієнт легкості	динамічна змінна, закріплена за кожною окремою карткою, що визначає темп зростання інтервалу між її показами. Чим складнішою для користувача є картка, тим швидше зменшується цей коефіцієнт
	Інтервал повторення	Обчислений математичною моделлю проміжок часу між поточним та наступним показом картки під час навчальної сесії

Для реалізації цієї концепції та досягнення високої ефективності вирішення завдань (забезпечення високого рівня утримання уваги користувачів та покращення показників довгострокового засвоєння інформації) до проектованої системи висуваються вимоги щодо впровадження низки рішень:

а) вимоги щодо необхідних структурних змін:

– перехід до централізованої веб-архітектури: відмова від традиційної структури локальних настільних додатків на користь єдиного серверного програмного забезпечення з зрозумілим користувацьким інтерфейсом. Це дає змогу повністю усунути необхідність встановлення додаткового програмного забезпечення користувачем та забезпечує миттєву синхронізацію навчального прогресу;

– розподіл обчислювального навантаження: винесення ресурсомістких завдань, таких як розпакування та оброблення великих стиснених архівів із навчальними колодами, у фонові черги завдань. Це забезпечує безперебійну та швидку роботу інтерфейсу під час виконання важких операцій імпортування;

б) вимоги щодо необхідних функціональних змін:

– програмна реалізація алгоритму SM-2: веб-застосунок повинен

містити математичний модуль, який на основі оцінки якості пригадування динамічно перераховуватиме значення коефіцієнта легкості, кількість успішних повторень та розраховуватиме оптимальний час наступного показу картки;

– управління мультимедійним контентом: забезпечення можливості не лише створення текстових карток, але й прикріплення мультимедійних файлів (зображень та аудіозаписів) із подальшим їх збереженням у спеціалізованому файловому сховищі;

– аналітичний моніторинг та візуалізація: Впровадження функцій агрегації транзакційних даних для формування деталізованих графічних звітів (прогнозування навантаження, розподіл інтервалів, аналіз стану карток) з метою візуального підкріплення мотивації користувачів;

в) вимоги щодо необхідних конструктивних змін:

– оптимізація структури реляційної бази даних: проектування схеми даних, здатної ефективно обробляти велику кількість транзакційних записів (історії повторень), підтримувати швидкодію під час вибірки карток, термін повторення яких уже настав, та забезпечувати цілісність зв'язків між колодами, мультимедійними даними та обліковими записами;

– модульність програмного коду: використання сучасних архітектурних шаблонів для логічного розділення бізнес-логіки, доступу до даних та візуального оформлення, що забезпечує високу підтримуваність коду та можливість його подальшого масштабування;

г) вимоги до процесу тестування:

– повнота охоплення: тестуванню підлягають усі основні бізнес-процеси: реєстрація користувачів, створення та редагування колод, процес перегляду (review) карток, імпорт даних (Anki, CSV);

– коректність алгоритму: алгоритм SM-2 повинен демонструвати математичну точність при будь-яких вхідних оцінках користувача. Помилки в розрахунках коефіцієнту легкості або інтервалу є критичними.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕРВАЛЬНОГО ЗАПАМ'ЯТОВУВАННЯ

2.1 Загальна структура проєктованого веб-застосунку

Для забезпечення високої надійності, масштабованості та зручності супроводу програмного коду, архітектуру розроблюваного веб-застосунку побудовано на основі класичного шаблону проектування Модель-Вигляд-Контролер (англ. Model-View-Controller, MVC). Цей шаблон проектування передбачає чітке розділення системи на три взаємопов'язані складові, кожна з яких відповідає за окремий аспект оброблення даних та взаємодії з користувачем. Такий підхід дає змогу ізолювати бізнес-логіку від користувацького інтерфейсу, що суттєво спрощує процес тестування та модернізації окремих підсистем без ризику порушення цілісності всього програмного продукту.

Відповідно до обраного шаблону, структура веб-застосунку побудована таким чином:

1) модель (model): цей рівень архітектури відповідає за інкапсуляцію бізнес-логіки та взаємодію з базою даних. У контексті розроблюваного веб-застосунку об'єктно-реляційне відображення (ORM) перетворює записи з таблиць (користувачі, колоди, навчальні картки, журнали навчання) у програмні об'єкти. Моделі не лише вилучають чи зберігають інформацію, але й містять правила валідації даних та визначають складні взаємозв'язки (наприклад, належність безлічі карток до однієї колоди або зв'язок історії повторень з конкретним обліковим записом);

2) вигляд (view): цей компонент відповідає виключно за візуалізацію даних, отриманих від Моделі, та формування клієнтського веб-інтерфейсу. Для генерації сторінок використовується механізм динамічної шаблонізації, який дозволяє вбудовувати керуючі конструкції безпосередньо в розмітку сторінки. Представлення реалізує багатомовний, адаптивний інтерфейс, який коректно

відображається на екранах різної роздільної здатності. Воно не містить жодної обчислювальної логіки, лише приймає підготовлені масиви даних (наприклад, статистичні показники або текст питання картки) і виводить їх у графічному форматі;

3) контролер (controller): виконує роль центрального диспетчера системи. Контролер перехоплює вхідні запити від користувача (натискання кнопок, відправлення форм оцінювання), обробляє їх, звертається до відповідної Моделі для оновлення чи отримання даних, після чого передає результат у Вигляд для оновлення екрана. Саме на рівні контролерів відбувається ініціалізація алгоритму інтервального повторення, розрахунок нових інтервалів та прийняття рішень щодо маршрутизації користувача.

Для наочного відображення взаємозв’язків між складовими цього шаблону, наведено діаграму на рисунку 2.1.

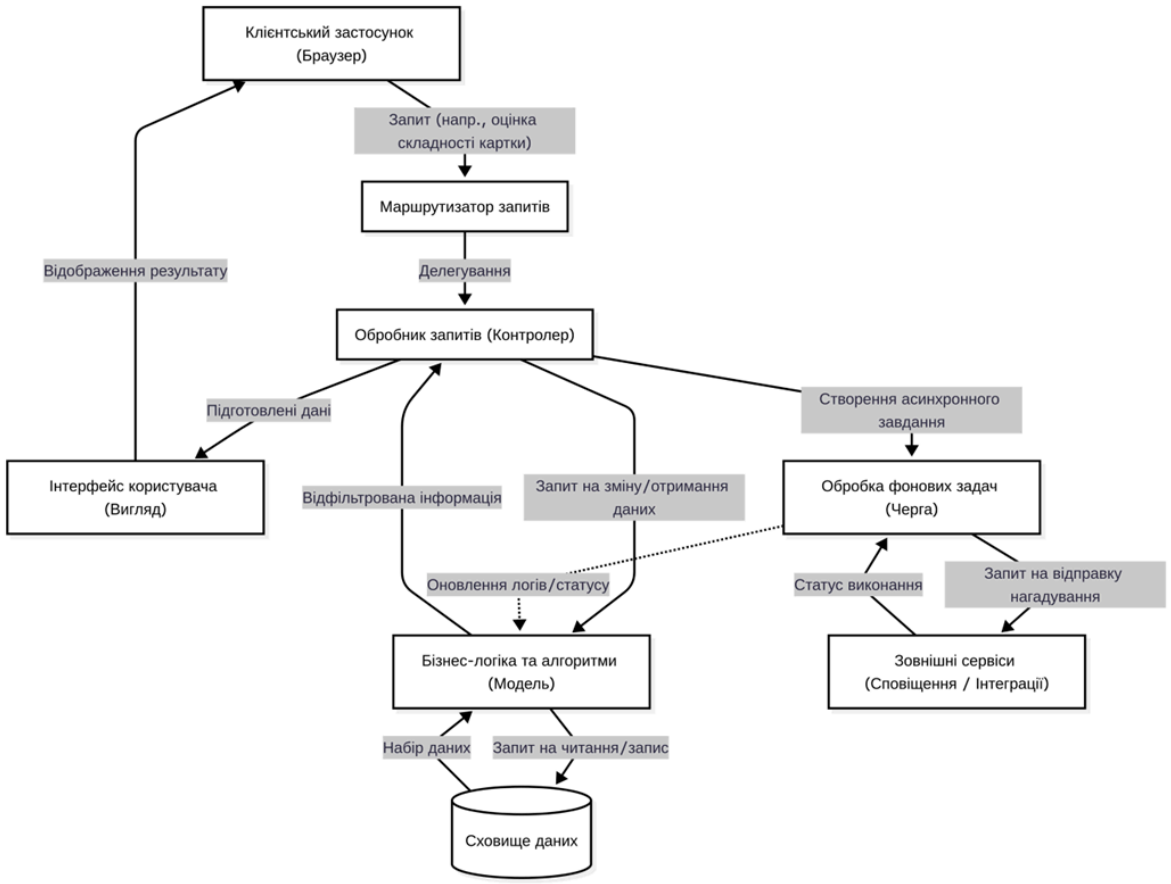


Рисунок 2.1 – Схема взаємодії складових шаблону “Модель-Вигляд-Контролер”

Також для імплементації функціоналу спроектовано такі програмні модулі:

– маршрутизатор (router) – перший рівень серверного середовища, який виконує роль вхідного шлюзу. Маршрутизатор аналізує URL-адресу мережевого запиту та його метод (GET, POST тощо), після чого спрямовує цей запит до відповідного модуля управління (Контролера). Він також забезпечує базовий рівень безпеки, перевіряючи наявність прав доступу (ідентифікаційних токенів або сесій) перед тим, як дозволити виконання чутливих операцій;

– модуль авторизації – блок управління обліковими записами, який інкапсулює логіку реєстрації нових учасників, перевірки автентичності введених облікових даних, керування життєвим циклом сесій та відновлення втраченого доступу. Модуль гарантує, що кожен користувач має ізольований доступ виключно до власних навчальних матеріалів та статистики;

– модуль навчального контенту – основний компонент для управління інформаційною базою системи. Забезпечує виконання повного циклу операцій (створення, читання, оновлення, видалення) над колодами та навчальними картками. Цей модуль також відповідає за оброблення завантажених користувачем файлів (аудіо та зображень), здійснюючи їх перевірку на допустимість форматів та делегуючи їх збереження до файлового сховища;

– модуль тестування – ключовий транзакційний компонент системи, який керує процесом активного навчання. Під час ініціалізації сесії він формує вибірку карток, що потребують повторення, керує їх почерговим відображенням та збирає зворотний зв'язок (оцінки якості пригадування). Після отримання оцінки модуль передає керування математичному ядру для перерахунку інтервалів;

– модуль аналітики – відповідає за збирання, агрегацію та математичне оброблення транзакційних даних для формування статистичних звітів. Модуль аналізує історію повторень, групує картки за їхніми станами (нові, незрілі, зрілі) та генерує масиви даних, необхідні для побудови інтерактивних графіків і прогнозування навчального навантаження на наступні календарні періоди;

– підсистема оброблення архівів (фонові процеси) – скільки процес імпортування готових навчальних баз (у форматах стиснених даних) потребує

значних обчислювальних ресурсів та часу на розпакування, цю задачу винесено у відокремлену асинхронну підсистему (чергу завдань). Вона функціонує у фоновому режимі, послідовно вилучаючи мультимедійні файли та текст із завантажених архівів і зберігаючи їх у базу даних. Таке архітектурне рішення запобігає блокуванню користувацького інтерфейсу під час виконання важких серверних операцій.

– ядро обчислення SM-2 – ізольований алгоритмічний компонент, що містить математичну логіку веб-застосунку інтервального повторення. Ядро приймає на вхід поточні параметри картки (коефіцієнт легкості, кількість попередніх повторень) та оцінку користувача, після чого повертає обчислений новий часовий інтервал до наступного показу.

– модуль доступу до даних (ORM) – програмний прошарок, який абстрагує бізнес-логіку від специфіки мови структурованих запитів (SQL). Модуль дозволяє контролерам та фоновим процесам взаємодіяти зі сховищами інформації за допомогою інтуїтивних об'єктно-орієнтованих методів, автоматично генеруючи безпечні запити до реляційної бази даних.

– сховища даних – фізичний рівень збереження інформації, що концептуально поділений на дві частини. Реляційна база даних гарантує структуроване та надійне збереження текстового контенту, налаштувань профілів та числових показників повторень, забезпечуючи високу швидкість вибірки даних. Файлове сховище, у свою чергу, оптимізоване для збереження неструктурованих бінарних даних (статичних ресурсів та завантаженого користувачами мультимедійного вмісту), забезпечуючи їх швидку віддачу клієнтському середовищу за прямими посиланнями.

– користувацький веб-інтерфейс (клієнтське середовище) – єдиною точкою взаємодії користувача із системою є інтерактивний веб-інтерфейс, який завантажується у веб-оглядачі. Цей компонент формує мережеві запити на основі дій користувача (реєстрація, додавання матеріалу, проходження тестування) та відображає отримані від сервера відповіді. Використання сучасних веб-технологій дозволяє реалізувати клієнтське середовище за принципом "тонкого клієнта", де

всі ресурсомісткі обчислення винесено на серверну сторону, а браузер відповідає виключно за рендеринг графіки та відтворення мультимедійних файлів.

Взаємодію модулів між собою зображено на рисунку 2.2.

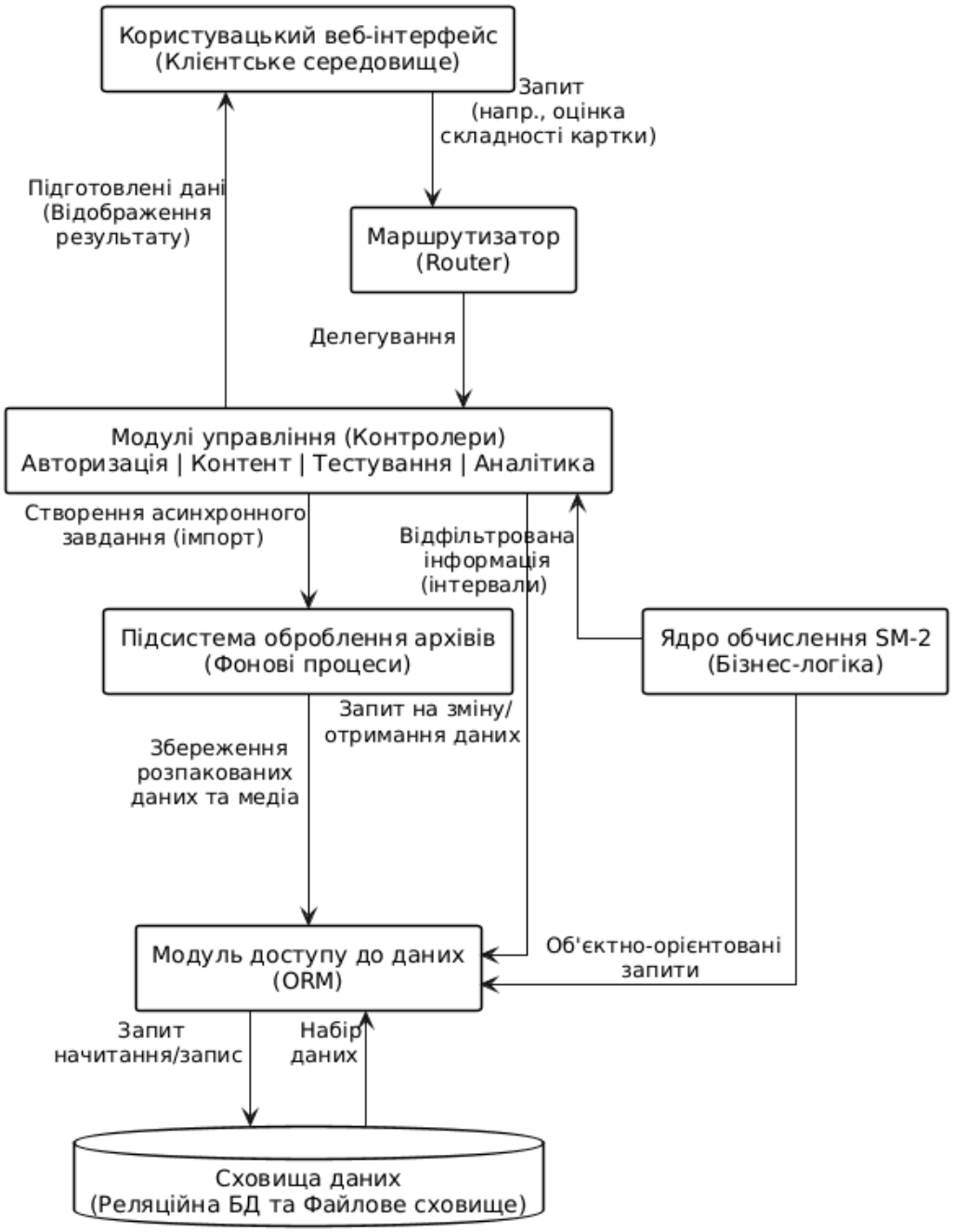


Рисунок 2.2 – Схема взаємодії окремих модулів програми

Отже, спроєктована багаторівнева архітектура системи забезпечує чіткий розподіл функцій.

2.2 Алгоритмічне забезпечення веб-застосунку

Основним обчислювальним ядром розроблюваної системи, яке відрізняє її від звичайних програм для створення електронних нотаток, є алгоритм інтервального повторення. Його головна задача – математично точно спрогнозувати момент часу, коли ймовірність забування інформації користувачем наближається до критичної межі, і саме в цей момент ініціювати повторний показ матеріалу.

Для реалізації цієї логіки у програмному додатку імплементовано алгоритм SuperMemo-2 (SM-2). Вибір саме цієї версії зумовлений її оптимальним балансом між обчислювальною простотою та високою ефективністю: на відміну від новіших ресурсомістких ітерацій (SM-15+), SM-2 не потребує збирання великих масивів історичних даних перед початком роботи і чудово підходить для реляційних баз даних веб-застосунків, виконуючи основну роль – позбавляє від необхідності вчити все підряд щодня, залишаючи в роботі лише те, що користувач ось-ось забуде.

Математична модель алгоритму оперує чотирма основними змінними:

– q (Quality);

– EF (E-Factor);

– n (Number of repetitions);

– I (Interval).

Ці змінні зберігаються в базі даних для кожної окремої картки і динамічно оновлюються після кожної навчальної сесії. Нижче детальніше описано роль кожної зі змінних, звідки вони отримуються та як і за якими формулами обчислюються.

q (Quality) – оцінка якості пригадування. Це дискретна величина, яку вводить користувач після спроби згадати відповідь. Набуває цілочислових значень від 0 до 5, позначення яких описано в таблиці 2.1.

Таблиця 2.1 – Можливі варіанти змінної q

Оцінка	
5	ідеальне пригадування без жодних вагань
4	правильна відповідь після незначних роздумів
3	правильна відповідь, згадана зі значними зусиллями
2	неправильна відповідь, але правильна здається знайомою після її відображення
1	неправильна відповідь, правильна згадується насилу
0	повне забування інформації

EF (E-Factor) – коефіцієнт легкості – раціональне число, що визначає темп зростання інтервалу між повтореннями. Для нових карток початкове значення завжди дорівнює 2.5.

n (Number of repetitions) – лічильник кількості успішних повторень картки підряд (безперервна серія)

I (Interval) — обчислений інтервал часу до наступного показу, виражений у календарних днях

Після отримання оцінки q від користувача, веб-застосунок першочергово перераховує коефіцієнт легкості за формулою:

$$EF_{new} = EF_{old} + (0.1 - (5 - q) \times (0.08 + (5 - q) \times 0.02)) \qquad (2.1)$$

Згідно з цією формулою, якщо користувач ставить високі оцінки (4 або 5), коефіцієнт легкості залишається незмінним або дещо зростає. Якщо ж оцінка становить 3 або менше, коефіцієнт зменшується, що вказує системі на підвищену складність картки. Важливим алгоритмічним обмеженням є те, що значення EF

ніколи не може опускатися нижче встановленого мінімуму: якщо обчислене значення $EF_{new} < 1.3$, веб-застосунок примусово встановлює $EF = 1.3$.

Наступним кроком є обчислення нового інтервалу I та оновлення лічильника n. Логіка розрахунку залежить від успішності пригадування (порогове значення $q = 3$):

– якщо $q < 3$ (забування або груба помилка): серія успішних повторень переривається. веб-застосунок встановлює $n = 1$, а інтервал скидається до базового значення: $I(n) = 1$ день. Картка повертається до стадії активного вивчення (незріла картка);

– якщо $q \geq 3$ (успішне пригадування), інтервал зростає залежно від поточного номера повторення:

1) при $n = 1$ (перше успішне повторення): $I(1) = 1$;

2) при $n = 2$ (друге успішне повторення): $I(2) = 6$;

3) при $n > 2$ (подальші повторення): інтервал обчислюється шляхом множення попереднього інтервалу на коефіцієнт легкості з подальшим округленням до найближчого цілого числа:

$$I(n) = [I(n - 1) \times EF]. \quad (2.2)$$

Базовий алгоритм SM-0 базувався на спрощених статичних інтервалах для запам'ятовування, тоді як SM-1 запровадив перші елементи адаптації, що дозволяли коригувати графік повторень на основі факту забування матеріалу. У свою чергу, SM-2 став ключовим етапом еволюції, запропонувавши інтеграцію коефіцієнта легкості, що дозволило системі гнучко реагувати на суб’єктивну складність навчальної одиниці, перетворюючи механічне повторення на адаптивний процес.

Цей перехід відображає науковий шлях від спрощених підходів до створення систем, здатних враховувати індивідуальні особливості пам’яті користувача, що і стало передумовою для вибору SM-2 як оптимального рішення для розроблюваної системи. На рисунку 2.3 наведено схему алгоритму, що ілюструє логіку обчислення параметрів, які забезпечили ефективність версії SM-2.

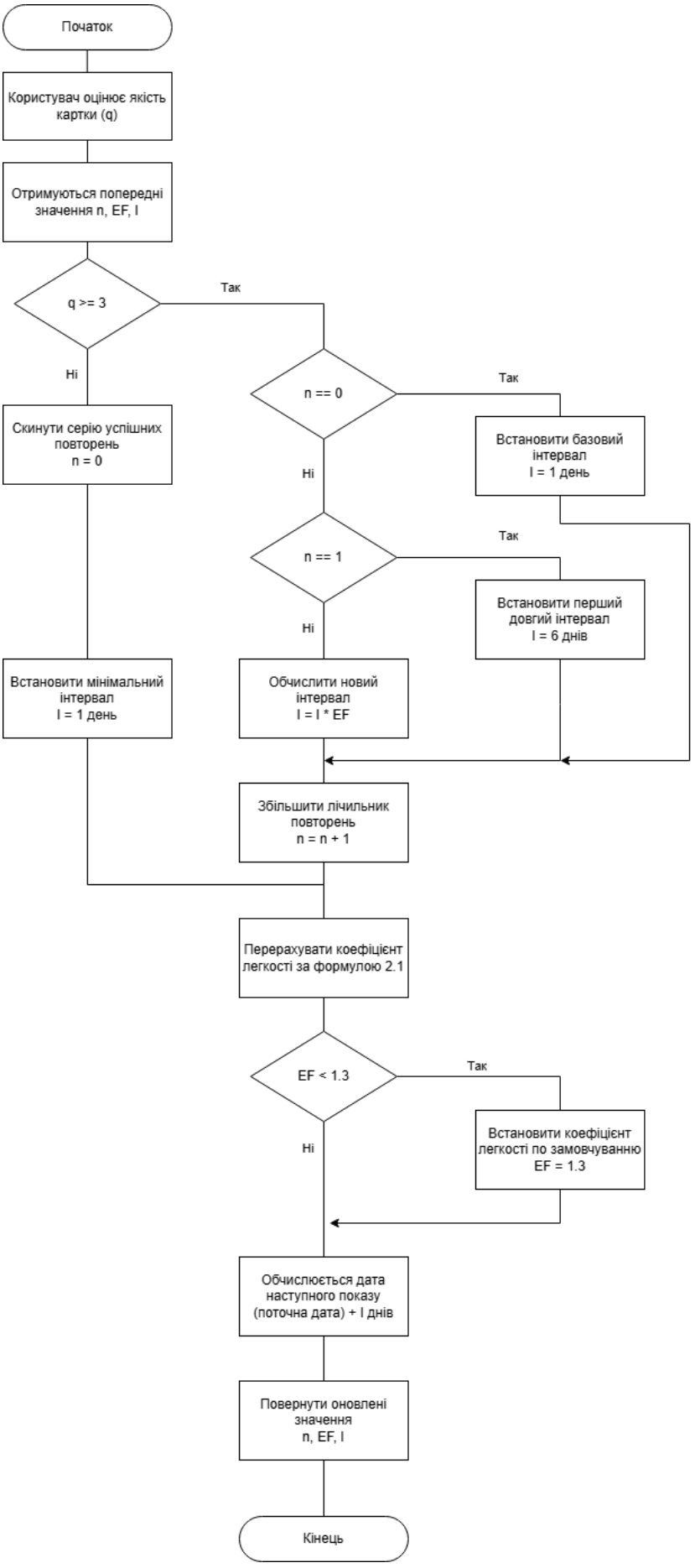


Рисунок 2.3 – Схема алгоритму SM-2

Отже, алгоритмічним ядром розробленого веб-застосунку виступає адаптивна математична модель SM-2, імплементація якої забезпечує персоналізований розрахунок інтервалів повторення та ефективне переведення знань у довготривалу пам'ять користувача.

2.3 Інформаційне забезпечення веб-застосунку

Ефективне функціонування системи інтервального повторення залежить від надійності, швидкодії та цілісності підсистеми збереження даних [8]. Інформаційне забезпечення розроблюваного веб-застосунку проектується з урахуванням необхідності оброблення як строго структурованих реляційних даних (користувачі, картки, колоди, логування повторень), так і неструктурованих бінарних об'єктів (мультимедійні файли навчальних карток). Відповідно, сховище даних фізично та логічно розділено на реляційну базу даних та файлову систему сервера.

2.3.1 Опис вхідної та вихідної інформації

Функціонування предметної області веб-застосунку передбачає постійний обмін інформацією між користувачем (клієнтським середовищем) та сервером. Для формалізації потоків даних необхідно чітко розмежувати вхідну та вихідну інформацію.

Вхідна інформація – це масив даних, що надходить до системи ззовні для подальшого збереження, оброблення або ініціалізації обчислювальних процесів. До неї належать:

– реєстраційні та авторизаційні дані: ідентифікатори користувачів (адреси електронної пошти), хешовані паролі та маркери сесій;

– навчальний контент (структурований): текстові масиви, що формують лицьову (питання) та зворотну (відповідь) сторони карток, метадані колод (назви, описи, теги). Навчальний контент (неструктурований): імпортовані архіви

стиснених даних формату .apkg, а також окремі медіафайли (зображення, аудіозаписи), що завантажуються користувачем;

– транзакційні дані тестування: ключовий потік даних, що генерується в процесі навчання — оцінки якості пригадування (q), які користувач виставляє кожній картці (значення від 0 до 5).

Вихідна інформація – це результат оброблення вхідних даних алгоритмами системи, який повертається користувачеві у придатному для сприйняття вигляді. До неї належать:

– динамічні черги тестування: відфільтровані масиви карток, у яких розрахункова дата наступного показу менша або дорівнює поточній календарній даті;

– оновлені параметри математичної моделі: перераховані після кожної відповіді значення коефіцієнта легкості (EF), кількості повторень (n) та нового інтервалу (I). Аналітичні та статистичні звіти: агреговані дані щодо кількості вивчених карток, прогнозованого навантаження на наступні дні (у вигляді масивів точок для побудови графіків) та розподілу карток за статусом зрілості.

2.3.2 Концептуальне інфологічне проєктування

Метою концептуального інфологічного проектування є створення абстрактної моделі предметної області, незалежної від конкретної системи керування базами даних (СКБД). На цьому етапі визначаються ключові сутності системи та типи зв'язків між ними.

Виділено чотири головні інформаційні сутності:

– користувач (user): головна ідентифікуюча сутність системи. Володіє персональними налаштуваннями та навчальним матеріалом;

– колода (deck): логічний контейнер для групування навчального матеріалу за темами чи предметами. Належить конкретному користувачу;

– картка (card): базова одиниця інформації. Належить конкретній колоді. Містить текст, посилання на мультимедіа та поточні параметри алгоритму SM-2;

– історія повторень (review): транзакційна сутність, що фіксує кожну спробу відповіді користувача. Належить конкретній картці.

Між зазначеними сутностями існують суворі ієрархічні зв'язки типу «один до багатьох» (1:M). Один користувач може створити багато колод, але кожна колода належить лише одному користувачу. Одна колода містить безліч карток, а одна картка протягом свого життєвого циклу генерує безліч записів в історії повторень.

2.3.3 Проєктування глобальної даталогічної моделі даних

Для візуалізації спроектованої глобальної даталогічної моделі даних розроблено ER-діаграми (англ. Entity-Relationship diagram) зображену на рисунках 2.4 – 2.6.

Рисунок 2.4 – ER-діаграма основних сутностей

Рисунок 2.5 – ER-діаграма зі службовими таблицями для користувачів

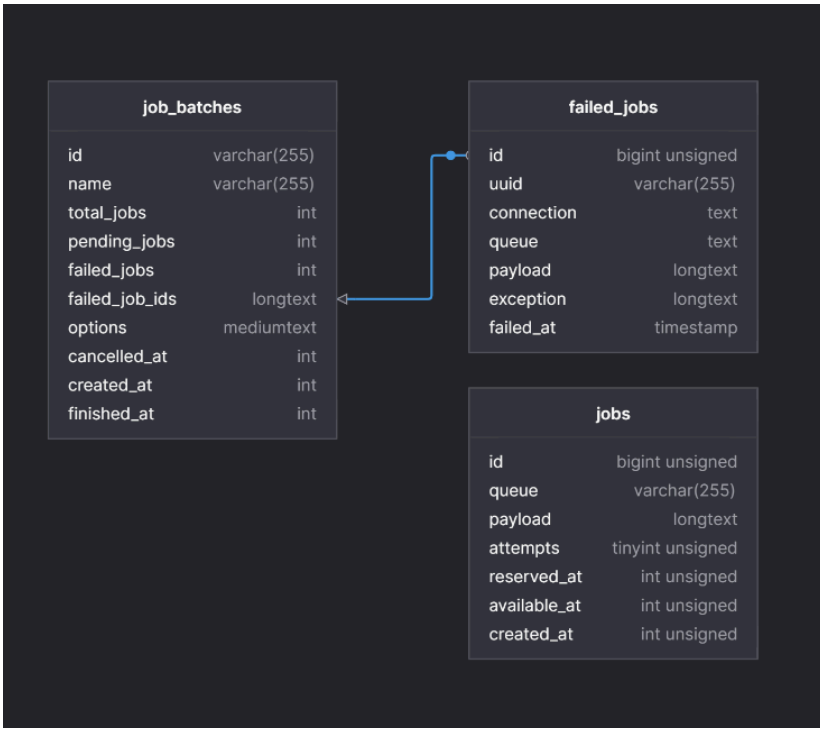


Рисунок 2.6 – ER-діаграма з таблицями фонових процесів

Спроектована глобальна даталогічна модель даних дотримується 3 принципів нормалізації: усі атрибути є атомарними (1NF), кожна таблиця має простий первинний ключ (2NF), всі неключові атрибути залежать виключно від первинного ключа (3NF).

3 РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕРВАЛЬНОГО ЗАПАМ'ЯТОВУВАННЯ

3.1 Реалізація програмного забезпечення

Програмна реалізація веб-застосунку базується на використанні сучасного TALL-стеку (Tailwind CSS, Alpine.js, Laravel, Livewire), який забезпечує високу продуктивність та швидкість ітерацій розробки. Вибір технологій обумовлений необхідністю побудови адаптивного веб-інтерфейсу з низькою затримкою відгуку.

Кожен компонент стеку виконує чітко визначену роль:

– Laravel [9] (Backend Framework) – слугує серверним ядром системи. Забезпечує надійну роботу з базами даних (Eloquent ORM), авторизацію, валідацію даних та управління фоновими чергами (Jobs). Вибір Laravel зумовлений його розвиненою екосистемою, що значно пришвидшує розробку стандартизованих бізнес-процесів;

– Livewire [10]: ключовий компонент стеку, що дозволяє створювати складні інтерактивні інтерфейси, використовуючи виключно PHP. Завдяки Livewire, логіка оновлення карток та роботи алгоритму SM-2 відбувається на сервері, а клієнтська частина отримує оновлені фрагменти DOM-структури. Це дозволяє уникнути написання та підтримування громіздкого API (наприклад, REST або GraphQL) для кожного дрібного оновлення стану інтерфейсу;

– Alpine.js [11]: використовується для легких, декларативних DOM-маніпуляцій, які не потребують серверного запиту: відкриття модальних вікон, анімація приховування відповідей або миттєва реакція на перемикачі (toggle). Alpine.js є максимально компактним та ідеально доповнює Livewire;

– Tailwind CSS [12]: дозволяє реалізувати адаптивний дизайн за принципом "utility-first". Замість написання великих файлів CSS-стилів, дизайн будується шляхом комбінування атомарних класів, що гарантує високу швидкість рендерингу сторінок та легкість підтримки єдиного корпоративного стилю.

Вибір середовища розроблення описано нижче:

Для забезпечення ідентичності середовищ розробки та сервера, застосовано технологію контейнеризації Docker. Конфігурація docker-compose включає три ізольовані сервіси:

– Web Server (PHP-FPM/Nginx)[13]: виконує програмну логіку на базі Laravel;

– Database (MySQL)[14]: забезпечує надійне збереження реляційних даних;

– Queue Worker: фоновий процес, що виконує асинхронні завдання з імпорту та розпакування архівів.

Веб-застосунок організовано за стандартом PSR-4, що гарантує строге розділення рівнів системи:

а) app/Models – містить класи сутностей для роботи з даними (ORM):

– Card – описує поля картки, зв’язок з колодою та описує процес повторення картки з використанням SM-2;

– Deck – описує поля колоди, та зв’язок з користувачем;

– Review – описує поля повторення картки, які потрібні для логування взаємодії користувача із карткою;

– User – описує основні поля користувача та логіку хешування паролю;

б) app/Http/Controllers – інкапсулює бізнес-логіку та диспетчеризацію запитів:

– CardController – описує створення, збереження, оновлення та видалення карток;

– DeckController – описує створення, збереження, оновлення та видалення колод;

– ReviewController – описує процес повторення картки та зберігає логування повторень;

– StatController – описує виведення статистики користувача;

в) app/Jobs – логіка фонових завдань:

– ProcessDeckImport – описує фоновий процес імпортування колоди з формату csv чи apkg;

г) resources/views – файли користувацького інтерфейсу (View), побудовані на базі шаблонізатора Blade, містять компоненти, вигляди та шари для відображення інтерфейсу веб-застосунку.

Повну файлову структуру веб-застосунку зображено на рисунку 3.1.

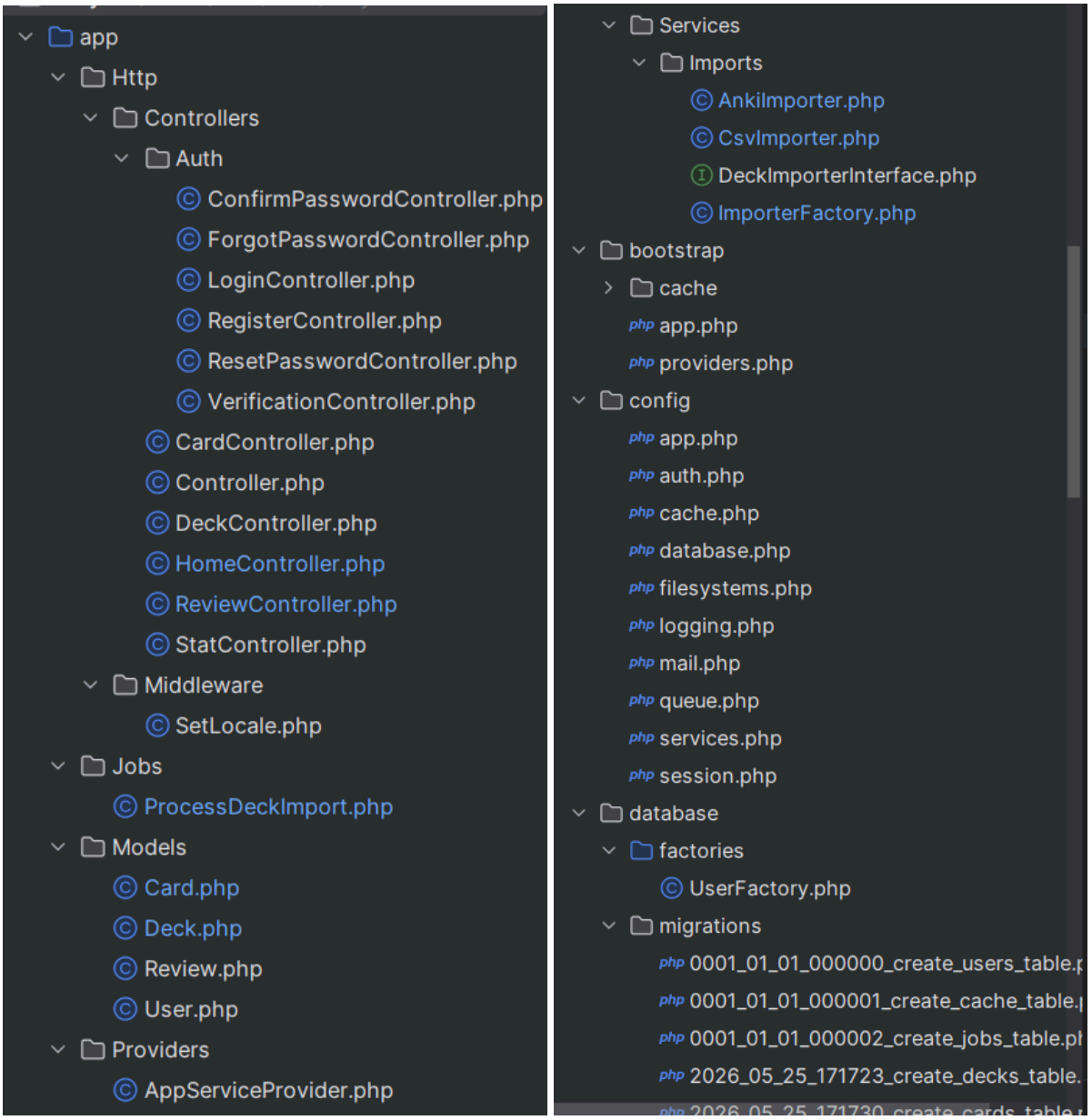


Рисунок 3.1 – Файлова структура розробленого веб-застосунку

Спрощену UML-діаграму основних класів зображено на рисунку 3.2, більш деталізована структура зображена у додатку А.

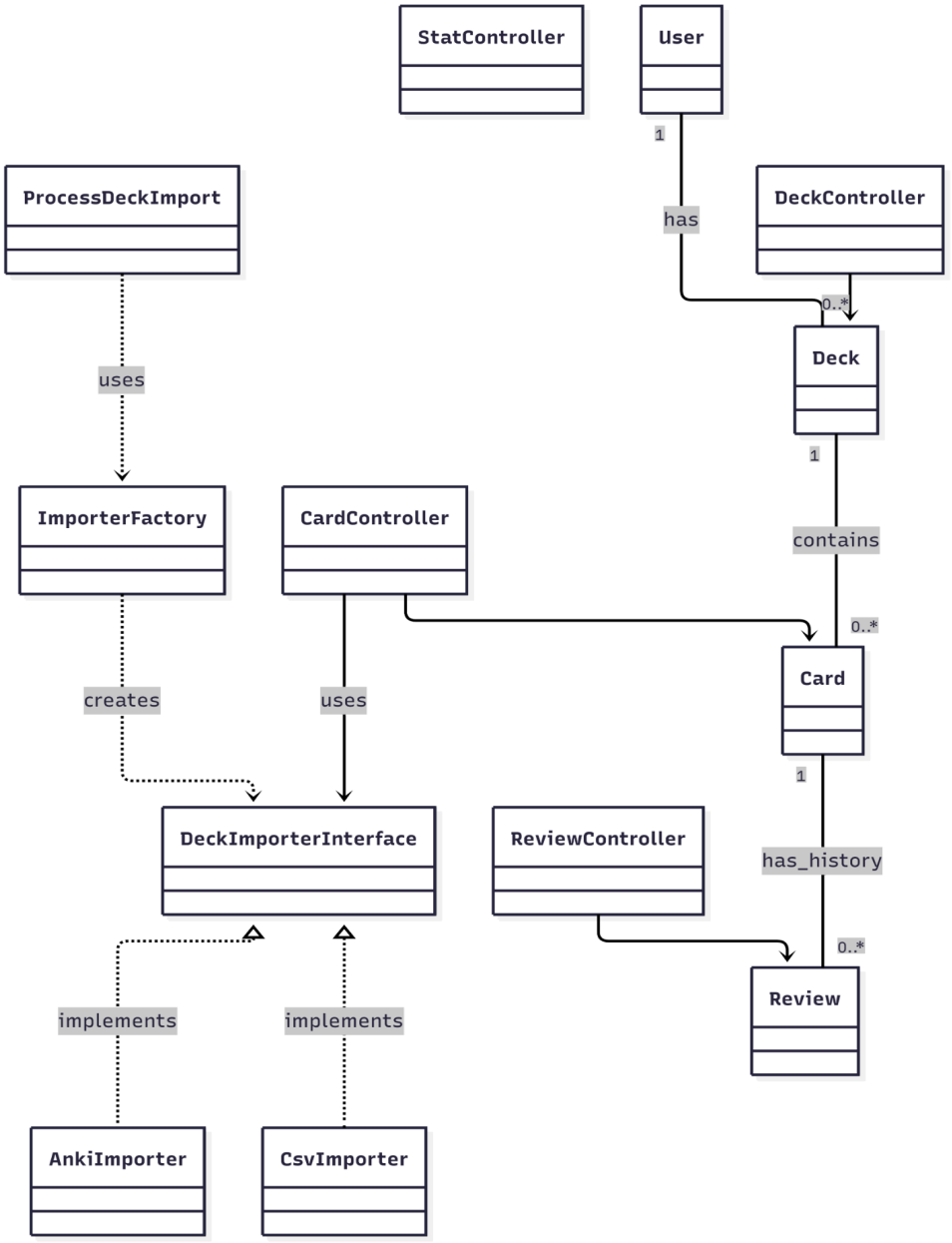


Рисунок 3.2 – Спрощена структура класів розробленого веб-застосунку

У додатку Б наведено програмний код логіки обчислення часових інтервалів та коефіцієнтів легкості, яка виконується в межах моделі Card. У даному додатку продемонстровано методику маніпуляції станами об'єкта, оброблення граничних значень коефіцієнта легкості ($EF < 1.3$) та процес планування дати наступного показу, що забезпечує коректність роботи алгоритму згідно з математичною моделлю.

Операції імпортування навчальних колод з архівних файлів формату csv, та відкритого формату apkg було винесено у фоновий процес. Програмний код, що

відповідає за розпакування apkg архівів, асинхронне парсинг-опрацювання структури бази даних та пакетне збереження мультимедійних даних, наведено у додатку В. Реалізація цього модуля демонструє роботу з чергами завдань Laravel (Job Queues), що дозволяє уникнути блокування основного потоку виконання під час імпорту великих масивів даних.

3.2 Інтерфейс користувача

Інтерфейс веб-застосунку спроектовано з урахуванням принципів мінімалізму та фокусування на навчальному процесі. Головною метою було створення інтерфейсу, де користувач отримує максимально швидкий відгук при взаємодії з навчальними матеріалами.

Інтерфейс побудовано на основі компонентного підходу. Використання Tailwind CSS дозволило реалізувати систему "функціональність-найголовніше", де кожна кнопка, картка або елемент навігації мають консистентний вигляд. Для забезпечення адаптивності було розроблено систему гнучких сіток, що дозволяє інтерфейсу коректно масштабуватися на пристроях з різною роздільною здатністю – від мобільних телефонів до широкоформатних моніторів, що зображено на рисунках 3.3 – 3.5.

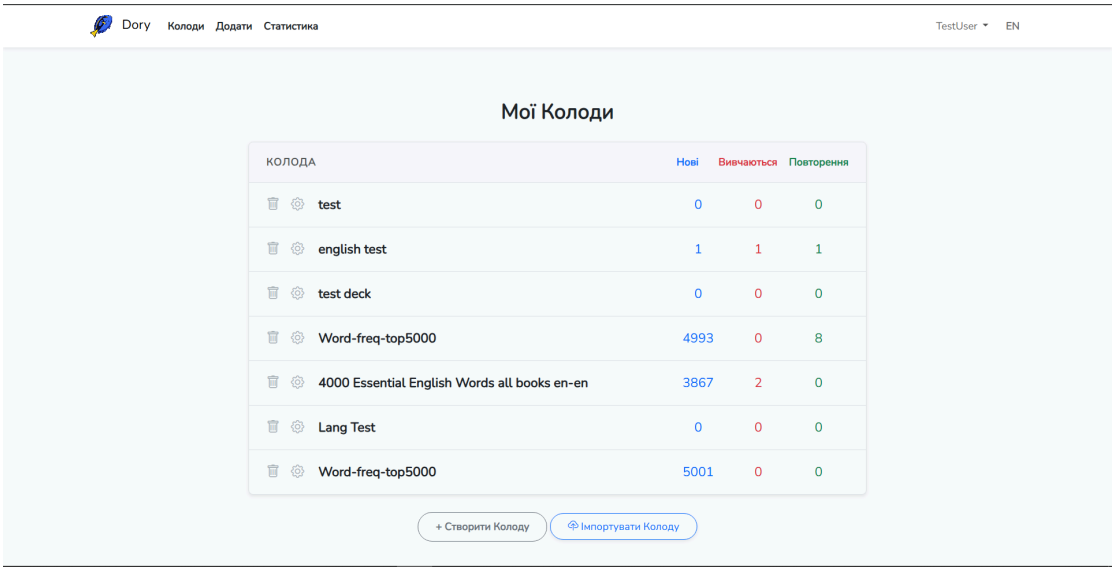


Рисунок 3.3 – Вигляд головної сторінки на лептопі

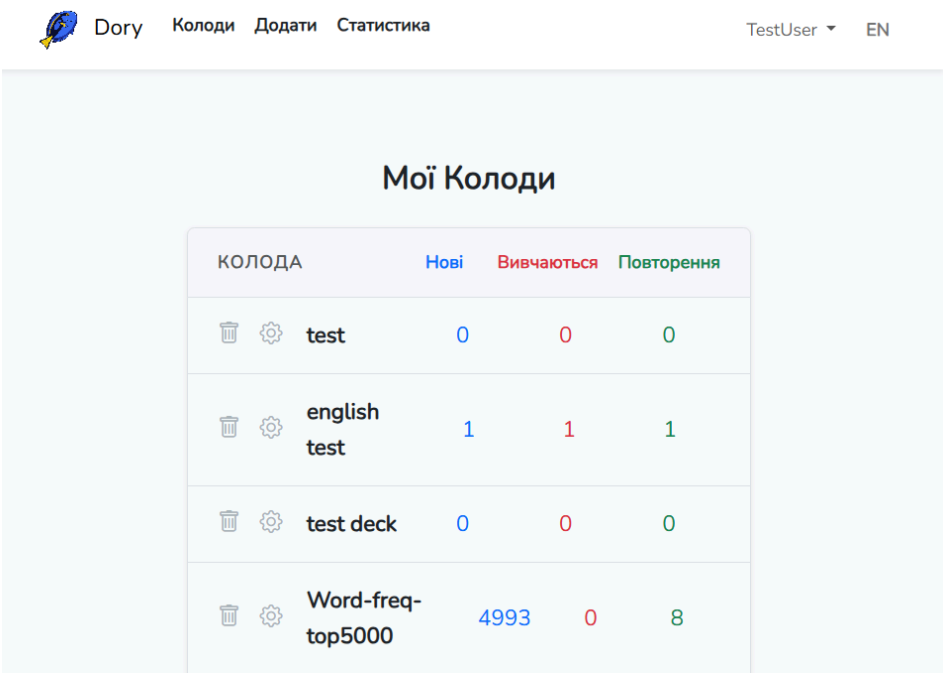


Рисунок 3.4 – Вигляд головної сторінки на планшеті

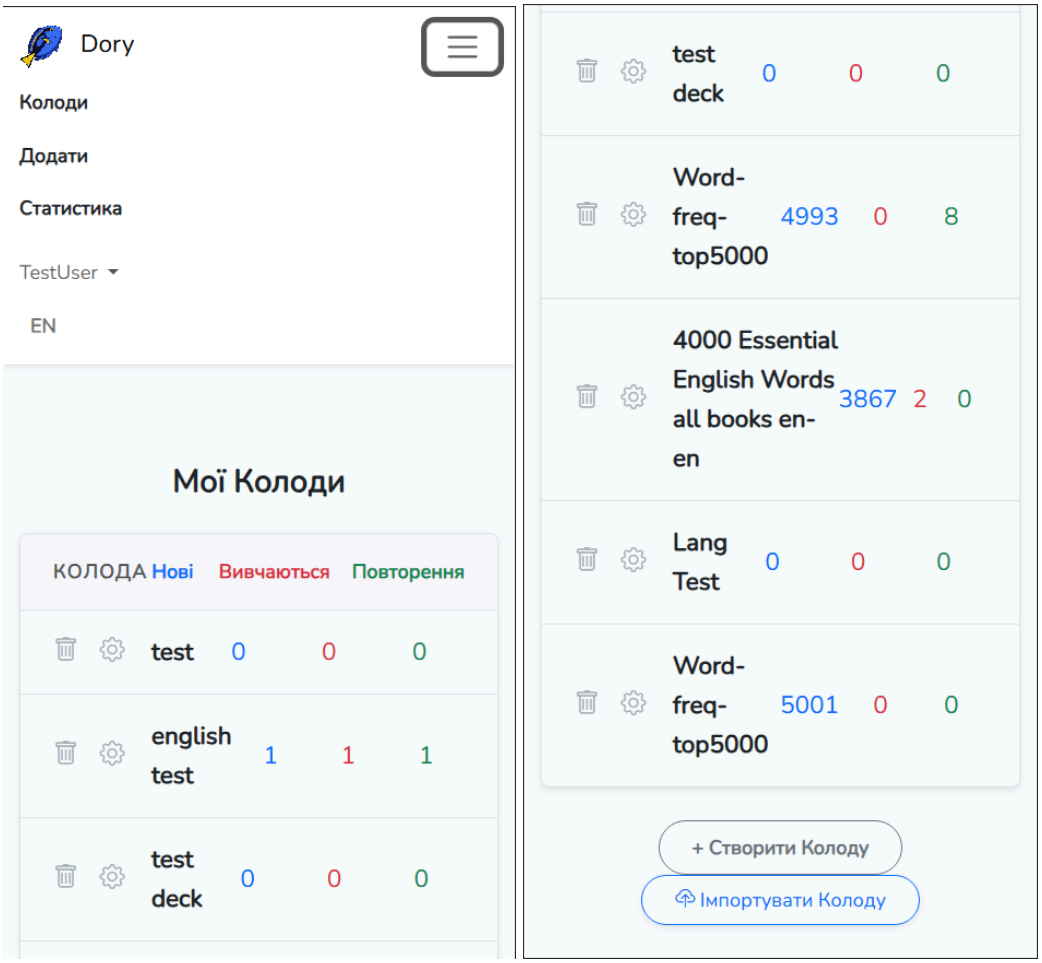


Рисунок 3.5 – Вигляд головної сторінки на смартфоні

У додатку Г наведено програмний код компоненту для оцінки картки.

У загальному було розроблено такі форми користувацького інтерфейсу:

– вікно реєстрації;

– вікно логіну;

– вікно підтвердження паролю;

– вікно-привітання нового користувача;

– вікно колод – є домашнім-вікном (зображено на рисунках 3.3–3.5). Містить модальні вікна створення колоди (зображено на рисунку 3.6) та імпорту колоди (зображено на рисунку 3.7);

– вікно створення картки (зображено на рисунку 3.8) – на цьому вікні присутній функціонал додавання тексту та медіафайлів (аудіофайлів та зображень) до передньої і зворотньої сторін картки;

– вікно повторення картки – в вікні показано передню сторону картки, яку користувач повинен пригадати (зображено на рисунку 3.9) коли він обирає варіант розкрити картку відображається задня сторона де можна побачити відповідь та оцінити картку (зображено на рисунку 3.10);

– вікно статистики (зображено на рисунку 3.11) – можна обрати статистику по всім колодам або по одній обраній з доступних колод.

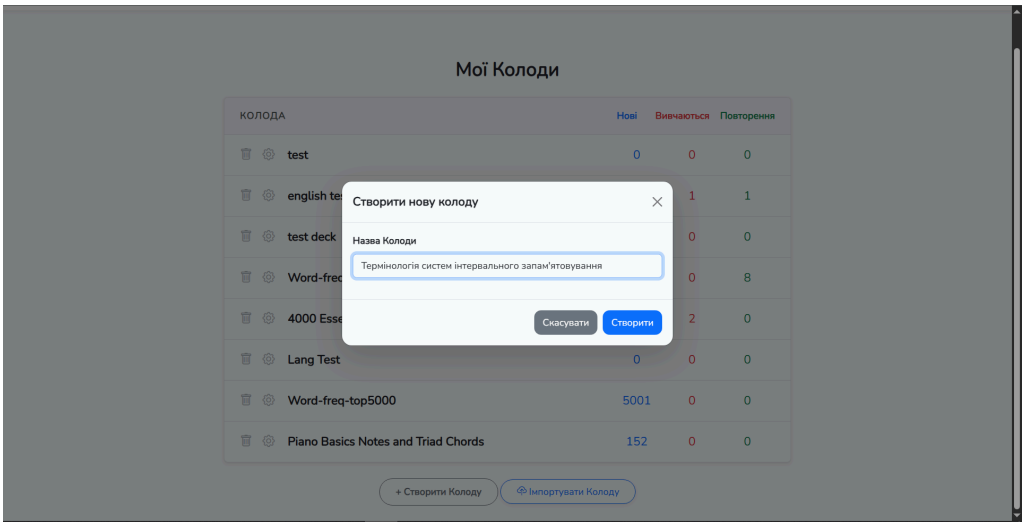


Рисунок 3.6 – Модальне вікно створення колоди

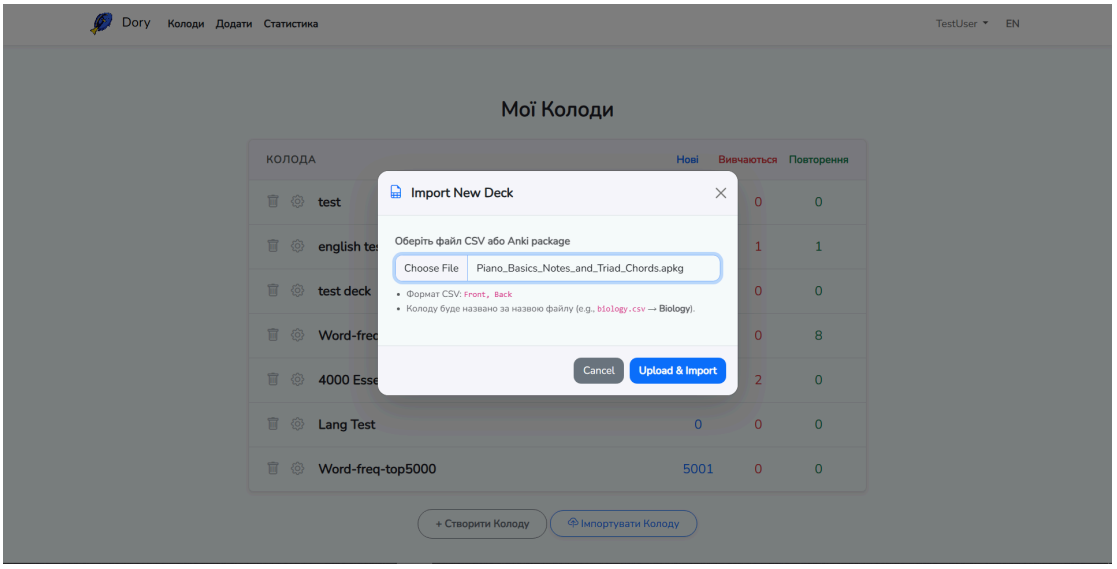


Рисунок 3.7 – Модальне вікно імпорту колоди

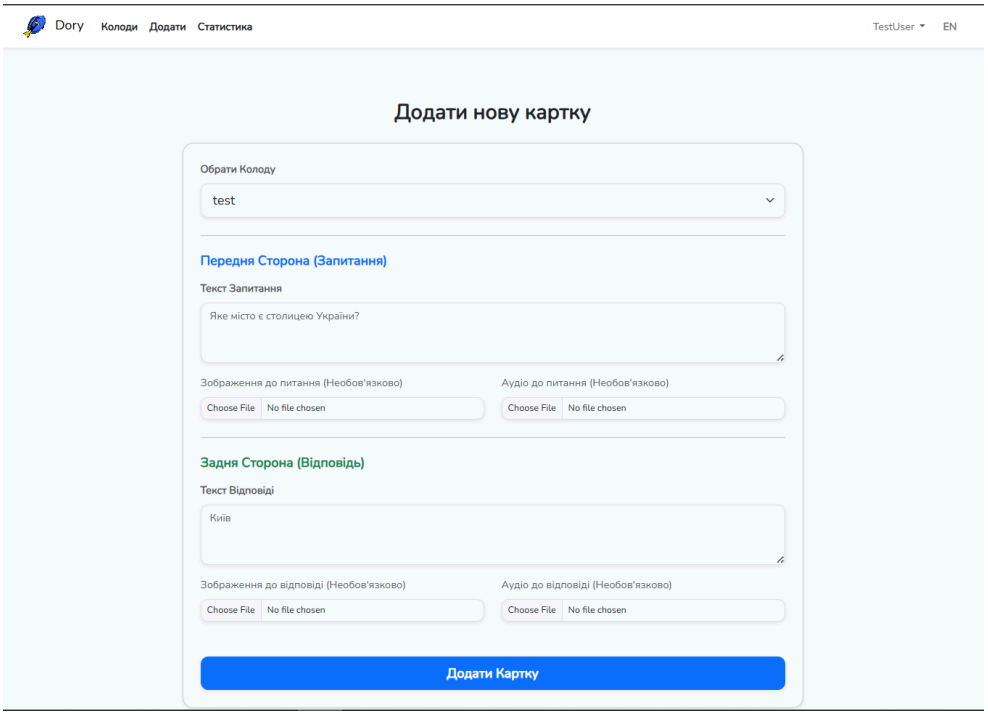


Рисунок 3.8 – Вікно створення картки

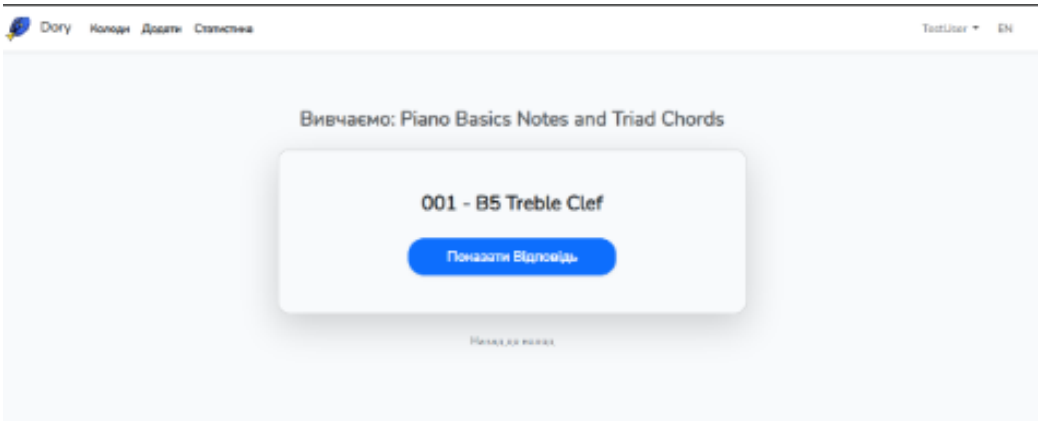


Рисунок 3.9 – Перегляд передньої сторони картки

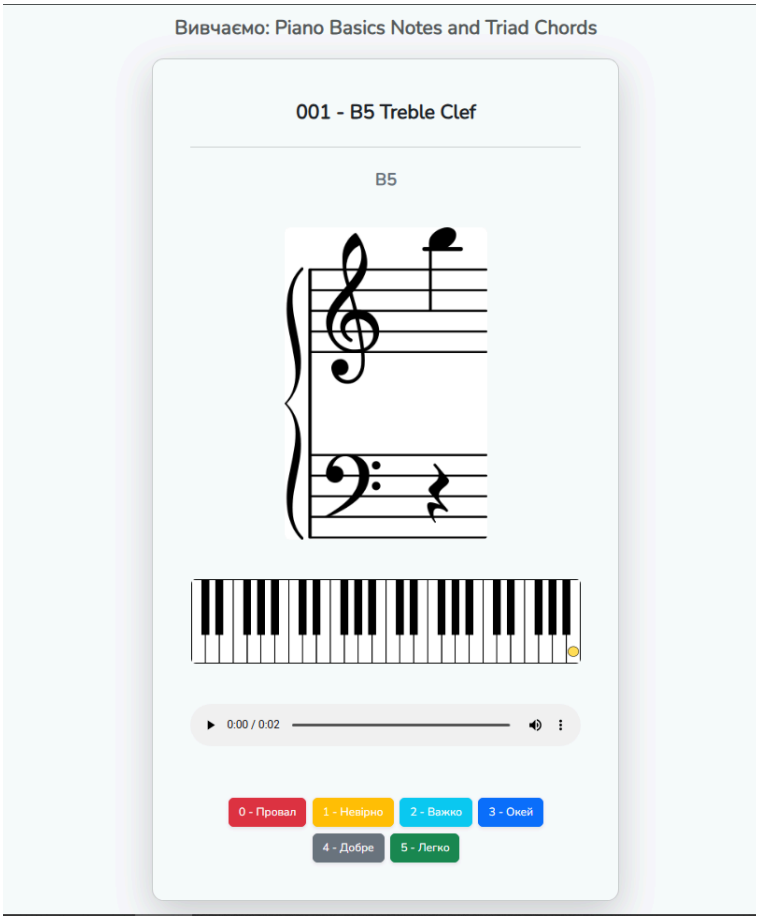


Рисунок 3.10 – Перегляд задньої сторони картки

Рисунок 3.11 – Перегляд статистики по обраній колоді

Спроектований інтерфейс поєднує мінімалістичний дизайн із високою інтерактивністю компонентів Livewire, що дозволяє мінімізувати когнітивне навантаження на користувача під час навчальних сесій. Завдяки адаптивній верстці Tailwind CSS забезпечується комфортна робота з системою.

3.3 Тестування розробленої програми

Результати функціонального тестування описано нижче.

Найбільш важливим вузлом системи є метод processReview у моделі Card (код наведено в додатку Б. У таблиці 3.1 наведено результати перевірки основних сценаріїв роботи алгоритму.

Таблиця 3.1 – Результати функціонального тестування алгоритму SM-2

№	Сценарій	Вхідні дані $(q,\ n,\ I,\ EF)$	Очікуваний результат	Фактичний результат
1	Успіх: перша відповідь	$3,\ 0,\ 0,\ 2.5$	$n = 1, I = 1,$ $EF = 2.5$	Пройдено
2	Успіх: друга відповідь	$5,\ 1,\ 1,\ 2.5$	$n = 2, I = 6,$ $EF = 2.6$	Пройдено
3	Помилка: забування	$0,\ 5,\ 20,\ 2.5$	$n = 0, I = 1,$ $EF = 1.7$	Пройдено
4	Обмеження EF	$2,\ 5,\ 30,\ 1.4$	$n = 0, I = 1,$ $EF = 1.3$	Пройдено

Під час тестування продуктивності було проведено тестування імпортування.

Підсистема імпорту (код наведено в додатку В) використовує черги завдань Laravel для фонової обробки. Для тестування було змодельовано імпорт архівів розміром 10, 100 та 500 карток. Встановлено, що при використанні фонового воркера час відповіді веб-інтерфейсу (TTFB) залишається стабільним – на рівні 120-150 мс, оскільки процес парсингу та збереження даних до бази даних відбувається асинхронно.

Для підтвердження математичної коректності імплементації алгоритму SM-2 було розроблено набір модульних тестів (Unit Tests) з використанням інструментарію PHPUnit (додаток Д наводить Unit-тест для алгоритму SM-2).

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було:

1. Проаналізовано предметну область з точки зору психології навчання та когнітивних навантажень, що дозволило визначити ключові фактори ефективності запам’ятовування інформації.

2. Проведено порівняльний аналіз існуючих систем інтервального повторення, результатом якого стало виявлення технологічних обмежень у доступних рішеннях та обґрунтування необхідності розробки веб-застосунку.

3. Сформульовано технічне завдання на проектування системи, що поєднує математичну точність алгоритму SM-2 із сучасною архітектурою веб-застосунку для забезпечення персоналізованого навчання.

4. Розроблено багаторівневу архітектуру веб-системи, яка забезпечує модульність компонентів та ефективну взаємодію між сервером бази даних і клієнтським інтерфейсом.

5. Обґрунтовано вибір і математично інтерпретовано алгоритм SM-2 як основу для керування станами пам’яті

6. Розроблено інформаційне забезпечення та модель даних у 3-й нормальній формі, що гарантує цілісність, мінімізацію надмірності та високу швидкість обробки інформації про прогрес користувача.

7. Здійснено імплементацію програмного забезпечення з використанням TALL-стеку технологій, що дозволило забезпечити високу швидкість розробки та ефективну інтеграцію серверної бізнес-логіки.

8. Спроектовано ергономічний інтерфейс користувача з використанням підходу “функціональність-найголовніше”, спрямований на мінімізацію когнітивного навантаження та адаптивність до різних пристроїв.

9. Проведено тестування розробленого застосунку, яке підтвердило коректність його роботи та відповідність програмного продукту функціональним вимогам до надійності та відмовостійкості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wozniak P. Did Ebbinghaus invent spaced repetition? SuperMemo, 2017. URL: https://www.supermemo.com/en/blog/did-ebbinghaus-invent-spaced-repetition
2. Tabibian B. et al. Enhancing Human Learning via Spaced Repetition Optimization. Proceedings of the ACM Web Conference, 2021. URL: https://dl.acm.org/doi/10.1145/3442381.3450036
3. Wozniak P. A. Optimization of learning. Master’s thesis. Poznań : University of Economics in Poznań, 1990. 102 p. URL: https://super-memory.com/english/ol.htm
4. Anki. URL: https://apps.ankiweb.net/
5. SuperMemo. URL: https://super-memo.com/
6. SuperMemo.com. URL: https://www.supermemo.com/
7. Quizlet. URL: https://quizlet.com/
8. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 1. Організація баз даних та знань: підручник. Львів: Магнолія 2006, 2024. 675 с.
9. Stauffer M. Laravel: Up and Running. 3rd ed. O'Reilly Media, 2024
10. Windmill T. Building Dynamic Web Interfaces with Livewire 3. Packt Publishing, 2023.
11. Alpine.js. Official documentation and reference. URL: https://alpinejs.dev/
12. Advanced UI Patterns with Utility-First CSS. Tailwind CSS Mastery, 2025.
13. PHP: Hypertext Preprocessor. Official Documentation. The PHP Group. URL: https://www.php.net/manual/en/
14. MySQL. Official reference manual. URL: https://dev.mysql.com/doc/
15. Laravel Documentation. Laravel – The PHP Framework For Web Artisans. URL: https://laravel.com/docs
16. Settles B., Meeder B. A Trainable Spaced Repetition Model for Language Learning. Journal of Educational Data Mining, 2023

17. Кухар Л. О. Архітектура та етапи розробки веб-орієнтованих систем дистанційного навчання. Інформаційні технології і засоби навчання. 2018. Т. 65, № 3. С. 132–145.

18. Cross-Origin Resource Sharing (CORS). MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

19. Client-server overview. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-server_overview

20. Литвин В.В., Пасічник В.В., Шаховська Н.Б. Проектування інформаційних систем: навчальний посібник. Львів: Вид-во «Магнолія 2006», 2021. 380 с.

21. Ушенко Ю.О., Ковальчук М.Л., Гавриляк М.С., Негрич А.Л. Методологія інформаційних систем та баз даних: теоретичний і практичний підходи: навч. посібник. Чернівці: ЧНУ ім. Ю. Федьковича, 2021. 240 с.

22. Ткаченко К. Гармонія в UX/UI дизайні. Київ: Book Chef, 2025. 208 с.

23. Діаграма послідовності (sequence diagrams). URL: https://www.maxzosim.com/sequence-diagrams/.

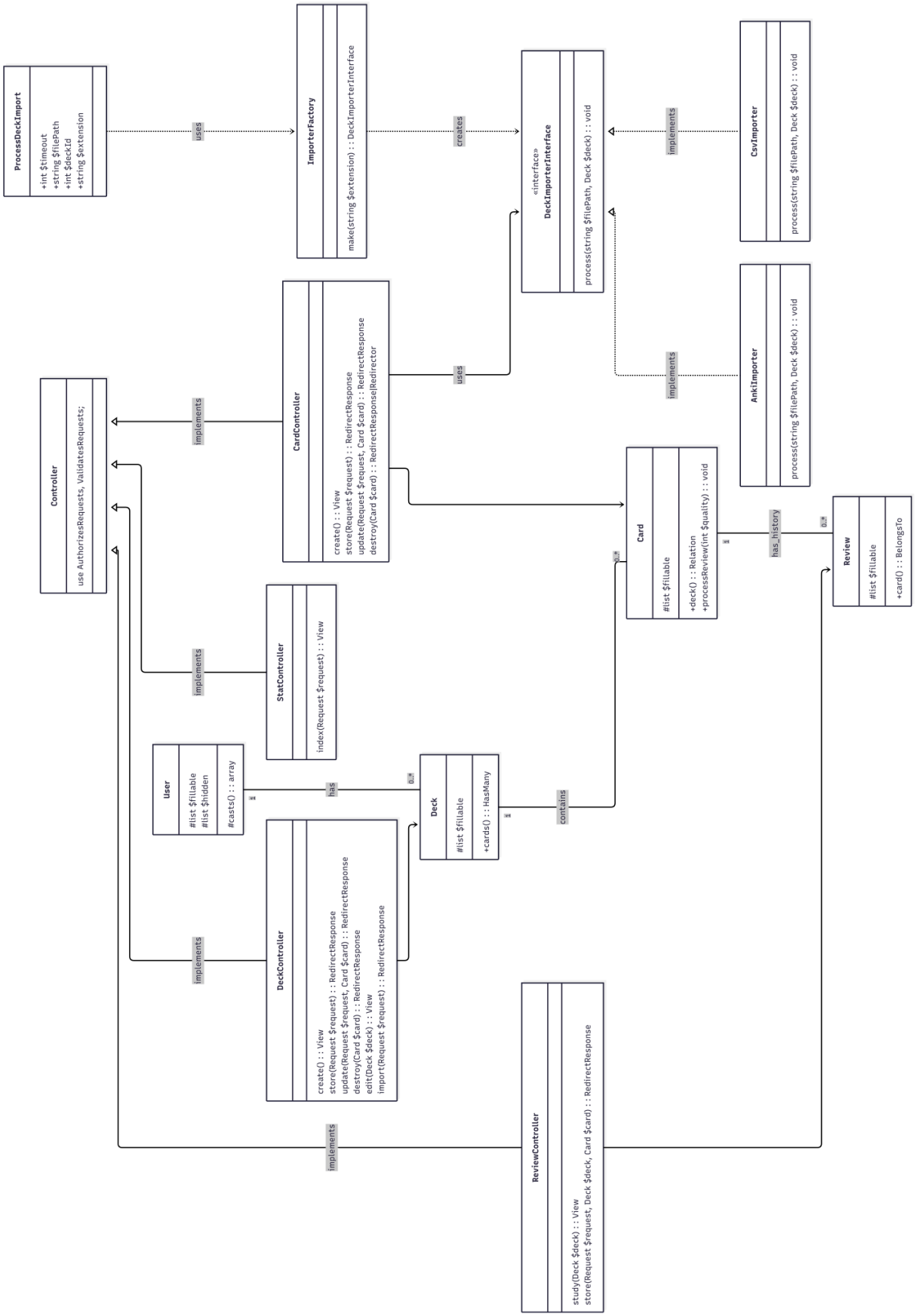
24. UML Tutorial. Tutorialspoint. URL: https://www.tutorialspoint.com/uml/index.htm

25. Хархаліс В.Ю., Турченко І.В. Веб-застосунок для інтервального запам'ятовування. CSPM 2026: Інтелектуальні обчислювальні системи та управління проєктами: збірник тез доповідей міжнародної науково-практичної конференції студентів і молодих вчених (Тернопіль, 21-22 травня 2026 р). Тернопіль: ЗУНУ, 2026. С. 165-167.

26. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Деталізована структура класів розроблюваного веб-застосунку



Додаток Б

Програмний код логіки обчислення часових інтервалів та коефіцієнтів легкості

```
public function processReview(int $quality): void
{
   // If the user scores less than 3, they forgot the
card.
   // Repetitions drop back to 0.
   if ($quality >= 3) {
       if ($this->repetition == 0) {
           $this->interval = 1;
       } elseif ($this->repetition == 1) {
           $this->interval = 6;
       } else {
           $this->interval = (int) round($this->interval *
$this->ease_factor);
       }
       $this->repetition++;
   } else {
       $this->repetition = 0;
       $this->interval = 1;
   }

   // Calculate the new Ease Factor
   $this->ease_factor = $this->ease_factor + (0.1 - (5 -
$quality) * (0.08 + (5 - $quality) * 0.02));

   // The EF should never drop below 1.3
   if ($this->ease_factor < 1.3) {
       $this->ease_factor = 1.3;
   }

   // Schedule the next review date
   $this->next_review_at =
now()->addDays($this->interval);
   $this->save();
}
```

Додаток В

Програмний код підсистеми імпорту

```
public function process(string $filePath, Deck $deck):
void
{
   $tempDir = storage_path('app/temp/' . uniqid('anki_'));
   File::makeDirectory($tempDir, 0755, true, true);

   $fullPath = storage_path('app/private/' . $filePath);

   try {
       $zip = new \ZipArchive;
       if ($zip->open($fullPath) === true) {
           $zip->extractTo($tempDir);
           $zip->close();
       } else {
           throw new Exception('Не вдалося відкрити .apkg
файл.');
       }

       $mediaDir = storage_path('app/public/anki_media');
       File::makeDirectory($mediaDir, 0755, true, true);

       $mediaJsonPath = $tempDir . '/media';
       if (file_exists($mediaJsonPath)) {
               $mediaMap =
json_decode(file_get_contents($mediaJsonPath), true);

               if (is_array($mediaMap)) {
                   foreach ($mediaMap as $fileNumber =>
$realName) {
                       @rename($tempDir . '/' .
$fileNumber, $mediaDir . '/' . $realName);
                   }
               }
           }

       $sqlitePath21 = $tempDir . '/collection.anki21';
       $sqlitePath20 = $tempDir . '/collection.anki2';

       if (file_exists($sqlitePath21)) {
           $sqlitePath = $sqlitePath21;
       } elseif (file_exists($sqlitePath20)) {
           $sqlitePath = $sqlitePath20;
```

```
        } else {
            throw new Exception('Базу даних Anki не
знайдено.');
        }

        config(
            [
                'database.connections.anki' => [
                'driver'                    => 'sqlite',
                'database'                  => $sqlitePath,
                ]
            ]
        );

        $notes =
DB::connection('anki')->table('notes')->cursor();
        $cardsBatch = [];
        $batchSize = 500;

        foreach ($notes as $note) {
            $fields = explode(chr(31), $note->flds);
            $processedFields = [];

            foreach ($fields as $field) {
                $clean = trim($field);

                // Замінюємо Anki аудіо [sound:file.mp3] на
стандартний HTML плеєр
                $clean = preg_replace('/\[sound:(.*?)\]/i',
'<audio controls src="/storage/anki_media/$1" class="w-100
my-2"></audio>', $clean);

                $clean =
preg_replace_callback('/<img[^>]+src=["\']?([^"\'\s>]+)["\
']?[^>]*>/i', function($matches) {
                    $filename = $matches[1];
                    return '<img
src="/storage/anki_media/'.$filename.'" class="img-fluid
rounded my-2">';
                }, $clean);

                if (!empty($clean)) {
                    $processedFields[] = $clean;
                }
            }
```

```
            if (count($processedFields) === 0) continue;

            if (strlen(strip_tags($processedFields[0])) <=
5 && isset($processedFields[1])) {
                $front = $processedFields[0] . ' | ' .
$processedFields[1];
                $backFields = array_slice($processedFields,
2);
            } else {
                $front = $processedFields[0];
                $backFields = array_slice($processedFields,
1);
            }

            $back = implode("<br><br>", $backFields); //
Зберігаємо HTML-форматування Anki

            $cardsBatch[] = [
                'deck_id'        => $deck->id,
                'question'       => $front,
                'answer'         => $back,
                'repetition'     => 0,
                'ease_factor'    => 2.5,
                'interval'       => 0,
                'next_review_at' => now(),
                'created_at'     => now(),
                'updated_at'     => now(),
            ];

            if (count($cardsBatch) >= $batchSize) {
                Card::insert($cardsBatch);
                $cardsBatch = [];
            }
        }

        if (!empty($cardsBatch)) {
            Card::insert($cardsBatch);
        }

    } finally {
        DB::purge('anki');
        File::deleteDirectory($tempDir);
        Storage::disk('local')->delete($filePath);
    }
```

```
}
```

Додаток Г

Програмний код компоненту інтерфейсу для оцінки картки

```
@props(['card', 'deck'])

{{-- Головний контейнер з x-cloak --}}
<div x-data="{ showAnswer: false }" x-cloak class="card
shadow-lg mx-auto mt-4" style="max-width: 600px; border-radius:
15px; overflow: hidden;">
   <div class="card-body p-5 text-center">

       <div class="mb-4">
           {{-- ПИТАННЯ --}}
           <div class="fs-3 fw-bold text-dark mb-3">
               {!! $card->question !!}
           </div>

           @if($card->question_image)
               <div class="mb-3" style="min-height: 250px;">
                   <img src="{{ asset('storage/' .
$card->question_image) }}"
                        class="img-fluid rounded shadow-sm"
loading="lazy"
                        style="max-height: 250px; object-fit:
contain;">
               </div>
           @endif
       </div>

       {{-- Кнопка --}}
       <button
           x-show="!showAnswer"
           @click="showAnswer = true"
           class="btn btn-primary btn-lg px-5 shadow-sm fw-bold"
           style="border-radius: 20px;"
       >
           {{__('Show Answer')}}
       </button>

       <div x-show="showAnswer"
            x-cloak
            style="display: none; min-height: 200px;"
            x-transition>

           <hr class="my-4 text-muted opacity-25">

           <div class="mb-5">
```

```
                <div class="fs-4 text-secondary mb-3
fw-semibold">
                    {!! $card->answer !!}
                </div>

                @if($card->answer_image)
                    <div class="mb-3">
                        <img src="{{ asset('storage/' .
$card->answer_image) }}"
                             class="img-fluid rounded shadow-sm"
                             style="max-height: 250px;
object-fit: contain;">
                    </div>
                @endif
            </div>

            {{-- ФОРМА --}}
            <form action="{{ route('decks.reviews.store', [$deck,
$card]) }}" method="POST" class="d-flex justify-content-center
flex-wrap gap-2">
                @csrf
                <button type="submit" name="quality" value="0"
class="btn btn-dark fw-semibold px-3">0 -
{{__('Blankout')}}</button>
                <button type="submit" name="quality" value="1"
class="btn btn-danger fw-semibold px-3">1 -
{{__('Wrong')}}</button>
                <button type="submit" name="quality" value="2"
class="btn btn-warning fw-semibold text-dark px-3">2 -
{{__('Hard')}}</button>
                <button type="submit" name="quality" value="3"
class="btn btn-info fw-semibold text-white px-3">3 -
{{__('OK')}}</button>
                <button type="submit" name="quality" value="4"
class="btn btn-primary fw-semibold px-3">4 -
{{__('Good')}}</button>
                <button type="submit" name="quality" value="5"
class="btn btn-success fw-semibold px-3">5 -
{{__('Easy')}}</button>
            </form>
          </div>
     </div>
</div>
```

Додаток Д
Unit-тест для алгоритму SM-2

```
/** @test */
public function test_sm2_algorithm_calculation_logic()
{
   $deck = Deck::factory()->create(['id' => 1]);
   // 1. Створюємо картку з початковими параметрами
   $card = Card::factory()->create([
       'repetition' => 0,
       'ease_factor' => 2.5,
       'interval' => 0,
   ]);

   // 2. Симулюємо успішну відповідь (якість 3)
   $card->processReview(3);

   // 3. Перевіряємо результати згідно з таблицею
   $this->assertEquals(1, $card->repetition);
   $this->assertEquals(1, $card->interval);
   $this->assertEquals(2.5, $card->ease_factor);
}

/** @test */
public function test_sm2_algorithm_reset_on_failure()
{
   $card = Card::factory()->create([
       'repetition' => 5,
       'ease_factor' => 2.5,
       'interval' => 30,
   ]);

   // Симулюємо "забування" (якість 0)
   $card->processReview(0);

   // Перевіряємо скидання параметрів
   $this->assertEquals(0, $card->repetition);
   $this->assertEquals(1, $card->interval);
}
```

Додаток Е

Копія опублікованих результатів