Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія для запам'ятовування інформації»

Виконала: студентка 2-го курсу, групи 1КН-23м спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Домбровська В.Є.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Озеранський В.С.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. САІТ
Варчук І.В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
11.10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Домбровській Валерії Євгенівній
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи: «Інформаційна технологія для запам'ятовування інформації»
   керівник роботи:к.т.н., доцент каф КН Озеранський В.С.
   затверджені наказом ВНТУ від «17» 09 2024 року № 310
2. Строк подання студентом роботи 11 11 2024 року
3. Вхідні дані: Мова програмування – об'єктно-орієнтована, кількість паттернів для запам'ятовування – не менше 10шт, кількість тестів – не менше 50шт; відтримка виконання когнітивних завдань.
4. Зміст текстової частини: Вступ, аналіз предметної області запам'ятовування інформації, розробка інформаційної технології запам'ятовування інформації, програмна реалізація інформаційної технології запам'ятовування інформації, економічна частина, висновки, перелік використаних джерел, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): Алгоритм роботи програми «Інформаційна технологія для запам'ятовування інформації» UML діаграма класів програми, робочі вікна програми, результати тестування програми, програми-аналоги.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Виконання прийняв
1-3	Озеранський В.С., к.т.н., доц. каф. КН		
4	Лесько О.Й., к.е.н., проф. каф. ЕПВМ		

7. Дата видачі завдання 02 09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ етапу	Назва етапу	Строк виконання етапів роботи	При- мітка
1	Аналі предметної області запам'ятовування інформації	02.09.24	15.09.24
2	Проєктування інформаційної технології запам'ятовування інформації	17.09.24	29.09.24
3	Програмна реалізація технології запам'ятовування інформації	30.09.24	15.10.24
4	Тестування та аналіз результатів роботи розробленої програми	16.10.24	17.10.24
5	Розробка інструкції користувача. Підготовка економічної частини	18.10.24	28.10.24
6	Оформлення матеріалів до захисту МКР	05.11.24	11.11.24

Студент _____ Домбровська В.Є.
(підпис)

Керівник роботи _____ Озеранський В.С
(підпис)

АНОТАЦІЯ

УДК 621.374.415

Домбровська В.Є. «Інформаційна технологія для запам’ятовування інформації». Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма – Системи штучного інтелекту. Вінниця: ВНТУ, 2024.119 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 19; табл. 8.

Ця магістерська робота присвячена розробці інформаційної технології для поліпшення процесу запам'ятовування інформації. Вивчено та проаналізовано сучасні методи та інструменти, що сприяють ефективному зберіганню знань, а також проаналізовано когнітивні моделі пам'яті для вибору оптимального підходу. На основі проведених досліджень було обрано методику інтервального повторення, яка найкраще підходить для підтримки довготривалого запам'ятовування. Створено програмне забезпечення, яке допомагає користувачам запам'ятовувати великі обсяги інформації, використовуючи персоналізовані інтервали повторення. Програму написано на мові програмування Python з використанням бібліотек для обробки даних і побудови адаптивних алгоритмів повторення.

Графічна частина роботи включає візуалізації алгоритмів повторення та інтерфейсу користувача.

В економічному розділі проведено розрахунок витрат на розробку та обслуговування системи, оцінено рентабельність і переваги для кінцевих користувачів, включаючи підвищення продуктивності та зменшення часу на вивчення матеріалів.

Ключові слова: запам'ятовування, інтервальне повторення, когнітивні моделі, інформаційна технологія, алгоритми повторення.

ABSTRACT

Dombrovska V.E. "Information technology for memorizing information". Master's qualification work on specialty 122 - computer science, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2024. 130 p.

In Ukrainian speech Bibliography: 21 titles; Fig.: 19; table 8.

This master's thesis is devoted to the development of information technology to improve the process of memorizing information. Modern methods and tools contributing to effective knowledge storage were studied and analyzed, as well as cognitive models of memory were analyzed to choose the optimal approach. Based on the research, the technique of interval repetition was chosen, which is best suited for maintaining long-term memorization. Created software that helps users memorize large amounts of information using personalized repetition intervals. The program is written in the Python programming language using libraries for data processing and building adaptive repetition algorithms.

The graphic part of the work includes visualizations of the repetition algorithms and the user interface.

The economic section calculates the costs of developing and maintaining the system, evaluates the profitability and benefits for end users, including increased productivity and reduced time to study materials.

Keywords: memorization, interval repetition, cognitive models, information technology, repetition algorithms.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світ вимагає від людини швидкого і ефективного оброблення великих обсягів інформації. У цьому контексті розвиток та вдосконалення когнітивних здібностей, зокрема пам'яті, стає все більш важливим завданням. Інформаційні технології можуть значно сприяти тренуванню пам'яті, надаючи інноваційні методи та інструменти для покращення її функціонування.

Інформаційні технології для запам’ятовування інформації дозволяють створювати індивідуалізовані програми навчання, які адаптуються до особливостей кожного користувача. Це допомагає не лише зберегти час і ресурси, але й підвищити ефективність навчання, уникнути помилок, пов'язаних із людським фактором, та забезпечити стійкі результати. Впровадження таких технологій знаходить застосування в освіті, професійній діяльності, медицині, а також у повсякденному житті.

Завдяки інформаційним технологіям запам’ятовування інформації стало доступнішим та ефективнішим. Спеціалізовані програми можуть допомогти школярам, студентам, фахівцям різних галузей та літнім людям покращити свою здатність запам'ятовувати та використовувати інформацію. Це не лише підвищує продуктивність і успішність, але й сприяє загальному поліпшенню якості життя.

Використання інформаційних технологій для запам’ятовування інформації дозволяє досягти цих цілей більш ефективно та результативно. Звідси можна зробити висновок, що тема створення інформаційної технології запам’ятовування інформації є надзвичайно актуальною в сучасних умовах.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напряму наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання дослідження.

Метою магістерського дослідження є розширення функціональних можливостей програмного забезпечення для запам’ятовування інформації.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- здійснити аналіз сучасних технологій запам’ятовування інформації;
- здійснити порівняльний аналіз програм аналогів запам’ятовування інформації;
- здійснити моделювання інформаційної технології запам’ятовування інформації;
- розробити алгоритм функціонування інформаційної технології запам’ятовування інформації;
- здійснити програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного засобу та проаналізувати отримані результати;
- здіснити економічні розрахунки доцільності створення нової інформаційної технології.

Об’єкт дослідження – процес запам’ятовування інформації користувачем.

Предметом дослідження є програмні засоби для запам’ятовування інформації користувачем.

Методи дослідження. У роботі використано такі методи наукових досліджень: методи та моделі подання знань; методи тестування програмних засобів, зокрема функціональне тестування та тестування методом «чорного ящика» для перевірки ефективності технології; методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

Набула подальшого розвитку інформаційна технологія запам’ятовування інформації, яка відрізняється від існуючих використанням удосконаленою інформаційною моделлю запам’ятовування інформації, що забезпечує

розширення функціональних можливостей програмного забезпечення для запам’ятовування інформації.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення для запам’ятовування інформації.

Запропонована інформаційна технологія сприяє підвищенню ефективності процесу запам’ятовування інформації:

- розроблено алгоритм функціонування інформаційної технології запам’ятовування інформації;
- розроблено програмне забезпечення для тренування запам’ятовування інформації.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології запам’ятовування інформації. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, що наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Основні результати досліджень апробовано на міжнародній науково-практичній конференції "Молодь в науці: дослідження, проблеми, перспективи" (МН-2025). [1]

Публікації. За результатами досліджень опубліковано тези доповіді на науково-технічній конференції. [1]

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАПАМ'ЯТОВУВАННЯ ІНФОРМАЦІЇ

1.1 Постановка задачі технології запам'ятовування інформації

Постановка задачі технології запам'ятовування інформації є важливим етапом у розробці систем, що сприяють покращенню когнітивних функцій пам'яті користувачів. Вона включає визначення конкретних цілей і завдань, які необхідно досягти шляхом впровадження технологій для ефективного запам'ятовування. Основною метою є розробка програмних засобів або методів, які сприятимуть покращенню пам'яті, зроблять її більш надійною та швидкою. Для досягнення цієї мети в постановці задачі важливо врахувати низку суттєвих аспектів:

- Першим етапом є аналіз існуючих рішень, які вже застосовуються в сфері технологій пам'яті, щоб зрозуміти їх переваги та недоліки. Це дозволить виявити прогалини на ринку і вказати напрямки для покращення майбутніх рішень. ;
- Наступним кроком є порівняльний аналіз аналогічних технологій, що дозволить оцінити різноманітні підходи до вирішення задачі та вибрати найбільш ефективний;
- Далі необхідно чітко визначити основні завдання, які мають бути вирішені в процесі розробки. Це включає створення алгоритмів, розробку інтерфейсів, інтеграцію різних технологій та оптимізацію процесів;
- Важливим етапом є проектування, яке має бути орієнтованим на потреби користувачів, зокрема на інтуїтивно зрозумілий інтерфейс і високу ефективність;
- Крім того, потрібно вибрати відповідні програмні засоби, які дозволять реалізувати всі поставлені завдання. Це може включати вибір мови програмування, фреймворків, баз даних і так далі;

- Розробка фронтенд-частини для системи також є важливою частиною процесу, оскільки від цього залежить взаємодія користувача з технологією та її зручність у використанні.;
- Завершальний етап включає тестування розробленої системи, яке дозволить виявити можливі помилки та недоліки, а також перевірити ефективність у реальних умовах використання.

Правильна постановка задачі дозволить розробити ефективні програмні засоби або методи запам'ятовування інформації, які відповідають потребам та очікуванням користувачів.

1.2 Цільова аудиторія та галузі застосування

Інформаційна технологія запам'ятовування інформації має широку цільову аудиторію та галузі застосування. До цільової аудиторії належать учні та студенти різних навчальних закладів, які можуть використовувати технології для покращення запам'ятовування навчального матеріалу. Медичні працівники, такі як лікарі, медсестри та інший медичний персонал, можуть застосовувати ці методики для збереження медичних знань і процедур і не боятися що якісь знання забудуться з часом. ІТ-спеціалісти, розробники програмного забезпечення, інженери та інші фахівці також можуть використовувати тренування запам'ятовування інформації для запам'ятовування технічних деталей і алгоритмів, або підготовки до співбесід.

Також можна застосовувати в профілактичних цілях. Люди старшого віку, зокрема пенсіонери, можуть застосовувати ці технології для підтримки когнітивного здоров'я та профілактики вікових змін у роботі мозку. Особи з когнітивними порушеннями, такими як пацієнти з деменцією, можуть покращити якість життя та підтримувати ментальну активність завдяки цим методикам.

Галузі застосування інформаційної технології запам'ятовування інформації є також досить різноманітними. В освітній сфері ці технології можуть бути використані в навчальних закладах для покращення успішності учнів і студентів, а також для розробки спеціальних навчальних програм. У медицині їх можна інтегрувати в реабілітаційні програми для пацієнтів з когнітивними порушеннями та для підвищення ефективності запам'ятовування медичних знань. У бізнесі та професійному розвитку ці технології можуть застосовуватися для тренінгів, що допомагають співробітникам запам'ятовувати важливу інформацію та підвищувати продуктивність, а також у програмах корпоративного навчання.

Таким чином, інформаційні технології запам'ятовування інформації можуть бути ефективно застосовані у багатьох сферах, підвищуючи когнітивні здібності та ефективність діяльності різних груп людей. Отже ми ще раз підтвердили актуальність теми.

1.3 Особливості методів запам'ятовування інформації

Існують різні методи сприйняття та обробки наданої інформації людиною. Наукові дослідження про різні методи запам'ятовування, такі як візуальне, аудіальне, кінестетичне та дискретне, проводяться в галузі когнітивної психології та нейронауки.

Одним із визначень когнітивних технологій є розуміння їх як способу перетворення когнітивної поведінки людей, організацій та націй через підвищення їх інтелектуального потенціалу або впровадження їх в сучасну Інформаційну система"

В основі когнітивних технологій лежить когнітивна наука, яка вивчає, як люди сприймають світ, як вони мислять, на що звертають увагу, як запам'ятовують інформацію і т.д. таким чином, когнітивні технології є основою неврології, теорії синергетичних ефектів (самоорганізації), комп'ютерних та інформаційних технологій, математичного моделювання людської свідомості, а раніше і основою і принципом самоорганізації.[2]

1.3.1 Візуалізатори

Багато досліджень підтверджують, що:

- 80% інформації, яку людина сприймає, походить через зір;
- 70% сенсорних рецепторів розташовані в очах, а приблизно половина нейронів головного мозку задіяна у обробці візуальної інформації;
- обробка візуальної інформації вимагає на 19% менше когнітивної функції мозку, яка відповідає за аналіз інформації;
- продуктивність людини, яка працює з візуальною інформацією, вище на 17%, і вона краще запам'ятовує деталі візуальної інформації на 4,5%;
- візуальна інформація сприймається в 60 000 разів швидше, ніж текстова; графіки дозволяють читачам швидше знайти мінімальні та максимальні значення.[3]

Отже людина, яка найкраще сприймає і запам'ятовує інформацію візуально, називається "візуалом". Для таких людей зорові відчуття мають визначальне значення у процесі сприймання інформації. Вони здатні швидко помічати й запам'ятовувати такі елементи, як колір, форму та розмір об'єктів, надаючи перевагу візуальній інформації над іншими типами сприйняття. Зазвичай для візуалів більш ефективними є графічні матеріали, схеми, зображення або візуально організовані тексти, що дає змогу їм легше сприймати й запам'ятовувати інформацію.

Більшість творчих людей серед візуалів сприймають зовнішній світ через образи, фантазію та уяву. Їм властива підвищена жестикуляція, оскільки вони

часто відчувають нестачу слів для повного опису своїх думок. Це через те, що візуали думають картинкою, яка має набагато більше фарб, ніж слів. Часто вони здатні відтворити образ, подивившись на нього, та описати його за допомогою своєї уяви. А з появою електронних носіїв інформації в навчанні виникла можливість використання їх з різних причин, таких як скорочення часу на отримання друкованого матеріалу, відсутність територіальних обмежень та ергономічність електронних пристроїв. Крім того, вони дають змогу додавати до навчального матеріалу не лише зображення, як у друкованих носіях, а й відеоінфографіку та інтерактивні елементи, що значно розширює можливості навчання таких людей.[4]

1.3.2 Аудіали

При спілкуванні аудіали зазвичай спрямовують свій погляд по середній лінії і легко відволікаються на звуки. Вони запам'ятовують інформацію, слухаючи та обговорюючи, а також зазвичай пам'ятають імена, але можуть забувати обличчя. Аудіали віддають перевагу читанню діалогів і п'єс, користуються усними інструкціями, наданими як вчителем, так і самими собою. Вони також більш схильні до того, щоб чути або читати вголос інформацію, ніж читати її мовчки.

Для цього типу людей краще всього підходить слухання аудіо-записів, таких як лекції, аудіокниги або аудіоуроки. Все що потрібно це знайти матеріали на цікаву тему. Також для аудіалів гарним тренуванням буде проговорювати матеріали в голос. Повторення або розповідь про вивчений матеріал вголос може сприяти кращому запам'ятовуванню та розумінню. Можна записувати свої власні аудіо-записи, які потім можна прослуховувати знову.

Отже, аудіали краще вміють писати переказ, з великим задоволенням слухають аудіозаписи, а не читають надруковані тексти в книгах. Вони з радістю розігрують діалоги, беруть активну участь в дискусіях і переказують текст у

формі інтерв'ю. Вони насолоджуються сприйняттям мовного матеріалу з аудіо- та відеозаписів.[5]

1.3.3 Кінестетики

Кінестетики — це люди, які віддають перевагу навчанню через фізичну активність і взаємодію з оточенням. Вони найкраще засвоюють інформацію, коли можуть її відчути на власному досвіді, рухатися або маніпулювати об'єктами. Такий підхід до навчання називають кінестетичним або тактильним.[6]

Кінестетики під час розмови часто дивляться вниз і їм складно зосередитися, вони легко відволікаються. Запам'ятовують інформацію через рух, дотик, запах, і запам'ятовують загальне враження. Під час виконання домашнього завдання кінестетик довго шукає необхідні підручники, знаходить потрібні сторінки і відзначає завдання прямо в підручниках. Зауваження сприймають краще, якщо торкнутися їхнього плеча. Учні-кінестетики мають труднощі з сприйняттям інформації на слух і візуально, їм складно працювати з підручниками та сидіти на місці, свої емоції вони виражають мімікою та жестами.

Як краще працювати з кінестетиками:

- Використовуйте якомога більше практичних занять;
- Дозволяйте писати на дошці під час вирішення завдань;
- Використовуйте флеш-картки (flash cards) для навчання;
- При спілкуванні використовуйте жести, дотики, повільну мову.

Ці методи допомагають кінестетикам не лише ефективніше засвоювати нову інформацію, а й залишатися зацікавленими та залученими до процесу навчання. Завдяки фізичній активності та інтерактивним завданням, вони можуть краще концентруватися та активно включатися в навчальний процес. [7]

1.4 Аналіз сучасних методик запам’ятовування інформації

1.4.1 Мнемонічні прийоми

Мнемонічні прийоми — це техніки, які допомагають запам'ятовувати інформацію. Вони засновані на використанні асоціацій, образів, рим, акронімів і структур, що полегшують запам'ятовування. Ось деякі популярні мнемонічні прийоми:[8]

1. Асоціації: Створення зв’язків між новою інформацією та вже відомою. Наприклад, асоціювати слово з яскравим образом.
2. Рими і ритми: Використання рим або ритмічних фраз, щоб зробити інформацію більш запам'ятовуваною. Наприклад, віршовані рядки.
3. Акроніми: Створення слів з перших літер слів, які потрібно запам'ятати. Наприклад, «NASA» (National Aeronautics and Space Administration).
4. Метод місць (метод loci): Уявлення собі знайомого місця і "розміщення" інформації в різних його точках. Коли потрібно згадати інформацію, уявляєте це місце.
5. Майнд-карти: Візуальні схеми, які допомагають організувати та структуризувати інформацію, використовуючи малюнки, кольори та ключові слова.
6. Візуалізація: Створення яскравих образів у розумі, щоб уявити інформацію. Це може включати уявлення про сценарії або картинки.

1.4.2 Метод повторення

Метод повторення — це одна з найефективніших стратегій запам'ятовування, яка полягає у регулярному повторенні матеріалу для зміцнення пам'яті. Ось кілька ключових аспектів цього методу:

1. Інтервальне повторення: Цей підхід передбачає повторення інформації через поступово зростаючі інтервали часу.

2. Активне повторення: Це не просто читання матеріалу, а активне його згадування. Наприклад, закрийте книгу і спробуйте пригадати основні моменти. Це може включати:

   - Відповіді на запитання.
   - Пояснення матеріалу вголос.
   - Написання конспектів без підглядання.[9]

3. Використання флеш-карток: Флеш-картки є чудовим інструментом для повторення. На одній стороні картки пишеться термін або запитання, а на іншій — відповідь. Регулярне проходження через ці картки допомагає закріпити знання.

1.5 Аналіз програм аналогів для тренування пам’яті

1.5.1 Anki

Anki — це популярний додаток для створення флеш-карт, який дозволяє користувачам організовувати інформацію у зручному форматі карток. Кожна картка містить запитання на одній стороні та відповідь на іншій, що допомагає ефективно запам'ятовувати різноманітні факти, терміни, визначення або інші важливі дані. Anki дозволяє створювати власні колоди карток, додавати зображення та звуки, що робить процес навчання ще більш інтерактивним та

цікавим. Завдяки своїй гнучкості, додаток підходить для вивчення різних предметів, від мов до технічних наук.

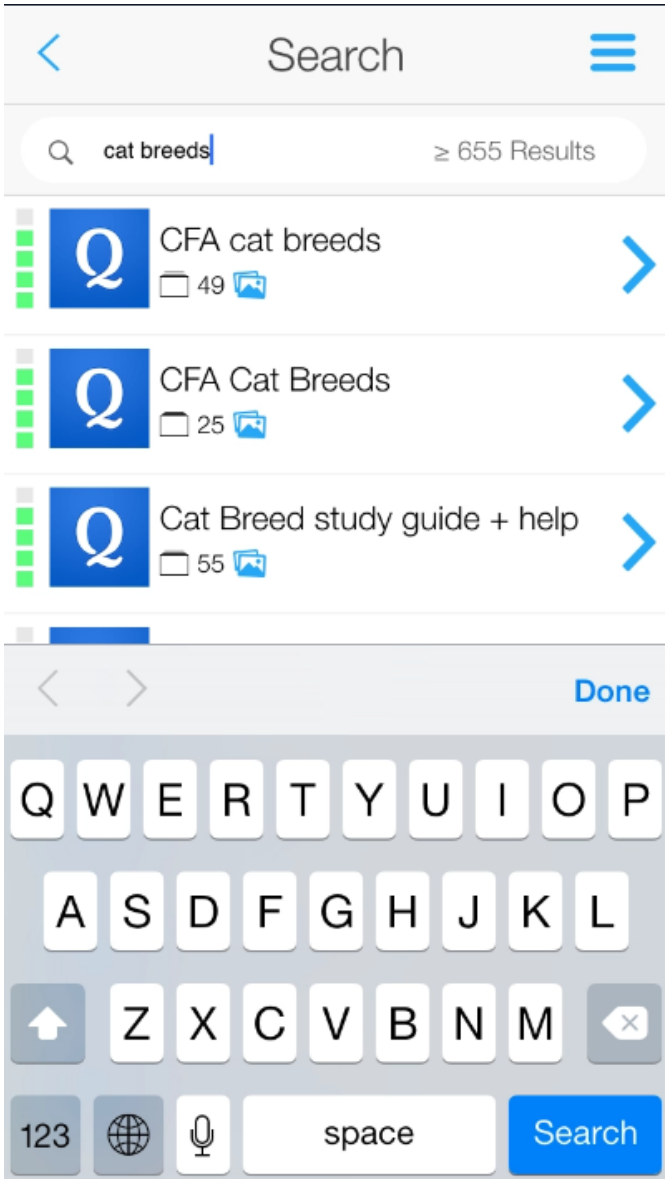

Рисунок 1.1 – Приклад інтерфейсу для додатку Anki

Особливості:

- Гнучкість: користувачі можуть створювати власні картки з текстом, зображеннями, аудіо та відео.
- Крос-платформеність: доступна на різних платформах (Windows, macOS, Android, iOS).

1.5.2 **Quizlet**

Quizlet — це платформа для навчання, яка дозволяє створювати флеш-карти, тести і ігри для запам'ятовування інформації.

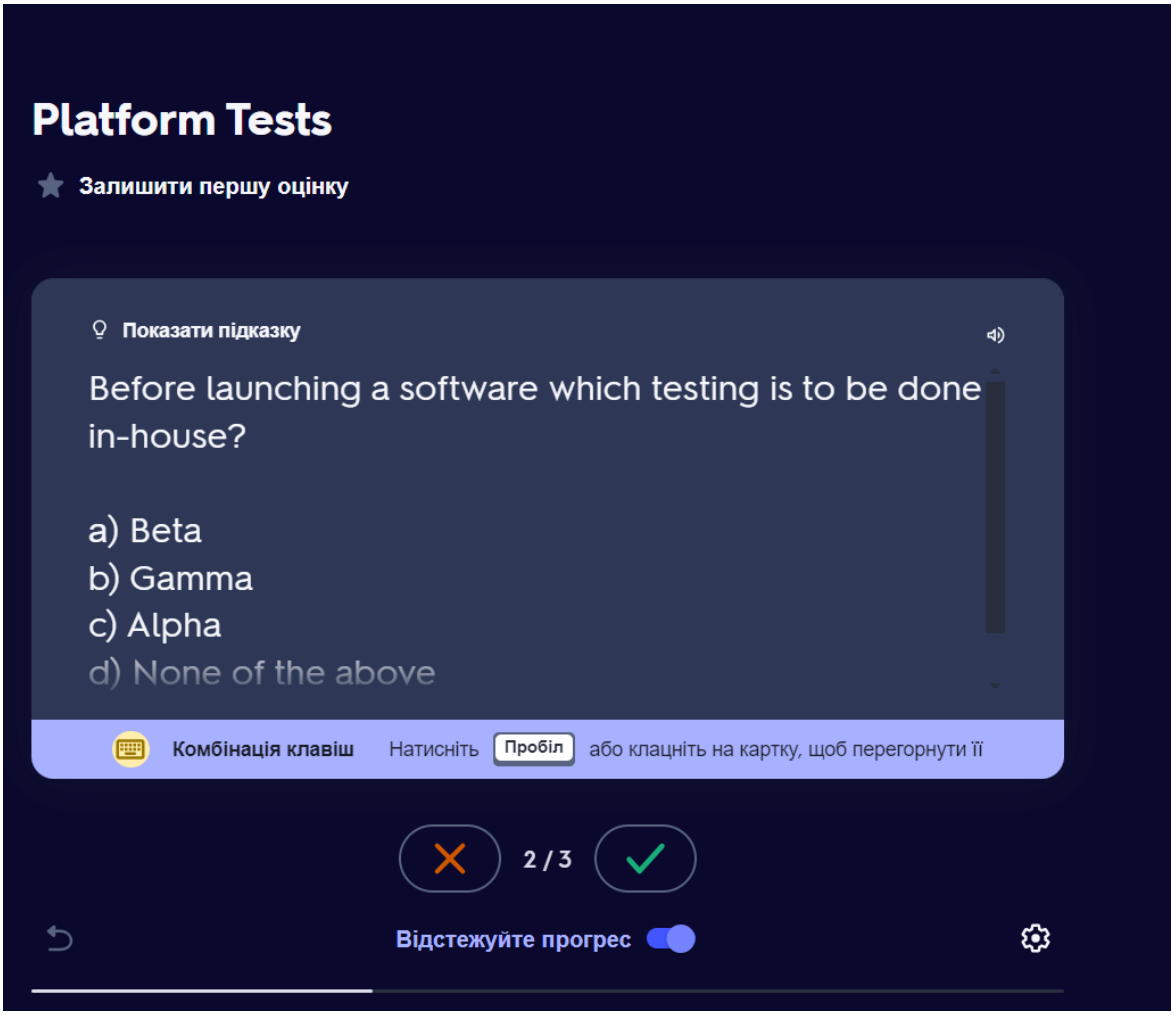

Рисунок 1.2 – Приклад інтерфейсу для додатку Quizlet

Особливості:

- Гнучкість: користувачі можуть створювати власні картки з текстом, зображеннями, аудіо та відео.
- Крос-платформеність: доступна на різних платформах (Windows, macOS, Android, iOS).

1.5.3 Memrise

Memrise — це освітня платформа, яка спеціалізується на вивченні мов та запам'ятовуванні інформації, використовуючи ігрові елементи. Платформа пропонує інтерактивні курси, що поєднують відео, аудіо та текстові матеріали для ефективного освоєння нових мов або знань. Memrise стимулює процес навчання через різноманітні вправи та завдання, що дають можливість користувачам отримувати бали та досягати нових рівнів. Такий підхід робить навчання більш захопливим та мотивуючим.

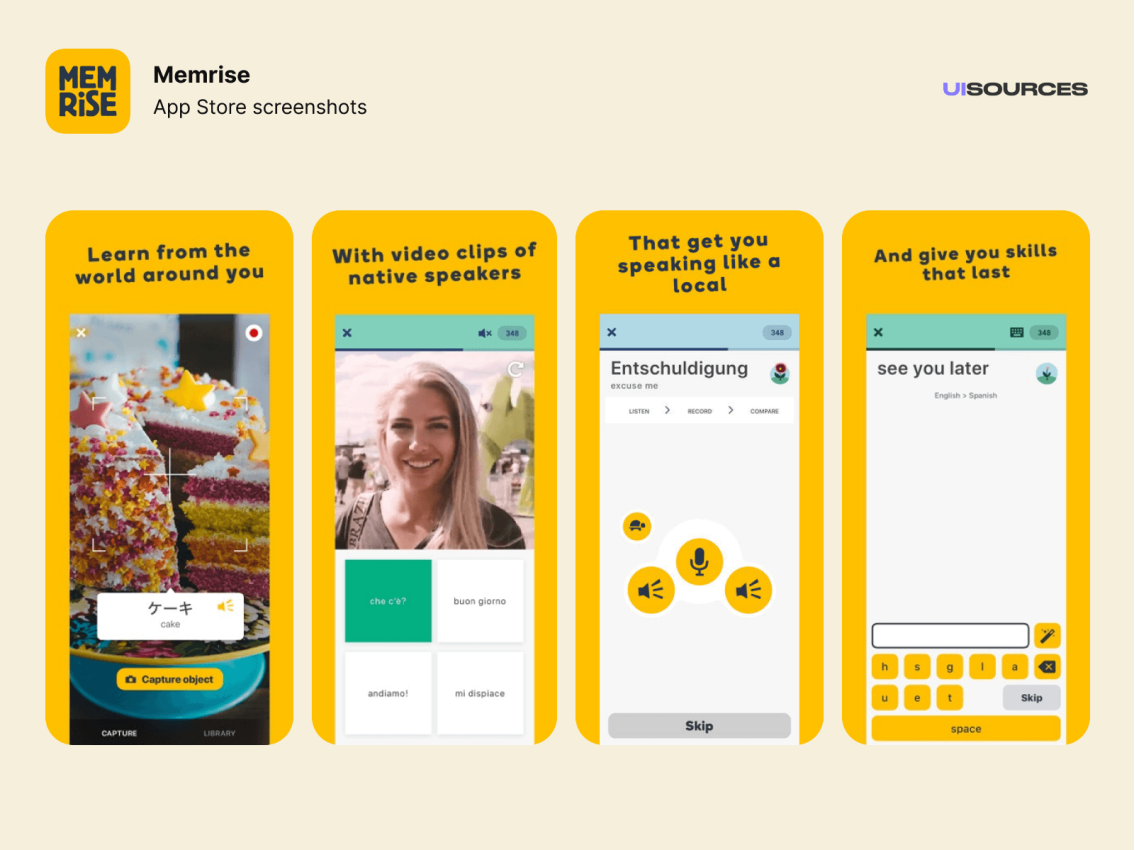

Рисунок 1.3 – Приклад інтерфейсу для додатку Memrise

Особливості:

- Гейміфікація: використання ігрових елементів для підвищення мотивації та залучення користувачів.

- Інтервальне повторення: вбудована система повторення для закріплення знань.
- Відео з носіями мови: можливість переглядати відео з носіями мови для покращення вимови та слухового сприйняття.

1.6 Висновок до розділу 1

Врахування індивідуальних особливостей сприйняття та запам'ятовування інформації дозволяє створити більш ефективні та адаптовані методи . Це сприяє підвищенню продуктивності та якості життя різних груп користувачів, підтверджуючи актуальність та важливість розвитку інформаційних технологій для запам'ятовування інформації .

Таким чином, у цьому розділі було проведено дослідження найкращих підходів до вирішення задачі запам'ятовування інформації за допомогою інформаційних технологій. Досліджено різні методи та техніки покращення когнітивних функцій, зокрема візуальні, аудіальні та кінестетичні підходи, їх переваги і недоліки. Було проаналізовано наявні програмні засоби

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЗАПАМ’ЯТОВУВАННЯ ІНФОРМАЦІЇ

2.1 Розробка структури інформаційної системи запам’ятовування інформації

Розробка моделі інформаційної технології запам'ятовування інформації є важливим етапом, що передбачає створення структурованої архітектури, яка забезпечує ефективний обробіток і зберігання даних. Основними компонентами моделі є користувацький інтерфейс, система управління базами даних (СУБД), обробка інформації, модулі інтерактивності та аналітика. Користувацький інтерфейс повинен бути інтуїтивно зрозумілим та адаптивним до потреб користувачів, що забезпечить зручність взаємодії. Система управління базами даних відповідатиме за зберігання та управління даними, надаючи безпечний доступ до інформації. Обробка інформації включає аналіз даних, створення мнемонічних асоціацій та застосування алгоритмів інтервального повторення, що сприяють запам'ятовуванню. Модулі інтерактивності забезпечують елементи гейміфікації, взаємодію між користувачами та можливість отримання зворотного зв'язку. Аналітичні інструменти дозволять збирати та аналізувати дані про прогрес користувачів, адаптуючи навчальні стратегії.

Модель інформаційної технології є абстрактним відображенням її структури та функціоналу, що включає всі компоненти, їхні взаємозв'язки та внутрішні процеси. Така модель надає загальну картину системи і слугує важливим інструментом для її розробки, аналізу та документування.

При моделюванні структури ми описуємо елементи системи та їхні взаємозв'язки. UML зазвичай використовується як об’єктно-орієнтована мова моделювання, тому основними складовими є класи і зв’язки між ними. У конкретний момент роботи системи можна виділити кінцевий набір конкретних об’єктів (екземплярів класів) і наявних між ними зв’язків. Однак під час

функціонування цей набір не є статичним: об’єкти створюються та знищуються, а зв’язки встановлюються і руйнуються.[10]

Діаграми варіантів використання надають візуальне уявлення про вимоги користувачів. Діаграма Use Case описує поведінку системи з точки зору користувача і слугує для виявлення вимог до розроблюваної системи, а також для їх фіксації у формі, що полегшує подальшу розробку. Діаграми варіантів використання включають елементи Use Case, акторів і зв’язки залежності, узагальнення та асоціації. Як і інші діаграми, діаграми Use Case можуть містити примітки та обмеження.

Залежно від цілей процедури розрізняють такі варіанти використання: основні (базові), які забезпечують необхідну функціональність; допоміжні, що надають можливість налаштування системи; та додаткові, які пропонують зручності для користувача. В загальному, модель включає внутрішню систему, що моделюється, зовнішнє середовище і зв’язок між системою та зовнішнім оточенням у вигляді асоціацій.

Структура інформаційної технології для запам’ятовування інформації повинна включати в себе наступні компоненти:

Система введення даних: Це перший етап роботи системи, на якому користувач вводить або завантажує інформацію, яку він планує запам'ятати. Це може бути текстова інформація, дати, зображення або навіть мультимедійні файли. Для полегшення введення даних можна використовувати інтерфейс із можливістю завантаження даних у різних форматах, наприклад, текстових документів чи презентацій.

- Система зберігання та організації даних: Ця складова відповідає за структурування інформації у відповідні сховища, де вона буде зберігатися для подальшого оброблення. Інформація може бути організована за темами, категоріями, пріоритетами або термінами запам'ятовування. Використання баз даних дозволяє ефективно організовувати доступ до цієї інформації для швидкого пошуку та зворотного відтворення.

- Модуль нагадування: Одним із ключових аспектів системи є механізм нагадування. Він відповідає за регулярне надсилання користувачам сповіщень про необхідність повторити чи переглянути запам'ятовання інформацію. Цей модуль може бути налаштований на основі різних інтервалів (згідно з методами інтервального запам'ятовування) або на основі специфічних подій (наприклад, дати іспиту чи терміну виконання завдання).
- Модуль аналізу прогресу: Ще однією важливою частиною є система збору й аналізу даних про успіхи користувача у запам'ятовуванні. Вона дає змогу адаптувати частоту нагадувань і методи подачі матеріалу, виходячи з ефективності процесу запам'ятовування кожного конкретного користувача.
- Модуль персоналізації: Виходячи з результатів користувача, система може змінювати алгоритм подачі інформації: скорочувати інтервали для важкої інформації, збільшувати їх для легко запам'ятовуваних тем або додавати нові нагадування для зміцнення вже засвоєного матеріалу.Цільовою аудиторією системи є люди, які прагнуть ефективно запам’ятовувати та організовувати інформацію, використовуючи сучасні технології. Основним завданням системи є забезпечення користувачів необхідною інформацією, а також можливістю організовувати свій навчальний процес.

Існує безліч стандартів для створення правильної архітектури системи, а також для розробки та інтеграції програмних рішень. Застосування цих стандартів значно підвищить шанси на успішне створення системи і її подальше безвідмовне функціонування. Однак доцільність їхнього застосування повинна визначатися на етапі проектування, оскільки складність системи при їх інтеграції може істотно зрости.

2.2 Математична модель інформаційної технології запам’ятовування інформації

Метод запам’ятовування, що використовується в інформаційній системі, повинен враховувати когнітивні особливості користувача та ґрунтуватися на наукових підходах до зберігання інформації в довготривалій пам’яті. Одним із найефективніших методів є інтервальне повторення, яке ґрунтується на принципах "кривої забування" Германом Еббінгаузом.

Принцип "кривої забування", розроблений німецьким психологом Германом Еббінгаузом у 19 столітті, описує, як швидко інформація втрачається з пам'яті з часом. Він виявив, що більшість знань забувається дуже швидко після їх отримання, причому найбільше інформації втрачається в перші години або дні. Згідно з його дослідженнями, відсоток згаданої інформації знижується, проте темпи забування сповільнюються з часом.

Еббінгауз також підкреслив важливість повторення для покращення збереження інформації в пам'яті. Чим частіше ми повторюємо матеріал, тим більше шансів, що він залишиться в пам'яті на тривалий період. Крім того, він звернув увагу на різницю між активним і пасивним запам'ятовуванням. Активне навчання, яке включає практику чи використання інформації, значно сприяє кращому закріпленню знань.

Важливим аспектом, виявленим у дослідженнях Еббінгауза, є ефект інтервалів. Повторення інформації через певні проміжки часу призводить до кращого засвоєння, ніж навчання безперервно за короткий час. Таким чином, крива забування має значний вплив на освітні практики та методи навчання, адже розуміння механізмів пам'яті може допомогти розробити ефективні стратегії, які підвищать успішність засвоєння інформації.

Крива має характерну форму, схожу на логарифмічну спадну криву. На початку вона стрімко падає, а потім вирівнюється, вказуючи на поступове уповільнення втрати інформації.

Практичне застосування "кривої забування" знайшло широке використання в освітніх методиках та інструментах, спрямованих на покращення ефективності навчання та закріплення знань. Одним із ключових підходів є використання інтервального повторення, яке враховує особливості пам'яті та принципи забування, відкриті Еббінгаузом. Цей метод передбачає регулярне повернення до вивченого матеріалу через певні проміжки часу. Наприклад, інформація повторюється через день після початкового засвоєння, потім через три дні, через тиждень і так далі. Така стратегія дозволяє максимально ефективно переміщувати знання з короткочасної пам’яті до довготривалої.

Інший важливий спосіб практичного застосування кривої забування — це активне відтворення матеріалу. Замість того щоб просто перечитувати текст чи переглядати записи, учень намагається самостійно відтворити основні ідеї чи деталі. Такий підхід не лише закріплює знання, але й активізує пам'ять, стимулюючи її ефективніше працювати. Наприклад, замість механічного заучування нових термінів, людина намагається пояснити їх своїми словами або знайти асоціації, що пов’язують нову інформацію з уже відомою.

Крім того, враховуючи криву забування, особливу увагу приділяють контекстуальному навчанню. Інформація краще запам'ятовується, коли вона має чітке практичне значення або зв'язок із реальними життєвими ситуаціями. Це означає, що навчальні матеріали мають бути представлені у формі, яка відповідає цілям і потребам учня. Наприклад, для вивчення іноземної мови можна застосовувати методику навчання через діалоги або практичні вправи, де слова використовуються у контексті, а не в ізольованих списках.

Викладачі й розробники навчальних програм також можуть використовувати принципи кривої забування для побудови своїх курсів. Наприклад, структурування занять таким чином, щоб ключові поняття повторювалися через певні проміжки часу, сприяє їх кращому засвоєнню. Сучасні онлайн-платформи, що використовують адаптивні алгоритми, автоматично аналізують, які аспекти матеріалу користувач забуває найшвидше, і пропонують додаткові вправи саме на ці теми.

Таким чином, знання про криву забування дозволяє не лише зрозуміти, як працює пам’ять, але й використовувати ці принципи для розробки ефективних стратегій навчання, які роблять процес засвоєння інформації більш продуктивним і довготривалим.[11]

Основні етапи методу запам'ятовування включають:

Визначення типу інформації: Різні типи інформації потребують різних підходів до запам'ятовування. Наприклад, текстова інформація може вимагати більше повторень у порівнянні з візуальною інформацією. В системі можна налаштувати різні режими роботи залежно від контенту (зображення, тексти, числа).

Розподіл інформації на блоки: Для полегшення засвоєння інформації вона повинна бути розбита на менші логічні частини. Це може бути досягнуто шляхом створення тематичних блоків, розподілу інформації за рівнями складності або важливості для користувача.

Інтервальне повторення: Основний принцип полягає в тому, що після першого засвоєння інформація повторюється через визначені часові проміжки. Ці інтервали можуть адаптуватися в залежності від результатів користувача, поступово збільшуючись або скорочуючись, залежно від того, як добре користувач запам'ятав матеріал. Система на основі аналізу активності може збільшувати або зменшувати частоту повторення.[12]

Використання мнемонічних технік: У системі також можуть бути впроваджені методи мнемоніки, такі як асоціативне запам’ятовування, візуальні образи, рифми та інші техніки для полегшення процесу засвоєння складної інформації.

Аналіз і зворотний зв'язок: Після кожного повторення система фіксує результати користувача, що дозволяє їй налаштувати подальший план навчання. Наприклад, якщо користувач успішно відтворює матеріал, наступне повторення може бути відкладене на довший період.

Опишемо математичну модель інформаційної технології що розробляється.

$C=\{c_1,c_2,\ldots,cn\}$ – множина карток. Кожен елемент c_i є окремою карткою, яка містить інформацію (наприклад, питання та відповідь) для запам'ятовування. Кардинальність множини $|C|=n$ відображає загальну кількість карток у базі даних системи.

$U=\{u_1,u_2,\ldots,u_m\}$ – множина користувачів. Кожен елемент uiu_iui представляє користувача, який може вивчати картки. $|U|=m$ є кількістю користувачів у системі.

$T=\{t_1,t_2,\ldots,t_k\}$ – множина тегів. Кожен елемент t_i є унікальним тегом, що характеризує певні категорії карток, які полегшують пошук і фільтрацію. Кардинальність множини $|T|=k$ – загальна кількість тегів.

$S=\{s_1,s_2,\ldots,s_l\}$ – множина статусів карток, які визначають, чи картка є "відомою" чи "невідомою". Кардинальність множини $|S|=l$.

Функція призначення статусу картки користувачем $f_1:C\times U\rightarrow S$, де $f_1(c,u)=s$ – функція, яка визначає статус картки c для користувача u. Відображення f_1 визначає, чи є картка c відомою (s=1) або невідомою (s=0) для конкретного користувача.

Функція додавання тегів до карток $f_2:C\rightarrow 2^T$, де $f_2(c)=\{t_1,t_2,\ldots,t_j\}\subseteq T$ – функція, що призначає тег(и) для кожної картки. Це дозволяє здійснювати фільтрацію карток на основі тегів, де кожен t_i вказує на конкретну категорію картки.

Функція створення нової картки $C:(front,back,type,tags)\rightarrow C\cup\{c_{new}\}$ – операція, яка додає нову картку c_{new} до множини карток C. При створенні картки вводяться основні дані: front і back (передній і зворотний бік картки), type (тип картки), а також асоціюються відповідні теги $tags\subseteq T$.

Функція оновлення статусу картки $U:(c,u,s')\rightarrow S'$, де $S'=S\cup\{(c,u,s')\}$ – операція, що оновлює статус картки c для користувача u до нового статусу s'.

Функція фільтрації карток за статусом $\forall u \in U$, $F_{status}(s):C \rightarrow C' \subseteq C$, де $C'=\{c \in C | f_1(c,u)=s\}$ – підмножина карток, відфільтрованих на основі статусу (наприклад, показує лише "відомі" або "невідомі" картки для користувача).

Функція фільтрації карток за тегами $F_{tags}(T'):C \rightarrow C' \subseteq C$, де $C'=\{c \in C | T' \subseteq f_2(c)\}$ – підмножина карток, які відповідають певному набору тегів $T' \subseteq T$. Це відображення дозволяє здійснювати пошук карток за категоріями.

Функція доступу до карток $R:id \rightarrow C$, де id є унікальним ідентифікатором картки. Функція R забезпечує доступ до даних картки за її ідентифікатором.

Мінімізація кількості "невідомих" карток $\min \sum_{c \in C}(1-f_1(c,u))$ – зведення загальної кількості "невідомих" карток для користувача u до мінімуму, що сприяє ефективному вивченню матеріалу.

Оптимізація використання карток за категоріями $\max \sum_{c \in C, t \in T} f_2(c,t)$ – збільшення використання карток з різними тегами T, що сприяє різноманітності матеріалу, доступного для вивчення.

Ця модель описує структуру та операції системи запам'ятовування карток, враховуючи статус карток, теги, функції фільтрації та оптимізації, що дозволяють користувачам ефективно взаємодіяти з матеріалом для запам'ятовування.

2.3 Розробка алгоритму функціонування інформаційної технології запам’ятовування інформації

Розробка алгоритму інформаційної технології запам’ятовування інформації передбачає ряд ключових процесів, які забезпечують ефективне функціонування системи. Першим етапом є збір та обробка вхідної інформації. На цьому етапі система приймає дані, введені користувачем, і структурує їх для подальшого використання. Наприклад, коли користувач вводить текстові дані,

дати або завантажує зображення, система аналізує метадані або ключові слова, щоб визначити оптимальний спосіб організації матеріалу для запам'ятовування.

Наступним кроком є створення індивідуального плану запам'ятовування, який генерується на основі аналізу введених даних. Цей план містить частоту повторень, методи подачі інформації та інтервали між повтореннями. Для складнішої інформації система може також використовувати мнемонічні техніки або підказки, що сприяють кращому запам'ятовуванню.

Третій етап включає інтервальне нагадування, коли система генерує сповіщення або нагадування для користувача відповідно до розробленого плану запам'ятовування. Наприклад, користувач отримує повідомлення, що настав час повторити певний блок інформації, що допомагає закріпити матеріал у пам'яті.

Після кожного повторення система переходить до адаптації плану. Вона збирає дані про те, наскільки успішно користувач відтворив матеріал. Якщо інформація була запам'ятована добре, інтервали між повтореннями можуть бути збільшені. В разі виникнення труднощів із запам'ятовуванням, інтервали скорочуються, а інформація подається частіше.

Останнім етапом є корекція результатів та оптимізація алгоритму. На основі аналізу результатів система може коригувати алгоритм, зменшуючи кількість повторень для легких тем і збільшуючи їх для складних. Це також включає оптимізацію способу подачі матеріалу, наприклад, через візуальні підказки або аудіовказівки.

Загалом, алгоритм має забезпечувати баланс між кількістю інформації для повторення та комфортом користувача. Це важливо, щоб уникнути перевантаження зайвими нагадуваннями, водночас гарантуючи ефективне закріплення матеріалу в пам'яті.

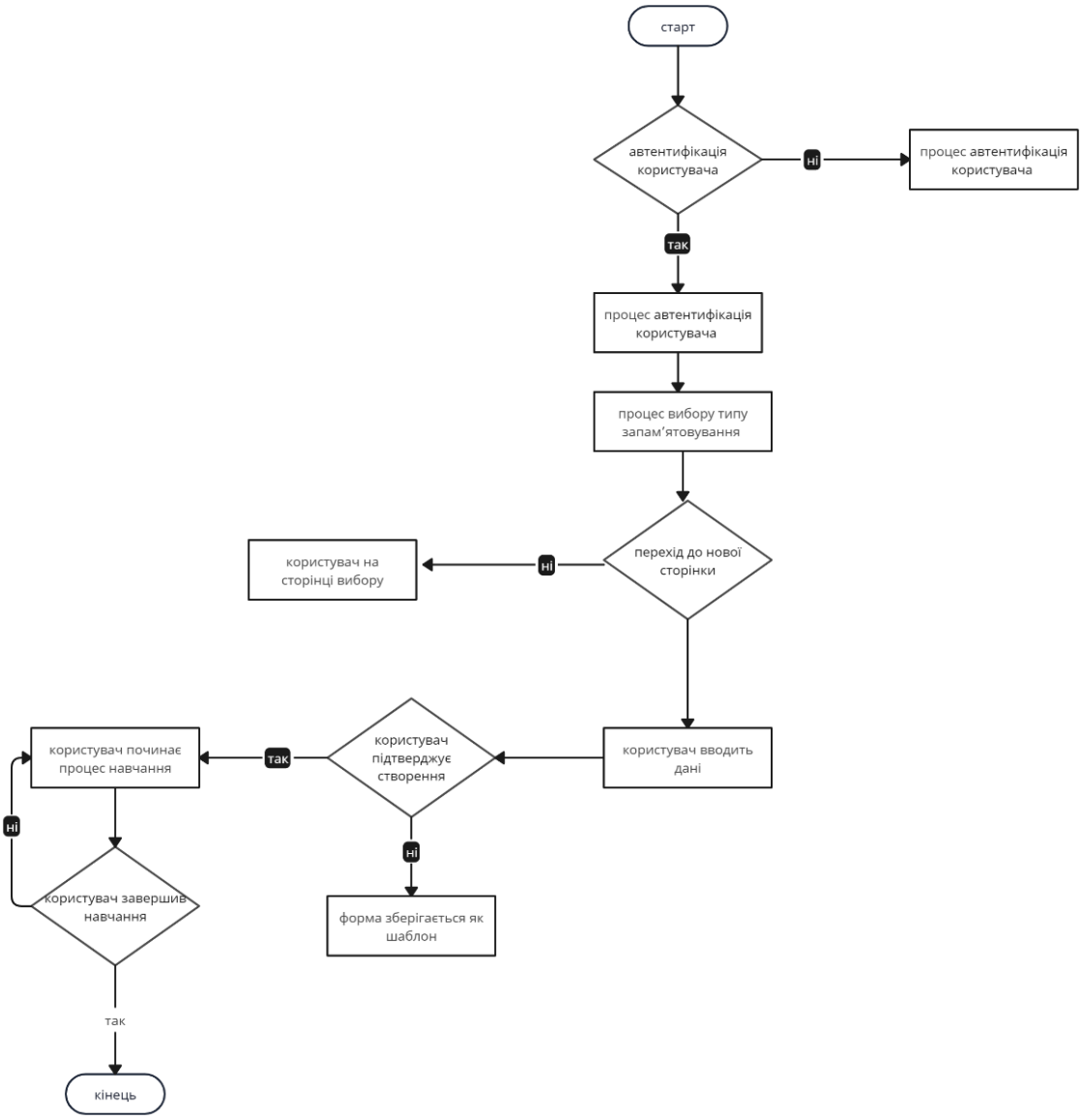

Рисунок 2.1- Схема алгоритму функціонування інформаційної технології запам'ятовування інформації

Алгоритм, представлений на схемі, описує послідовний процес роботи з інформаційною системою, що передбачає взаємодію користувача із системою для виконання певної задачі. Його структура враховує логіку перевірки доступу, вибору дій, заповнення інформації, перевірки результатів та завершення операцій.

На першому етапі відбувається старт процесу, який ініціює взаємодію користувача з системою. Після запуску система активує механізм аутентифікації

користувача. На цьому етапі користувач вводить свої облікові дані (логін та пароль), а система перевіряє їх коректність. Якщо дані правильні, користувач отримує доступ до наступного етапу. У разі невдалої аутентифікації система може повернути користувача назад для повторного введення даних або завершити процес.

Після успішної аутентифікації запускається процес вибору типу запиту або дії, який визначає, яку саме операцію бажає виконати користувач. Тут система пропонує список доступних опцій, наприклад, створення нового запиту, редагування існуючого чи інші функції. На цьому етапі користувач може залишитися на сторінці вибору, якщо він ще не визначився, або перейти до конкретної дії, вибравши потрібний варіант.

Якщо користувач вибрав створення нового запиту, він переходить до наступного етапу — введення даних. Цей крок є центральним, оскільки саме тут користувач вводить всю необхідну інформацію, яка буде використана для створення запиту. Введення даних може включати текстові поля, списки вибору або інші інтерактивні елементи форми. Система забезпечує перевірку введених даних на коректність, наприклад, через валідацію формату, обов'язковість полів тощо.

Після заповнення даних система пропонує підтвердити створення. Цей крок важливий для перевірки того, чи вся введена інформація відповідає очікуванням користувача. Якщо дані коректні, користувач підтверджує створення, і система переходить до генерації результату. У разі, якщо дані потребують змін, користувач може повернутися до попереднього етапу для редагування.

Далі система виконує генерацію або повернення шаблону, який слугує як проміжним результатом, так і кінцевим продуктом залежно від контексту. Шаблон може бути сформований автоматично на основі введених даних або пропонуватися користувачеві для подальшого налаштування. Користувач має можливість переглянути шаблон, внести в нього правки, якщо це необхідно, або зберегти.

На завершальному етапі користувач отримує повідомлення про успішне виконання операції. Якщо всі дії виконано правильно, процес завершується. У випадку, якщо користувач потребує внести корективи або повторити дії, система дозволяє повернутися на попередні етапи, що робить алгоритм гнучким і зручним.

Цей алгоритм реалізує лінійно-розгалужену структуру з можливістю повторення дій. Його можна застосовувати для багатьох інформаційних систем, зокрема для роботи з базами даних, формування звітів або автоматизації робочих процесів. Ключовими особливостями цього процесу є орієнтація на зручність для користувача, адаптивність на кожному етапі та контроль над усіма введеними даними.

2.4 Аналіз особливостей інформаційної технології для запам’ятовування інформації

Аналізуючи особливості інформаційних технологій для запам’ятовування інформації, варто звернути увагу на ключові аспекти, які визначають їхню ефективність, гнучкість та зручність використання. Такі технології спрямовані на оптимізацію навчального процесу, підвищення ефективності засвоєння знань і закріплення їх у довготривалій пам’яті. Основними особливостями цих технологій є інтерактивність, адаптивність, інтервальне повторення, можливість індивідуалізації навчання та використання різноманітних форм подання інформації.

Інтерактивність полягає в активному залученні користувача до процесу навчання. Це можуть бути інтерактивні завдання, такі як тести, кросворди чи картки з питаннями та відповідями. Завдяки інтерактивним елементам користувач не лише пасивно сприймає інформацію, але й активно взаємодіє з нею, що підвищує ефективність запам'ятовування.

Адаптивність системи є ще однією важливою особливістю. Сучасні технології, такі як платформи для вивчення мов чи освоєння професійних

навичок, аналізують прогрес користувача та підлаштовуються під його рівень знань. Наприклад, система може автоматично збільшувати складність завдань або змінювати інтервали повторення залежно від того, наскільки добре користувач запам'ятав матеріал. Така гнучкість дозволяє забезпечити ефективне навчання для кожного індивідуального користувача.

Інтервальне повторення є основою багатьох сучасних програм, розроблених для запам’ятовування інформації. Цей підхід базується на принципах кривої забування та передбачає повторення інформації через поступово зростаючі проміжки часу. Програми на кшталт Anki чи SuperMemo автоматично нагадують користувачу про необхідність повторення того чи іншого матеріалу, тим самим закріплюючи його в довготривалій пам’яті.

Індивідуалізація навчання є ще однією визначальною особливістю таких технологій. Завдяки можливості створення персоналізованих програм навчання кожен користувач може обирати власний темп, обсяг матеріалу, а також спосіб його вивчення. Наприклад, деякі платформи дозволяють обирати між візуальними, текстовими чи аудіоформатами для подання інформації, що сприяє максимальному комфорту і зручності навчання.

Також важливим є використання різноманітних форм подання інформації. Це можуть бути текстові матеріали, інфографіки, відео чи аудіо. Різні типи контенту дозволяють залучити різні канали сприйняття, що полегшує запам’ятовування. Наприклад, деякі люди краще засвоюють інформацію через зорові образи, інші — через слухові.

Ще однією особливістю інформаційних технологій для запам’ятовування є можливість відстеження прогресу користувача. Багато сучасних систем надають детальну статистику про те, які теми засвоєні добре, а які потребують додаткового повторення. Це дає змогу користувачу та викладачу виявити слабкі місця та спрямувати зусилля саме на їх покращення.

Таким чином, інформаційні технології для запам’ятовування інформації забезпечують індивідуальний підхід до навчання, підвищують ефективність засвоєння матеріалу та дозволяють інтегрувати сучасні методи повторення для

довготривалого збереження знань у пам’яті. Вони є незамінними інструментами як у самостійному навчанні, так і в освітніх установах, де потрібно враховувати різний рівень підготовки учнів та їхні індивідуальні потреби.

2.5 Обґрунтування вибору компонентів інформаційної технології, що розробляється

Проектування інформаційної технології для запам'ятовування інформації полягає в створенні системи, яка допомагає ефективно зберігати та повторювати отриману інформацію з метою її закріплення в пам'яті користувача. Це може бути застосовано в різних сферах, від навчальних платформ до особистих помічників для підтримки когнітивних процесів. Для такого проектування важливо врахувати кілька основних компонентів, які забезпечують оптимізацію навчального процесу та інтерфейс користувача.

Одним із центральних елементів системи є база даних, яка зберігає інформацію, що підлягає запам'ятовуванню, а також історію всіх повторень та змін, що відбулися з матеріалом. Це дозволяє відслідковувати прогрес користувача, визначати, яка інформація вимагає більш частих повторень, а яка вже засвоєна і може бути повторена рідше. Важливо, щоб база даних була гнучкою, здатною адаптуватися до різних видів інформації (текст, зображення, відео) і підтримувала різні стратегії повторення, такі як інтервальне повторення за моделями кривої забування.

Інтерфейс користувача має бути інтуїтивно зрозумілим, щоб користувач міг без зусиль додавати нові матеріали для вивчення, змінювати їх чи відстежувати свій прогрес. Для досягнення ефективного навчання інтерфейс повинен дозволяти користувачеві взаємодіяти з різними видами контенту, ставити завдання для себе, оцінювати рівень засвоєння матеріалу і адаптувати навчальний процес під свої потреби.

Сервер, як основна обчислювальна одиниця, обробляє запити користувача та здійснює взаємодію з базою даних. Він обробляє введені користувачем дані та

визначає, коли і які матеріали необхідно повторити на основі алгоритмів для оптимізації процесу запам'ятовування. Сервер також контролює збереження даних користувача, здійснює оновлення прогресу та надає зворотний зв'язок. Веб-клієнт є основою для доступу користувачів до системи через браузер. Він зручний для використання і дозволяє користувачам взаємодіяти з системою для додавання нової інформації, перевірки прогресу та виконання тестів на запам'ятовування. Крім того, він надає можливість персоналізувати налаштування, вибирати типи контенту та переглядати звіти про власний прогрес.

Система також може включати модуль інтеграції з іншими навчальними платформами або сервісами для автоматичного перенесення даних, що дозволяє користувачам комбінувати різні джерела інформації та отримувати персоналізовані поради. Інтеграція з системами відстеження прогресу та рекомендаційними системами дозволяє автоматично коригувати програми навчання в залежності від ефективності кожного користувача.

У проектуванні такої технології також необхідно враховувати фактор зручності інтерфейсу, адаптованого під різні платформи, а також підтримку різних медіа-форматів, що дозволяє задовольнити різні потреби користувачів у навчанні. Це підвищить мотивацію та результативність процесу навчання, дозволяючи краще закріплювати інформацію в пам'яті і використовувати її на практиці.

Основна мета інформаційної технології для запам'ятовування інформації полягає в оптимізації процесу збереження і засвоєння матеріалу користувачем. Приклад структурної схеми наведено на рисунку 2.2.

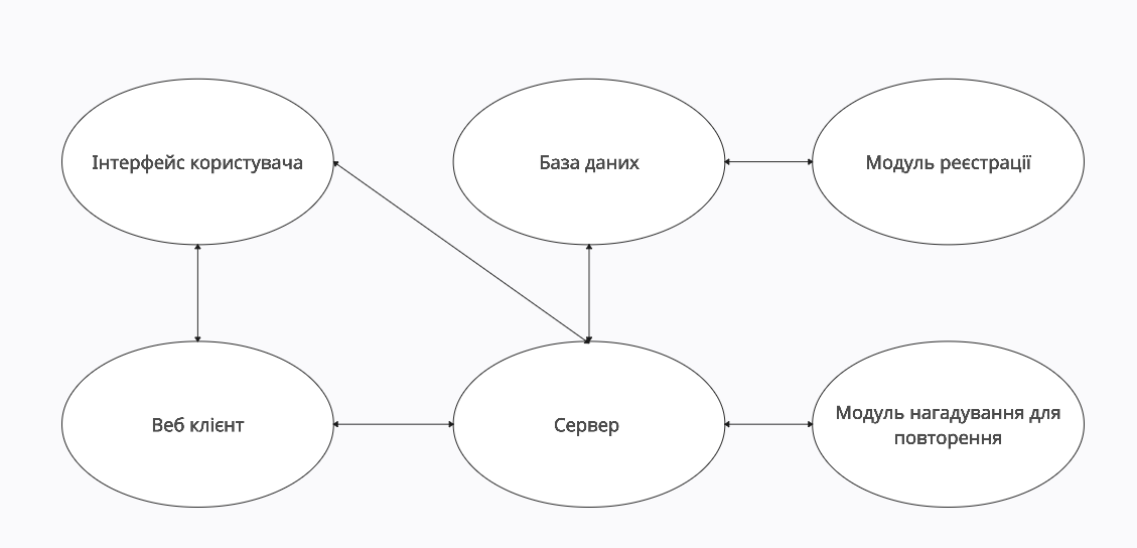

Рисунок 2.2 – Структурна схема взаємодії компонентів технології

Це досягається через ефективне управління інформаційними ресурсами, що дозволяє забезпечити швидкий доступ до необхідного контенту, зручний процес повторення та оцінки прогресу. Впровадження цієї технології сприяє підвищенню ефективності навчання, покращенню запам'ятовування та скороченню часу, необхідного для засвоєння нової інформації.

2.6 Висновок розділу 2

У цьому розділі було розглянуто основні аспекти розробки інформаційної системи для запам’ятовування інформації, а також методи й алгоритми, що лежать в її основі. Запропонована система дозволяє адаптувати процес запам'ятовування до індивідуальних потреб користувача, використовуючи інтервальне повторення, персоналізацію нагадувань та аналіз результатів для підвищення ефективності. Модель та алгоритм забезпечують надійний інструмент для оптимізації процесу засвоєння інформації, знижуючи когнітивне навантаження на користувача та сприяючи кращому засвоєнню великих обсягів матеріалу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ЗАПАМ'ЯТОВУВАННЯ ІНФОРМАЦІЇ

3.1 Обґрунтування вибору мови та середовища програмування для технології для запам'ятовування інформації

3.1.1 JavaScript

JavaScript — це потужна мова програмування, яка розвивалася від інструменту для надання інтерактивності вебсторінкам до універсальної мови, що працює і на клієнтській, і на серверній стороні завдяки платформі Node.js. Його головними характеристиками є інтерпретована природа та динамічність, що дозволяє виконувати код без компіляції безпосередньо у браузері або на сервері. Це надає розробникам гнучкість у швидкому впровадженні змін і тестуванні коду. JavaScript є слаботипізованою мовою, тому змінні в ньому не мають суворо фіксованих типів даних, що дає додаткову свободу у роботі з ними, хоча й може спричиняти помилки у складних проєктах.

Однією з особливих переваг JavaScript є підтримка асинхронного програмування, що здійснюється за допомогою колбеків, об'єктів Promise, та синтаксису async/await. Це робить мову чудовим вибором для програм, які потребують обробки тривалих операцій, як-от мережеві запити, без блокування інших процесів. JavaScript також підтримує об'єктно-орієнтоване програмування, хоча використовує прототипне наслідування, яке відрізняється від класичного підходу. Синтаксис класів було додано в ECMAScript 6 (ES6), що ще більше розширило можливості мови.

Основною областю застосування JavaScript залишається фронтенд-розробка, де він забезпечує динамічну зміну елементів на сторінці та взаємодію з користувачем через Document Object Model (DOM). Його екосистема включає фреймворки, такі як React, Angular і Vue.js, які спрощують створення сучасних вебдодатків. Крім того, Node.js перетворив JavaScript у мову для серверної

частини, дозволяючи створювати бекенд-застосунки, підтримувати бази даних і обробляти високий обсяг з'єднань, що підходить для сайтів із великим трафіком. JavaScript навіть використовується для кросплатформних мобільних застосунків завдяки React Native та Ionic, а також проникає у сферу вбудованих систем із Node.js, хоча поки цей напрямок менш поширений.

JavaScript вирізняється простотою та гнучкістю, що робить його доступним для новачків, а також великою спільнотою, яка підтримує проєкти, створює безліч бібліотек, фреймворків і документації. [13]

3.1.2 Java

Java – це потужна, об’єктно-орієнтована, строго типізована мова програмування, створена компанією Sun Microsystems у 1995 році (зараз розвивається Oracle). Вона вирізняється своєю універсальністю, завдяки чому широко використовується в корпоративних системах, веб-розробці, мобільних застосунках та навіть у вбудованих системах. Її слоган «Write Once, Run Anywhere» (WORA) відображає головну перевагу: Java-програми можуть виконуватися на різних платформах без змін у коді. Це досягається завдяки Java Virtual Machine (JVM), яка компілює код Java в байт-код, що потім виконується на будь-якій платформі з установленою JVM.

Однією з ключових характеристик Java є сувора об’єктно-орієнтована модель, де все побудовано навколо класів і об’єктів. Мова підтримує такі концепції, як спадковість, поліморфізм, інкапсуляція та абстракція, що полегшує створення структурованого та масштабованого коду. Завдяки цьому Java популярна в розробці великих і складних систем, де важлива надійність і можливість підтримки. Крім того, Java – це строго типізована мова, що означає, що тип кожної змінної має бути визначеним, а конвертація між типами є жорстко контрольованою. Це сприяє виявленню помилок на етапі компіляції, що робить код стабільнішим.

Java також підтримує багатопотоковість, тобто здатність виконувати кілька потоків одночасно, що робить її ідеальною для створення програм, що потребують обробки багатьох завдань паралельно (наприклад, вебсервери). Крім того, Java має автоматичний збирач сміття (Garbage Collector), який забезпечує автоматичне керування пам’яттю, звільняючи програміста від потреби самостійно очищати непотрібні об’єкти.

Java відома своєю величезною екосистемою та широким спектром бібліотек і фреймворків, що значно спрощує розробку. Зокрема, для веброзробки Java пропонує такі фреймворки, як Spring і Hibernate, які спрощують взаємодію з базами даних, управління залежностями та іншими аспектами створення вебдодатків. Для мобільних додатків Java була основною мовою для Android, хоча з появою Kotlin її використання на цій платформі поступово зменшується.

Java має велику й активну спільноту, яка забезпечує розробників підтримкою, документацією та інструментами для підвищення продуктивності. Завдяки поєднанню стабільності, високої продуктивності та сумісності на різних платформах, Java залишається однією з найбільш популярних мов у сфері програмування. [14]

3.1.3 C#

C# – це об'єктно-орієнтована, строго типізована мова програмування, створена Microsoft у рамках платформи .NET. Спочатку вона призначалася для розробки корпоративних додатків під Windows, але зараз, завдяки кросплатформному фреймворку .NET Core (який пізніше еволюціонував у .NET 5 і новіші версії), підтримується на різних операційних системах. Це робить C# універсальною мовою, придатною для розробки широкого спектра програм: від настільних і мобільних додатків до вебсервісів, ігор та хмарних сервісів. Сильна типізація мови означає, що тип кожної змінної повинен бути заздалегідь визначеним, що дозволяє уникати багатьох помилок і забезпечує надійність коду.

Об’єктно-орієнтована природа C# включає такі принципи, як спадковість, поліморфізм, інкапсуляцію та абстракцію, що допомагає створювати структурований і легкий для підтримки код. Мова також надає механізм збору сміття, який автоматично звільняє пам'ять, зайняту об'єктами, які більше не використовуються. Це позбавляє розробників від необхідності керувати пам'яттю вручну, що підвищує стабільність та спрощує розробку складних програм.

C# також підтримує асинхронне програмування завдяки ключовим словам async і await, що робить її ефективною для виконання операцій, які можуть затримуватися, таких як мережеві запити або читання файлів. Мова постійно оновлюється та вдосконалюється, що робить її актуальною для сучасних потреб. У нових версіях з'являються нові функції, такі як записи (records) для створення немутованих структур даних, шаблонне матчінг та синтаксис для спрощеного написання програм, що забезпечує зручність і сучасність.

C# активно використовується для розробки різних типів програмного забезпечення. Вона широко застосовується для створення додатків для Windows завдяки підтримці Windows Forms та Windows Presentation Foundation (WPF). У веброзробці C# набув значної популярності завдяки ASP.NET, що є фреймворком для побудови динамічних вебсайтів, API та серверних застосунків. ASP.NET Core підтримує кросплатформну розробку, дозволяючи запускати вебзастосунки на Linux та macOS, що робить C# зручним вибором для серверних рішень. В ігровій індустрії C# є основною мовою для розробки на Unity – одному з найпопулярніших ігрових рушіїв, що дозволяє створювати ігри для різних платформ, від мобільних пристроїв до VR. Використовуючи Xamarin, розробники можуть створювати кросплатформні мобільні додатки на C# для Android та iOS, що дозволяє підтримувати єдиний код для різних платформ.

C# поєднує в собі надійність, високу продуктивність та структурованість, що робить її чудовим вибором для великих проєктів, особливо тих, які вимагають стабільності та підтримки на високому рівні. Часті оновлення мови та підтримка Microsoft забезпечують її актуальність для широкого кола сучасних

задач, а її багата екосистема і велика спільнота розробників дозволяють розширювати та удосконалювати можливості програмного забезпечення, створеного на C#.[15]

3.1.4 Python

Python — це мова програмування високого рівня, відома своєю простотою, читабельністю та універсальністю, що робить її надзвичайно популярною серед розробників на всіх рівнях. Створена Гвідо ван Россумом і випущена в 1991 році, Python здобула популярність завдяки зрозумілому синтаксису, який наближений до природної мови, що дозволяє швидко опановувати мову навіть новачкам і при цьому зберігати високу продуктивність для професійних розробників. Python є інтерпретованою мовою, що означає виконання коду рядок за рядком, що зручно для швидкої розробки та тестування.

Мова Python підтримує кілька парадигм програмування, включаючи об’єктно-орієнтоване, процедурне та функціональне програмування, що надає розробникам значну гнучкість у створенні програмного забезпечення. Завдяки динамічній типізації Python дозволяє оголошувати змінні без явного вказання їх типів, що спрощує код і пришвидшує його написання. Проте, для складних проєктів ця особливість може створювати труднощі, адже помилки можуть з’являтися під час виконання, а не на етапі компіляції, як у строго типізованих мовах.

Одним із головних аспектів Python у веброзробці є його фреймворки. Django та Flask – два найпоширеніші фреймворки, кожен з яких має свої особливості і сфери застосування. Django – потужний фреймворк із «батарейками в комплекті», тобто з великою кількістю вбудованих модулів, що дозволяють розробникам швидко створювати складні проєкти. Django підходить для створення повноцінних вебсайтів і додатків, оскільки пропонує вбудовані інструменти для роботи з базами даних, автентифікації користувачів, обробки форм, керування сесіями і навіть для створення адміністративної панелі. Він

відомий своєю архітектурою MVC (Model-View-Controller), яка чітко розділяє логіку, представлення й управління даними.

Flask, з іншого боку, – це мікрофреймворк, який надає мінімальний набір інструментів для веброзробки. Його гнучкість і простота роблять його популярним вибором для створення невеликих застосунків або API, де не потрібна повноцінна структура, як у Django. Flask дозволяє розробнику легко розширювати функціонал, додаючи сторонні бібліотеки, тому він популярний серед проєктів, де потрібно більше контролю над архітектурою додатку або де важлива мінімальна початкова структура.

Крім Django і Flask, існують інші фреймворки та бібліотеки для веброзробки на Python. Наприклад, FastAPI є новим фреймворком, оптимізованим для створення API, що підтримують асинхронне програмування, яке дозволяє обробляти велику кількість запитів одночасно. Він забезпечує високу продуктивність, що робить його ідеальним для створення сучасних RESTful API або мікросервісів.

Завдяки Python також можна працювати з базами даних (через ORM, як-от Django ORM, SQLAlchemy або Tortoise ORM) і обробляти HTML-шаблони (за допомогою Jinja2 у Flask або вбудованих засобів у Django). Ці можливості дають змогу легко інтегрувати бази даних, створювати динамічний контент і організовувати запити до серверів. Крім того, Python дозволяє обробляти безпеку додатків, управління доступом та автентифікацію користувачів за допомогою таких бібліотек, як Django Allauth і Flask-Login.

Сфера веброзробки на Python також активно підтримується бібліотеками для тестування, такими як Pytest і Unittest, що дозволяють автоматизувати перевірку вебдодатків та API. Інструменти тестування, поряд із простим і читабельним синтаксисом, полегшують підтримку та розвиток коду, що робить Python підходящим вибором для проєктів, які потребують стабільності й масштабованості.

Загалом Python завдяки своїм потужним фреймворкам і багатій екосистемі залишається однією з найбільш ефективних мов для розробки вебдодатків різного масштабу, від простих API до великих корпоративних платформ.[16]

3.1.5 Обґрунтування вибору мови програмування

Зробимо порівняльну таблицю 3.1 з вибраними мовами щоб краще зрозуміти яка краще підходить до контексту проекту.

Таблиця 3.1 – Порівняльна таблиця

Критерій	**JavaScript**	**Java**	**C#**	**Python**
Типізація	Динамічна, слабка типізація	Статична, сильна типізація	Статична, сильна типізація	Динамічна, слабка типізація
Призначення	Веброзробка (клієнт та сервер)	Підприємницькі додатки, мобільні додатки, веб (сервер)	Розробка для Windows, ігри, мобільні додатки (через Xamarin)	Веброзробка, наука про дані, машинне навчання, автоматизація
Парадигми програмування	Об'єктно-орієнтоване, функціональне, імперативне	Об'єктно-орієнтоване, імперативне, декларативне	Об'єктно-орієнтоване, імперативне, функціональне	Об'єктно-орієнтоване, функціональне, імперативне
Платформи	Веб (браузери), сервери (Node.js)	Кросплатформенність через JVM	Кросплатформенність через .NET	Кросплатформенність (Linux, Windows, macOS)
Продуктивність	Висока (особливо для браузерних додатків)	Висока, але залежить від JVM	Висока, з хорошою оптимізацією під Windows	Зазвичай нижча через інтерпретацію, але ефективна для багатьох задач
Застосування	Веб-розробка (клієнт та сервер)	Веб-розробка (Java EE, Spring), мобільні додатки (Android)	Розробка Windows-додатків, ігри (Unity), сервери	Веб-розробка (Flask, Django), наука про дані, автоматизація

Таблиця 3.1 – Продовження

Критерій	JavaScript	Java	C#	Python
Інструменти та Бібліотеки	React, Angular, Vue, Node.js	Spring, Hibernate, Apache Struts	.NET, ASP.NET, Unity	Django, Flask, TensorFlow, NumPy, Pandas
Підтримка спільноти	Велика, активно розвивається	Велика, з багатьма бібліотеками та інструментами	Велика, підтримка Microsoft	Велика, активно розвивається, велика кількість бібліотек

Python є чудовим вибором для багатьох типів проєктів завдяки його гнучкості, простоті, активній спільноті та широкій екосистемі бібліотек. Однією з головних причин вибрати Python є його зрозумілий та читабельний синтаксис, який дозволяє навіть початківцям швидко освоювати мову, а досвідченим розробникам писати чистий, добре структурований код. Це прискорює розробку та знижує ймовірність помилок. Python також підтримує різні парадигми програмування, як-от об’єктно-орієнтоване, функціональне та процедурне програмування, що надає гнучкість у виборі стилю та архітектури для проєкту.

Python має багато переваг для веброзробки, що робить його чудовим вибором для створення вебзастосунків, зокрема через його зрозумілий синтаксис і велику екосистему фреймворків та бібліотек. Зокрема, Python пропонує кілька відомих фреймворків для веброзробки, таких як Django і Flask. Django, зокрема, надає готову структуру та функції для розробки комплексних вебзастосунків, включаючи автентифікацію, маршрутизацію URL, обробку форм, а також інструменти для роботи з базами даних. Це дозволяє розробникам швидко перейти від ідеї до готового продукту, зосереджуючись більше на бізнес-логіці, ніж на налаштуванні базових компонентів. Flask, як мікрофреймворк, забезпечує більшу гнучкість та легкість, що дозволяє створювати мінімальні вебзастосунки або API, які легко інтегрувати з іншими сервісами.

Загалом, обрання Python для веброзробки виправдано його зручністю, потужністю інструментів, можливостями для інтеграції з різними сферами

розробки та підтримкою великої спільноти, що робить його хорошим вибором для створення як невеликих проєктів, так і масштабних вебзастосунків.

3.2 Вибір спеціалізованої бібліотеки для програмної реалізації технології для запам’ятовування інформації

Вибір бібліотеки для реалізації технології запам'ятовування інформації залежить від конкретних цілей, обсягу даних і типу зберігання (локального або серверного). Розглянемо кілька популярних бібліотек Python, які добре підходять для створення системи зберігання та обробки інформації.

3.2.1 **SQLite і бібліотека SQLite3**

Для невеликих обсягів даних, які можуть зберігатися локально. SQLite3 — легка бібліотека для Python, що дозволяє працювати з локальною базою даних SQLite, яка зберігається в одному файлі. Вона ідеально підходить для локального зберігання інформації, дозволяє створювати базові функції для роботи з даними, наприклад, додавання, видалення, пошук і оновлення записів. Це відмінний вибір для систем, де немає потреби в масштабованості чи складній інфраструктурі. [17]

3.2.2 TinyDB

JSON-база даних, яка добре підходить для зберігання невеликих обсягів даних. TinyDB зберігає дані у форматі JSON, дозволяє легко працювати з записами та не вимагає складного налаштування. TinyDB — це чудовий варіант для систем, де не потрібно великої кількості даних і де зберігання може бути організоване без використання повноцінної реляційної бази.[18]

3.2.3 MongoDB з бібліотекою PyMongo

MongoDB є нереляційною базою даних, яка працює за принципом «документного сховища» й зберігає дані у форматі BSON (бінарне кодування JSON). PyMongo — це офіційний драйвер для взаємодії з MongoDB на Python. MongoDB ідеально підходить для проєктів, де дані мають динамічну структуру або зберігаються в ієрархічному форматі, наприклад для зберігання складних об’єктів. MongoDB забезпечує високу масштабованість і підходить для вебзастосунків із великими обсягами даних.[19]

3.2.4 Обґрунтування вибору

SQLite є чудовим вибором для проєктів, які не потребують великої інфраструктури баз даних чи роботи з великими обсягами даних, завдяки своїй простоті, надійності та легкості у використанні. Як вбудована база даних, SQLite не вимагає окремого сервера, що спрощує інтеграцію та дозволяє уникнути додаткових налаштувань і встановлення серверних служб. Її дані зберігаються у форматі одного файлу, який легко переносити, що ідеально підходить для мобільних, десктопних і вбудованих застосунків з обмеженими ресурсами.

Продуктивність SQLite особливо помітна в невеликих та середніх проєктах, де вона забезпечує швидкий доступ до даних без надмірних витрат на оперативну пам’ять чи обчислювальні потужності. Підтримка основних функцій SQL дозволяє використовувати її як повноцінну реляційну базу, залишаючи можливість створення таблиць, індексації, транзакційності та забезпечення цілісності даних. У разі раптового завершення роботи програми SQLite забезпечує надійність зберігання завдяки ACID-властивостям, що гарантує цілісність записів.

Завдяки простоті розгортання, підтримці кросплатформеності й мінімальним вимогам до ресурсів, SQLite є універсальним вибором для багатьох застосунків. Вона добре підходить для локального зберігання та резервного

копіювання, а також забезпечує стабільну роботу на різних операційних системах, таких як Windows, Linux, macOS, iOS та Android. Для більш масштабних проєктів з великими обсягами даних та високим навантаженням краще використовувати серверні бази, як-от PostgreSQL чи MySQL, однак для невеликих програм SQLite пропонує оптимальне поєднання простоти та надійності.

3.3 Розробка UML-діаграми класів програми технології для запам’ятовування інформації

Для розробки UML-діаграми класів програми, що допомагає у запам’ятовуванні інформації, необхідно спочатку визначити основні компоненти системи та їх взаємодію. Припустимо, що наша програма включатиме базову структуру для створення і зберігання карток пам’яті (як у системах типу Anki чи Quizlet), організацію інформації за темами і категоріями, а також відстеження прогресу користувача. Наведемо ключові класи та зв’язки між ними.

1) **App**: Відповідає за конфігурацію програми та її запуск. Містить основні параметри конфігурації, такі як SECRET_KEY, USERNAME, та інші змінні.
2) **Database**: Виконує підключення до бази даних SQLite і забезпечує функції доступу до даних (connect_db(), get_db(), close_db(), init_db()).
3) **Card**: Відповідає за CRUD-операції з картками (додавання, редагування, видалення карток), а також за функції навчання (memorize, memorize_known), де можна відзначати картки як вивчені чи невивчені. Також цей клас обробляє фільтрацію карток за параметрами.
4) **Tag**: Виконує операції з тегами карток, включаючи створення, редагування та видалення тегів, а також їх отримання з бази даних.
5) **Request**: Відповідає за обробку HTTP-запитів, таких як login, logout, index, cards, tags, і за маршрутизацію до відповідних методів у класах Card

та Tag.Відповідає за обробку HTTP-запитів, таких як login, logout, index, cards, tags, і за маршрутизацію до відповідних методів у класах Card та Tag.

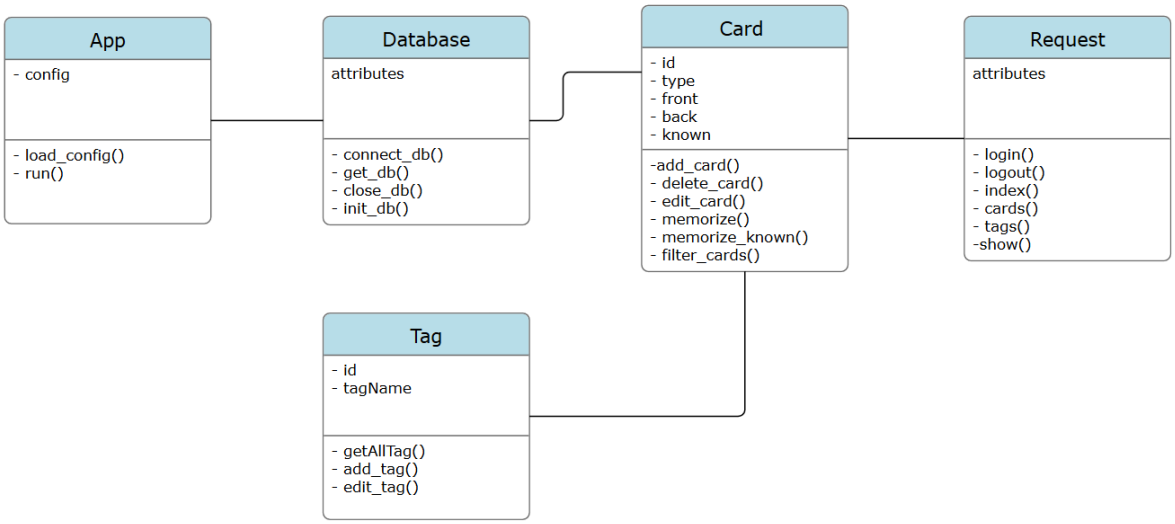

Рисунок 3.1 – UML діаграма класів

Пояснення UML-діаграми

- App клас є основним додатком, який ініціалізує конфігурацію та запускає сервер.
- Database клас відповідає за керування з'єднаннями з базою даних та виконує операції, такі як відкриття, ініціалізація, та закриття бази даних.
- Card клас представляє модель карток та операції, пов'язані з ними, як-от додавання, видалення, редагування та навчання.
- Tag клас відповідає за управління тегами, включаючи додавання нових тегів і редагування існуючих.
- Request клас представляє обробку HTTP-запитів, що дозволяє користувачам взаємодіяти з додатком через маршрути.

Ця UML-діаграма описує загальні класи та їх функціональність у контексті Flask-додатку для запам'ятовування інформації.

3.4 Програмна реалізація інформаційної технології для запам’ятовування інформації

3.4.1 Бібліотеки

Для створення веб-додатку на Python вам знадобляться кілька ключових бібліотек, які забезпечать основні функції, такі як робота з веб-серверами, маршрутизація, взаємодія з базами даних і інтерфейс користувача. Ось основні з них:

Flask — це мікрофреймворк для веб-розробки на Python, який був створений для того, щоб надати розробникам простий та легкий інструмент для створення веб-додатків. Основна ідея Flask полягає в тому, щоб залишити за розробником максимальний контроль над додатком, мінімізуючи вбудовані функції та дозволяючи додавати лише ті компоненти, які дійсно потрібні. Це робить його дуже гнучким і масштабованим, а також ідеальним для невеликих проектів, де важлива швидкість розробки.

Flask не має суворої структури або великої кількості попередньо визначених бібліотек, тому розробники можуть будувати додатки відповідно до своїх потреб. Однак цей мікрофреймворк має достатньо базових можливостей для створення простих веб-додатків. Наприклад, він дозволяє обробляти HTTP-запити, маршрутизувати URL-адреси, працювати з шаблонами HTML (за допомогою бібліотеки Jinja2), керувати сесіями користувачів, а також обробляти різні формати даних, такі як JSON.[20]

SQLite — це вбудована реляційна база даних, яка зберігає дані в одному файлі, що робить її ідеальним рішенням для додатків, які не потребують складної серверної інфраструктури. Її основною перевагою є простота використання та відсутність необхідності в окремому серверному програмному забезпеченні для зберігання даних. Всі дані зберігаються в одному файлі, що забезпечує зручність управління і транспортування бази даних між різними системами.

SQLite ідеально підходить для малих та середніх проектів, де не потрібно обробляти великі обсяги даних або забезпечувати високу навантажуваність. Вона забезпечує швидкий доступ до даних, оскільки не потребує підключення до віддаленого сервера — доступ здійснюється безпосередньо через програму, що робить її дуже ефективною для додатків, які не потребують складних розподілених систем. База даних може обробляти до кількох гігабайт даних, що підходить для більшості невеликих проектів.

SQLite також має низькі вимоги до системних ресурсів, що дозволяє використовувати її навіть на пристроях з обмеженими потужностями, таких як мобільні телефони чи вбудовані системи. Однак для великих систем, де є потреба в масштабуванні та забезпеченні високої доступності, SQLite може не бути оптимальним вибором. У таких випадках рекомендується використовувати більш потужні серверні бази даних.

Flask-Login — це бібліотека, призначена для реалізації механізму автентифікації та авторизації користувачів у веб-додатках, розроблених на Flask. Вона дозволяє зручно працювати з сесіями та забезпечує обробку процесу входу та виходу користувачів, що є важливою частиною безпеки веб-додатків. Flask-Login допомагає організувати логіку автентифікації без необхідності створювати складні рішення з нуля.

Бібліотека зберігає інформацію про користувачів в сесії, що дозволяє відстежувати стан їхнього входу між запитами. Це означає, що після автентифікації користувач може залишатися залогіненим під час навігації по веб-додатку, а також автоматично виходити після закінчення сеансу або за бажанням. Flask-Login надає прості функції для того, щоб додати механізм входу, виходу, а також для перевірки, чи користувач аутентифікований.

Requests — це популярна бібліотека для роботи з HTTP-запитами в Python. Вона значно спрощує процес взаємодії з веб-сервісами через протокол HTTP, дозволяючи відправляти запити до сторонніх API та обробляти їх відповіді. Це робить бібліотеку дуже корисною для інтеграції з іншими сервісами, збору даних з віддалених ресурсів або для взаємодії з веб-сайтами та веб-додатками.

Requests підтримує всі основні типи HTTP-запитів, включаючи GET, POST, PUT, DELETE та інші. Вона також дозволяє працювати з параметрами запитів, заголовками, даними форми, а також з різними форматами відповідей, такими як JSON, XML або просто текстові дані. Крім того, Requests має вбудовану підтримку для обробки cookies, сесій та автентифікації, що робить її дуже зручною для роботи з API, які вимагають додаткових заходів безпеки.

3.4.2 Загальна будова програми і коду

Додаток створено за допомогою Flask — популярного мікрофреймворку для Python, який дозволяє легко створювати веб-додатки. Він налаштовує базу даних SQLite, де зберігаються картки, теги та інша інформація. Перш за все, у коді є функції для підключення до бази даних, ініціалізації її та виконання SQL-запитів. Також є кілька функцій для отримання даних, зокрема для карток, які мають типи, передню та задню сторону, та статус (чи вивчено).

Створення карток у цьому веб-додатку реалізовано через кілька функцій, що дозволяють користувачам додавати нові картки, редагувати їх, а також відзначати їх як "вивчені" або "невивчені". Картки можуть мати різні типи (наприклад, загальні або спеціалізовані, наприклад, для коду), та кожна картка має передню й задню частину, що містить запитання та відповідь.

Щоб додати нову картку, користувач звертається до маршруту /add, де відправляється POST-запит з даними картки (тип, передня та задня частини). При додаванні картки виконується SQL-запит, який додає запис до таблиці cards в базі даних, зберігаючи ці дані.

Функція для додавання картки виглядає так:

```
@app.route('/add', methods=['POST'])
def add_card():
    if not session.get('logged_in'):
```

```
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO cards (type, front, back) VALUES (?, ?, ?)',
               [request.form['type'],
                request.form['front'],
                request.form['back']
                ])
    db.commit()
    flash('New card was successfully added.')
    return redirect(url_for('cards'))
```

Ця функція перевіряє, чи користувач авторизований, після чого зчитує дані картки з форми (тип картки, текст передньої та задньої частини). Зібрані дані потім зберігаються в базі даних за допомогою SQL-запиту. Після цього виконується коміт до бази, і користувач отримує сповіщення про успішне додавання картки.

Важливим аспектом є використання сесії для перевірки того, чи є користувач авторизованим. Якщо користувач не увійшов в систему, його буде перенаправлено на сторінку входу.

Після того як картка додана, користувач перенаправляється на сторінку cards, де відображається список усіх карток. Для фільтрації карток існують окремі маршрути, які дозволяють сортувати картки за типами або статусами (вивчено чи не вивчено).

Крім того, користувач може редагувати картки через маршрут /edit/<card_id>, де можна змінити передню та задню частини картки. Функція редагування використовує SQL-запит для оновлення запису в таблиці cards за певним ID картки.

```
@app.route('/edit/<card_id>')
def edit(card_id):
```

```
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, type, front, back, known
        FROM cards
        WHERE id = ?
    '''
    cur = db.execute(query, [card_id])
    card = cur.fetchone()
    tags = getAllTag()
    return render_template('edit.html', card=card, tags=tags)
```

Ця функція надає користувачеві можливість відредагувати картку. Вона вибирає картку з бази даних за допомогою SQL-запиту і передає її в шаблон для редагування. У шаблоні користувач може змінити текст передньої та задньої частини картки, а також вибрати теги для картки.

Крім того, є функція для позначення картки як вивченої, що змінює її статус у базі даних.

```
    @app.route('/mark_known/<card_id>/<card_type>')
    def mark_known(card_id, card_type):
        if not session.get('logged_in'):
            return redirect(url_for('login'))
        db = get_db()
        db.execute('UPDATE cards SET known = 1 WHERE id = ?', [card_id])
        db.commit()
        flash('Card marked as known.')
        return redirect(url_for('memorize', card_type=card_type))
```

Ця функція оновлює статус картки, позначаючи її як "вивчену". Після цього користувач перенаправляється до сторінки для вивчення карток.

Таким чином, процес створення карток включає кілька етапів: додавання нових карток через форму, редагування карток через спеціальну сторінку, позначення карток як вивчених та фільтрацію їх за різними критеріями. Всі ці операції здійснюються через простий і зручний інтерфейс, що дозволяє користувачам ефективно керувати своїми картками для запам'ятовування інформації.

Функції для входу та виходу користувача є важливими частинами процесу аутентифікації в будь-якому веб-додатку. Вони дозволяють користувачам безпечно увійти в систему, зберігаючи їх сесії та забезпечуючи безпечний вихід. Веб-додаток може використовувати ці функції для перевірки даних користувача і керування доступом до інших ресурсів, таких як картки, навчальні матеріали або персоналізовані дані.

Функція для входу зазвичай передбачає перевірку введених користувачем даних (наприклад, логін і пароль), порівняння їх з даними в базі даних та створення сесії для збереження статусу користувача. Користувач не повинен повторно вводити свої дані під час використання додатку, доки його сесія активна.

Ось приклад функції для входу в систему:

```
def load_config():
    app.config.update(dict(
        DATABASE=os.path.join(app.root_path, pathDB, nameDB),
        SECRET_KEY='development key',
        USERNAME='admin',
        PASSWORD='default'
    ))
    app.config.from_envvar('CARDS_SETTINGS', silent=True)
```

Якщо запит є методом POST (тобто коли користувач надсилає форму), функція перевіряє, чи існує користувач з таким логіном у базі даних.

Якщо користувач знайдений, порівнюється хеш пароля з тим, що зберігається в базі даних. Для порівняння використовується функція

check_password_hash з бібліотеки werkzeug.security, яка гарантує, що пароль зберігається безпечно (як хеш, а не в відкритому вигляді).

Якщо пароль правильний, створюється сесія (створюється ключ logged_in), і користувач перенаправляється на головну сторінку.

Якщо дані неправильні, користувач отримує повідомлення про помилку.

Якщо користувач уже увійшов, його сесія зберігається, і він може безперешкодно переглядати інші сторінки без необхідності знову вводити пароль.

Функція для виходу з системи передбачає завершення сесії користувача та перенаправлення на сторінку входу або головну сторінку.

Приклад функції для виходу:

```
@app.route('/logout')
def logout():
    session.pop('logged_in', None)
    session.pop('user_id', None)
    flash('You have been logged out.')
    return redirect(url_for('login'))
```

Ці функції дозволяють користувачам легко входити в систему для доступу до персоналізованих функцій, а також забезпечують безпеку, дозволяючи їм вийти з додатку, коли це необхідно. Всі ці операції виконуються через веб-форму, що є зручним для користувачів.

У моєму веб-додатку реалізована система тегів для карток. Це дозволяє користувачам додавати, редагувати та фільтрувати картки за допомогою тегів. Важливі моменти в реалізації цієї функції можна описати так:

Ця функція getAllTag() дозволяє отримати всі теги з бази даних. Вона виконує SQL-запит для вибору всіх тегів із таблиці tags, після чого ці теги

повертаються для використання на сторінці, наприклад, для фільтрації карток або відображення всіх доступних тегів.

```
def getAllTag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, tagName
        FROM tags
        ORDER BY id ASC
    '''
    cur = db.execute(query)
    tags = cur.fetchall()
    return tags
```

У додатку є можливість додавати нові теги через форму. Коли користувач надсилає форму, введене значення записується в таблицю tags у базі даних. Це дозволяє користувачам створювати нові категорії для карток.

```
@app.route('/addTag', methods=['POST'])
def add_tag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               [request.form['tagName']])
    db.commit()
    flash('New tag was successfully added.')
    return redirect(url_for('tags'))
```

Для редагування існуючих тегів є дві функції: edit_tag і update_tag. Коли користувач хоче змінити назву тегу, він перенаправляється на сторінку редагування, де йому надається можливість змінити текст тегу. Після цього, при підтвердженні змін, виконується SQL-запит на оновлення значення тегу в базі даних.

```
@app.route('/editTag/<tag_id>')
def edit_tag(tag_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    tag = getTag(tag_id)
    return render_template('editTag.html', tag=tag)

@app.route('/updateTag', methods=['POST'])
def update_tag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    command = '''
        UPDATE tags
        SET
          tagName = ?
        WHERE id = ?
    '''
    db.execute(command,
               [request.form['tagName'],
                request.form['tag_id']
                ])
    db.commit()
    flash('Tag saved.')
```

```
    return redirect(url_for('tags'))
```

Веб-додаток дозволяє фільтрувати картки за тегами. Наприклад, ви можете фільтрувати картки за певним типом (якщо тег пов'язаний з типом картки). Для цього використовується SQL-запит, де картки вибираються на основі їх типу або іншого критерію. Функції для фільтрації карток, такі як filter_cards, мають можливість працювати з тегами при побудові запиту.

```
@app.route('/filter_cards/<filter_name>')
def filter_cards(filter_name):
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    filters = {
        "all":      "where 1 = 1",
        "general":  "where type = 1",
        "code":     "where type = 2",
        "known":    "where known = 1",
        "unknown":  "where known = 0",
    }

    query = filters.get(filter_name)
    if(query is None):
        query = "where type = {0}".format(filter_name)
        filter_name = int(filter_name)

    if not query:
        return redirect(url_for('show'))

    db = get_db()
```

```
    fullquery = "SELECT id, type, front, back, known FROM cards " + \
      query + " ORDER BY id DESC"
    cur = db.execute(fullquery)
    cards = cur.fetchall()
    tags = getAllTag()
    return render_template('show.html', cards=cards, tags=tags, filter_name=filter_name)
```

Функція bookmark дозволяє додавати теги до картки. Користувач може вибрати картку та прив'язати до неї певний тег (тип), змінюючи таким чином категорію картки. Це корисно, якщо картки мають різні категорії, такі як "код", "загальні", "закріплені" тощо.

```
@app.route('/bookmark/<card_type>/<card_id>')
def bookmark(card_type, card_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('UPDATE cards SET type = ? WHERE id = ?',[card_type,card_id])
    db.commit()
    flash('Card saved.')
    return redirect(url_for('memorize', card_type=card_type))
```

Коли ви створюється нова база даних (використовуючи функцію init_db), ініціалізується стандартні теги (наприклад, "general", "code", "bookmark"). Це необхідно для того, щоб користувачі могли працювати з певними тегами з самого початку, не створюючи їх вручну.

```
def init_tag():
```

```
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["general"])
    db.commit()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["code"])
    db.commit()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["bookmark"])
    db.commit()
```

Таким чином, система тегів в додатку забезпечує зручний спосіб організації карток за категоріями та надає можливість користувачам додавати, редагувати та фільтрувати картки за допомогою тегів.

Процес запам'ятовування карток у коді реалізований через функцію, яка допомагає користувачеві вивчати картки за допомогою різних методів, таких як фільтрація за тегами, перевірка вже відомих або невідомих карток, а також організація процесу запам'ятовування через кілька етапів.

Основна функція для запам'ятовування — це memorize, яка визначає, які картки показувати користувачу в залежності від типу фільтрації або вибору карток. Це дозволяє відображати картки для вивчення в залежності від їх статусу (наприклад, невідомі картки або всі картки).

```
@app.route('/memorize')
@app.route('/memorize/<card_type>')
def memorize(card_type=None):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
```

```
    db = get_db()
    filter_type = card_type or 'all'
    # Побудова SQL-запиту на основі типу картки
    fullquery = "SELECT id, front, back FROM cards WHERE type = ?"
    cur = db.execute(fullquery, [filter_type])
    cards = cur.fetchall()
    tags = getAllTag()
    return render_template('memorize.html', cards=cards, filter_type=filter_type, tags=tags)
```

Якщо не передано конкретний тип картки (card_type), за замовчуванням будуть вибрані всі картки.

Запит вибирає картки з бази даних відповідно до цього типу, після чого картки передаються на сторінку для відображення.

Перевірка, чи картка відома (known)**:** Картки мають атрибут known, який вказує, чи вже запам'ятав користувач цю картку. Це може бути використано для фільтрації карток, які користувач повинен повторно переглянути.

```
@app.route('/memorize/known')
def memorize_known():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    db = get_db()
    query = "SELECT id, front, back FROM cards WHERE known = 1"
    cur = db.execute(query)
    cards = cur.fetchall()
    return render_template('memorize.html', cards=cards, filter_type='known')
```

Це дозволяє користувачу переглядати тільки ті картки, які він уже знає (known = 1), щоб він міг вибрати картки для подальшого повторення.

Процес додавання картки в пам'ять (bookmarking): Дана функція bookmark дозволяє користувачам відзначати картки, які вони хочуть зберегти як відомі або закріплені для подальшого повторення. Користувач може змінити статус картки на "відома", щоб система знала, що цей етап вивчення пройдено.

```
@app.route('/bookmark/<card_type>/<card_id>')
def bookmark(card_type, card_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('UPDATE cards SET type = ? WHERE id = ?',[card_type, card_id])
    db.commit()
    flash('Card saved.')
    return redirect(url_for('memorize', card_type=card_type))
```

Тут відбувається зміна типу картки в базі даних, щоб позначити її як відому чи закріплену. Це також сприяє очищенню картки зі списку карток, які потрібно повторювати.

Система підтримує кілька типів фільтрів для карток, зокрема за статусом, типом і рівнем складності. Наприклад, ви можете фільтрувати картки за тегами або їх статусом (відома/не відома).

Приклад запиту для фільтрації карток:

```
@app.route('/filter_cards/<filter_name>')
def filter_cards(filter_name):
    if not session.get('logged_in'):
```

```
        return redirect(url_for('login'))

    filters = {
        "all":      "where 1 = 1",
        "general":  "where type = 1",
        "code":     "where type = 2",
        "known":    "where known = 1",
        "unknown":  "where known = 0",
    }
    query = filters.get(filter_name)
    if(query is None):
        query = "where type = {0}".format(filter_name)
        filter_name = int(filter_name)

    if not query:
        return redirect(url_for('show'))

    db = get_db()
    fullquery = "SELECT id, type, front, back, known FROM cards " + \
        query + " ORDER BY id DESC"
    cur = db.execute(fullquery)
    cards = cur.fetchall()
    tags = getAllTag()
    return render_template('show.html', cards=cards, tags=tags,
filter_name=filter_name)
```

Ця функція дозволяє відображати картки з різними статусами, наприклад, "відома" або "невідома". Фільтрація допомагає користувачам сфокусуватися на картках, які ще потребують повторення, або картках, які вже вивчені.

Оскільки система використовує типи карток (наприклад, "general", "code", "bookmark"), цей механізм дозволяє адаптувати запам'ятовування до потреб користувача, зберігаючи баланс між повторенням карток і вивченням нових матеріалів. Статус карток змінюється в залежності від того, як користувач взаємодіє з ними (наприклад, закріплює картку або відзначає її як відому).

3.5 Тестування та аналіз результатів роботи програми для запам’ятовування інформації

Тестування та аналіз роботи програми для запам’ятовування інформації можна здійснювати на кількох рівнях, перевіряючи коректність роботи основних функцій, а також оцінюючи ефективність системи з точки зору зручності користування та досягнення цільових результатів.

3.5.1 Тестування роботи програми

Тестування створення нових карток з різним вмістом та іншими форматами даних. Необхідно перевірити, чи правильно зберігаються та відображаються картки, чи працює функція призначення тегів і типів.

Для початку розглянемо форму створення картки для запам'ятовування. Ця форма є ключовим інструментом для організації навчального матеріалу, що дозволяє створювати зручні для використання картки. У формі передбачено кілька полів для введення даних, таких як основний текст або питання, відповідь, а також додаткові примітки чи пояснення. Окрім цього, передбачена можливість вибору колекції, до якої буде належати ця картка. Така функціональність дозволяє структурувати інформацію за темами чи предметами, забезпечуючи швидкий доступ до потрібного матеріалу під час повторення. Форма забезпечує інтуїтивно зрозумілий інтерфейс для полегшення процесу створення і збереження карток, що робить її універсальним інструментом для

запам'ятовування будь-якої інформації. Форма створення карток представлена на рисунку 3.2.

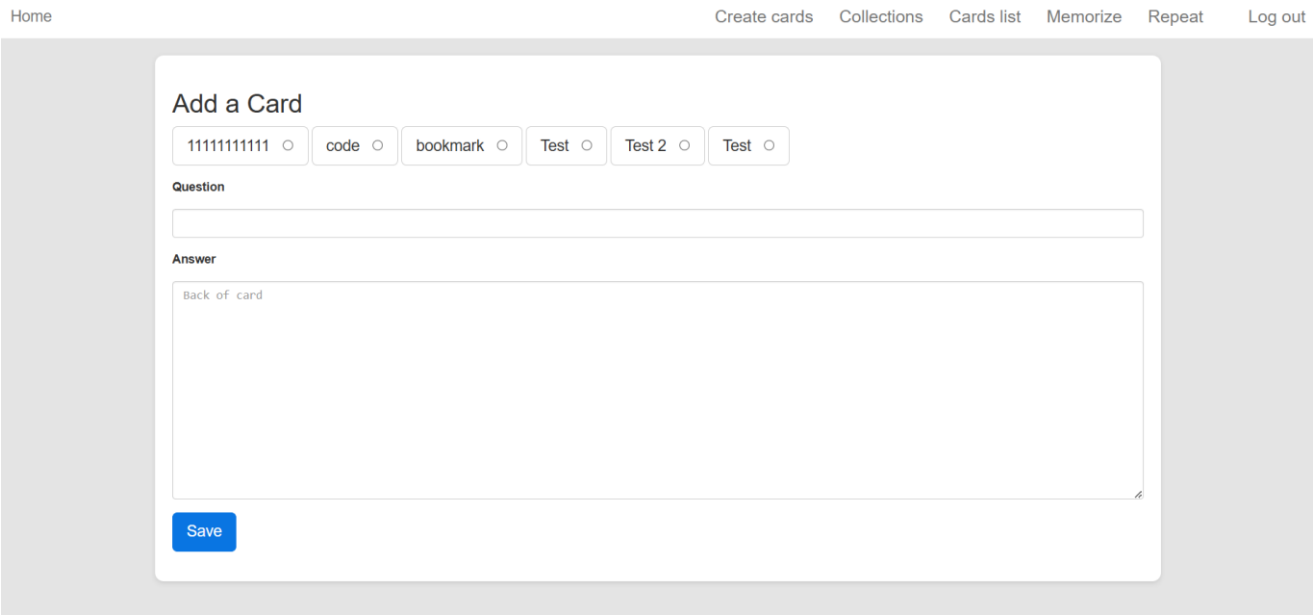

Рисунок 3.2 – Форма створення картки

Тепер спробуємо ввести текст у доступні поля, щоб перевірити роботу форми. Почнемо з основного тексту або питання, що буде основою для картки. Потім введемо відповідь, яка стане ключовим елементом для запам'ятовування. За потреби можна додати додаткові примітки чи пояснення, які допоможуть краще зрозуміти контекст або уточнити деталі. На завершення оберемо відповідну колекцію, до якої належатиме картка, що дозволить віднести її до потрібної категорії для майбутнього використання. Такий підхід дозволяє перевірити, чи коректно працюють усі елементи форми, зокрема, чи зберігаються дані, і чи відображаються вони відповідно до вибраних параметрів. Під час тестування не було виявлено проблем чи не зручностей, і картка була успішно створена та додана до колекції.

Цей функціонал представлено на рисунку 3.3.

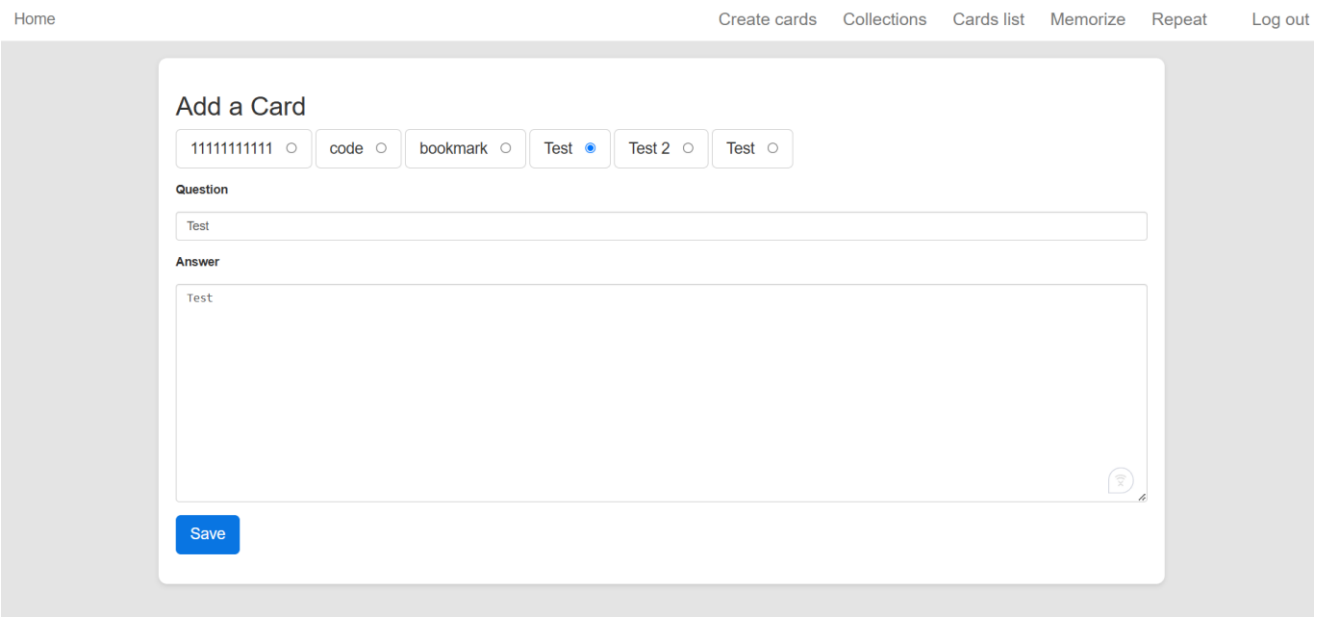

Рисунок 3.3 – Заповненні поля картки

Тепер розглянемо форму створення колекції представлену на рисунку 3.4. Ця форма дозволяє користувачеві організовувати картки для запам'ятовування за окремими категоріями, що значно спрощує навігацію та доступ до матеріалів. У формі є спеціальне поле для введення назви нової колекції представлену на рисунку 3.5. Після введення назви користувач натискає кнопку "Зберегти," яка ініціює додавання створеної колекції до загального списку.

Список колекцій представлений на рисунку 3.6. надає зручний огляд усіх доступних категорій, а поруч із кожною назвою передбачено кнопку редагування. Коли користувач натискає цю кнопку, відкривається форма редагування, яка дозволяє змінити назву існуючої колекції. Після внесення змін і збереження оновлена назва колекції автоматично замінює стару в списку. Така динамічність забезпечує зручність управління колекціями, дозволяючи користувачам швидко адаптувати систему до своїх потреб.

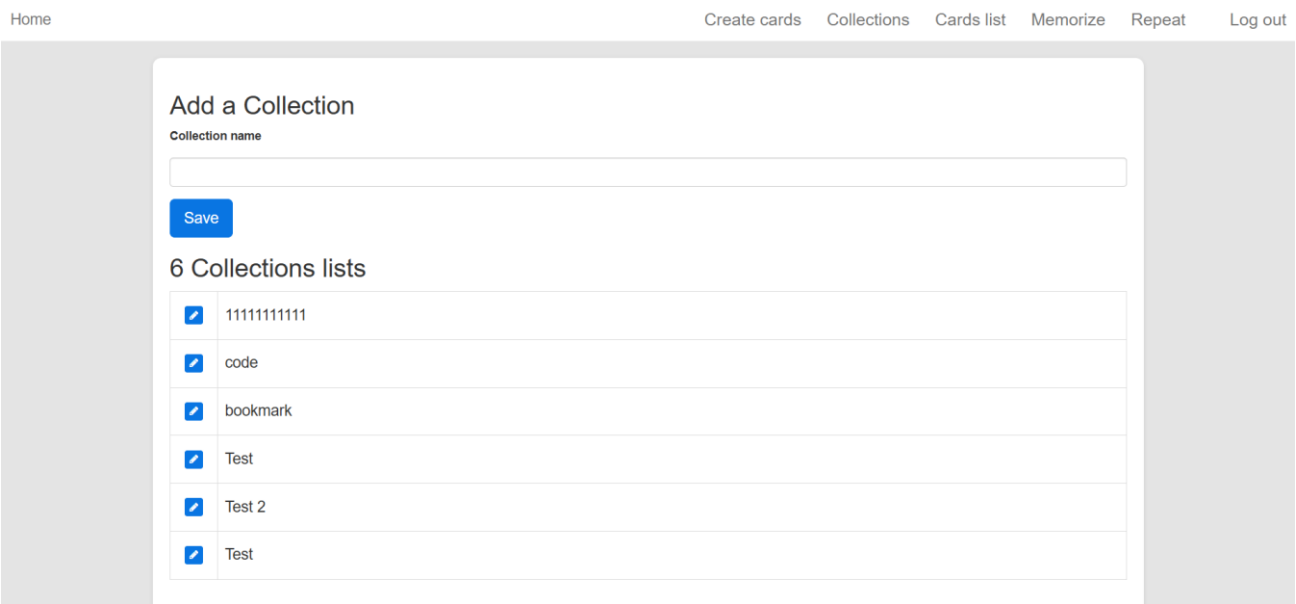

Рисунок 3.4 – Форма створення колекції

Форма створення колекції відкривається без затримок і працює стабільно. Користувач має можливість одразу розпочати введення даних для нової колекції, що робить процес створення максимально швидким і зручним. Така плавність у роботі форми підтверджує її надійність і добре оптимізовану функціональність.

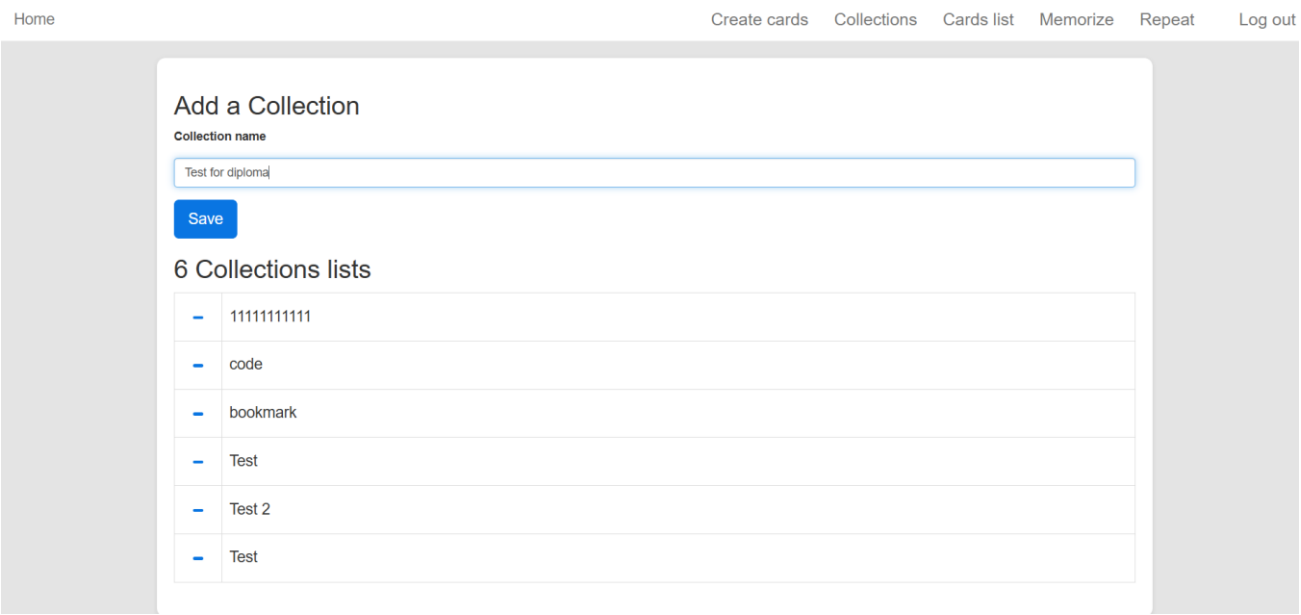

Рисунок 3.5 – Заповнення поля колекції

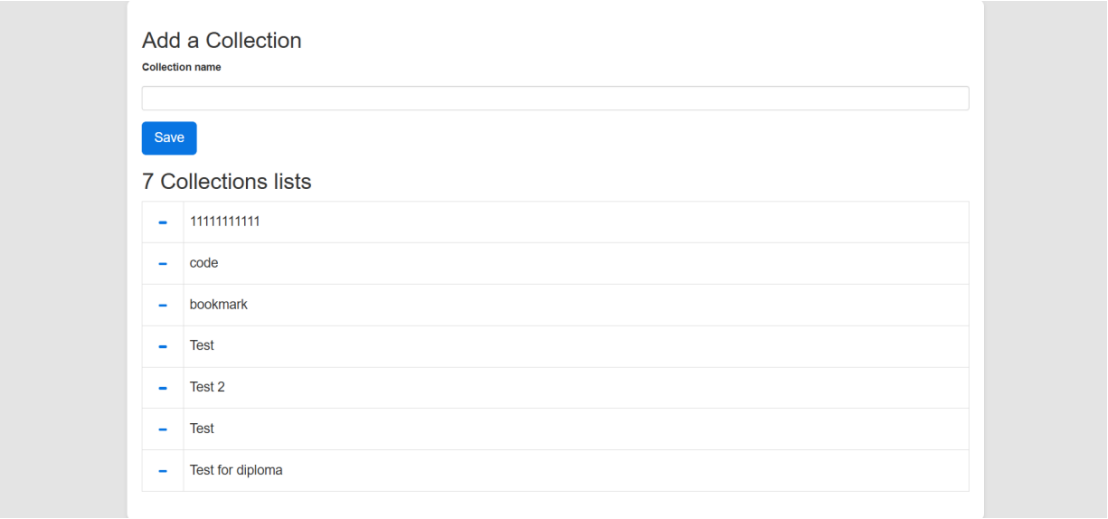

Рисунок 3.6 – Приклад створеної колекції

Форма створення колекції працює коректно і виконує всі необхідні функції. Вона дозволяє безперешкодно створювати нову колекцію, надаючи користувачеві можливість вводити бажану назву, а потім зберігати її. Після успішного збереження нова колекція миттєво відображається у списку разом з уже існуючими. Це підтверджує, що механізм додавання нових елементів до системи реалізований ефективно, забезпечуючи швидкий доступ до щойно створеної категорії.

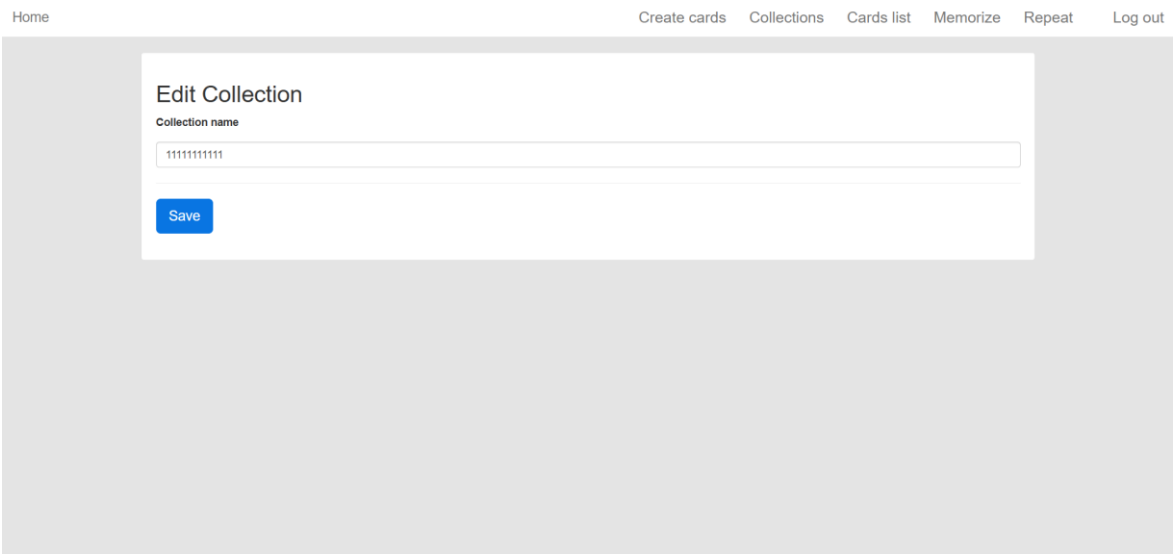

Рисунок 3.7 – Форма редагування колекції

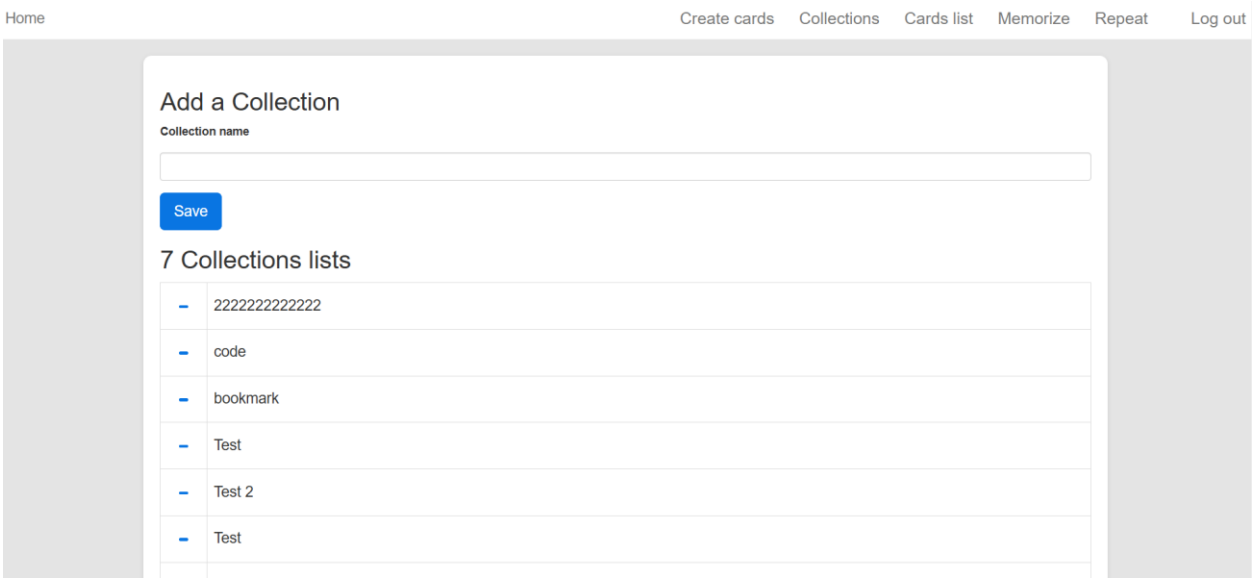

Рисунок 3.8 – Відредагована колекція

Форма редагування колекції функціонує належним чином. Після внесення змін у назву колекції та її збереження оновлення відбувається коректно: стара назва замінюється новою у списку колекцій. Це свідчить про стабільність роботи форми та правильну реалізацію механізму редагування

На сторінці з переліком усіх карток представлений зручний список, який відображає всі створені картки. Користувач має змогу легко орієнтуватися серед карток завдяки сортуванню за колекцією, що дозволяє швидко знаходити потрібні матеріали в рамках певної категорії. Окрім цього, передбачена можливість відкривати форму редагування картки. Ця функція дозволяє вносити зміни до тексту картки чи інших даних, забезпечуючи актуальність інформації. Загалом, сторінка забезпечує користувачеві ефективний інструмент для роботи з великою кількістю карток.

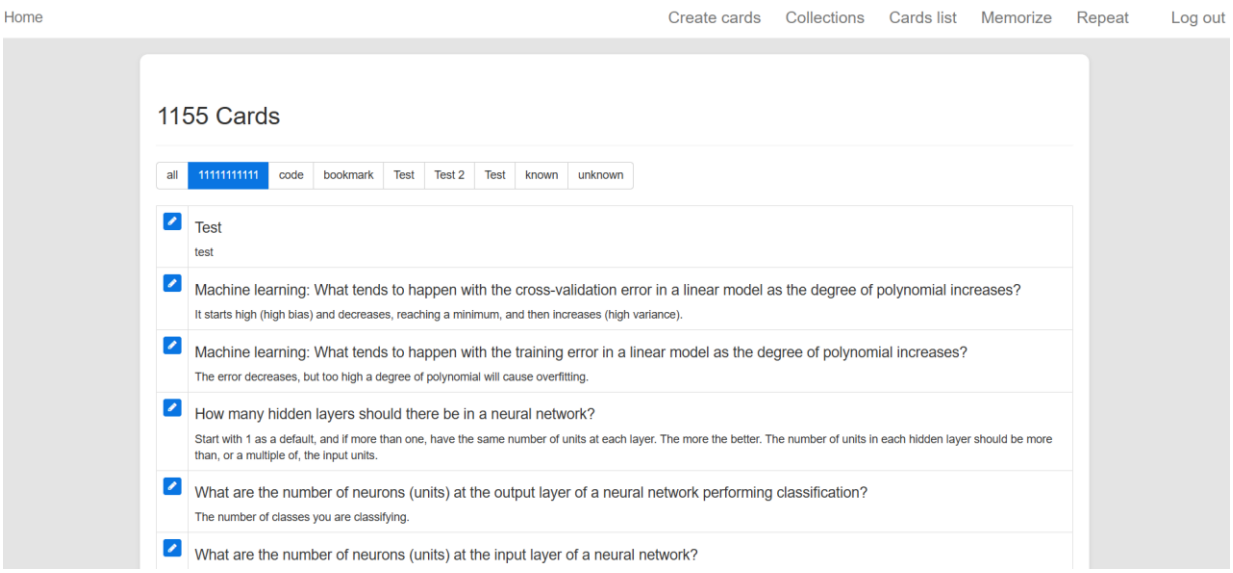

Рисунок 3.9 – Сторінка з переліком карток та колекцій

Усі елементи в таблиці відображаються коректно, що забезпечує повноту та точність інформації про картки. Функція сортування за колекціями також працює бездоганно, дозволяючи швидко впорядковувати картки відповідно до вибраної категорії. Це значно спрощує навігацію та пошук потрібних матеріалів, роблячи інтерфейс інтуїтивно зрозумілим і зручним для користувача.

Після натискання на форму редагування картки користувач перенаправляється на окрему сторінку, призначену для внесення змін. На цій сторінці передбачені поля для редагування тексту питання та відповіді, що дозволяє оновити чи уточнити інформацію на картці. Окрім цього, користувач може змінити колекцію, до якої належить картка, обравши іншу зі списку доступних. Такий підхід забезпечує гнучкість у редагуванні та дозволяє легко адаптувати картки до актуальних потреб, підтримуючи їхню організованість і релевантність.

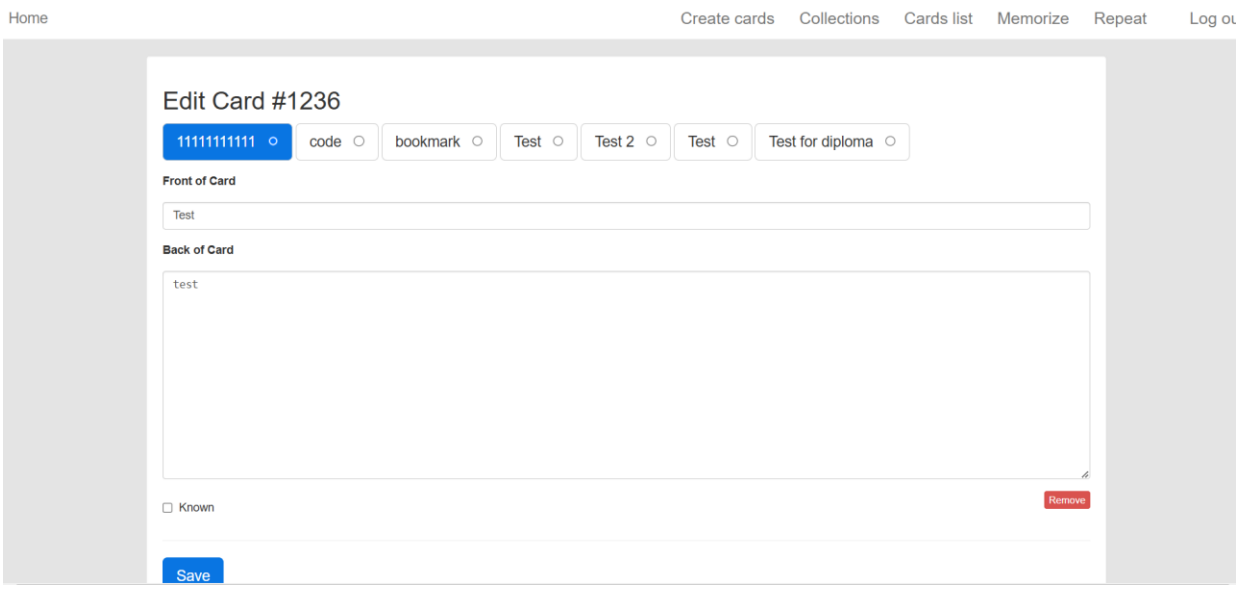

Рисунок 3.10 – Форма редагування картки

Форма редагування картки працює коректно. Після внесення змін до тексту питання, відповіді або вибору іншої колекції, оновлена картка зберігається належним чином. У загальному списку відображається вже змінена версія картки, що підтверджує правильну роботу механізму оновлення даних.

Тепер розглянемо процес запам'ятовування, реалізований на сторінці "Memorize". Тут представлено основний інструмент для роботи з картками. Користувач має можливість обрати колекцію, з якої він бажає почати запам'ятовування, що дозволяє зосередитися на конкретній темі чи категорії.

На екрані відображається випадкове питання з обраної колекції. Користувач читає питання, формулює відповідь подумки або вголос, а потім перевертає картку, щоб перевірити правильність своєї відповіді. Такий підхід сприяє активному залученню пам'яті та стимулює ефективне засвоєння інформації.

Якщо користувач упевнений, що запам'ятав зміст картки, він позначає її як "вивчену". Після цього система автоматично переходить до наступної картки, дозволяючи користувачеві продовжувати процес без перерв. Така механіка створює зручний і послідовний досвід навчання, що відповідає принципам активного повторення.

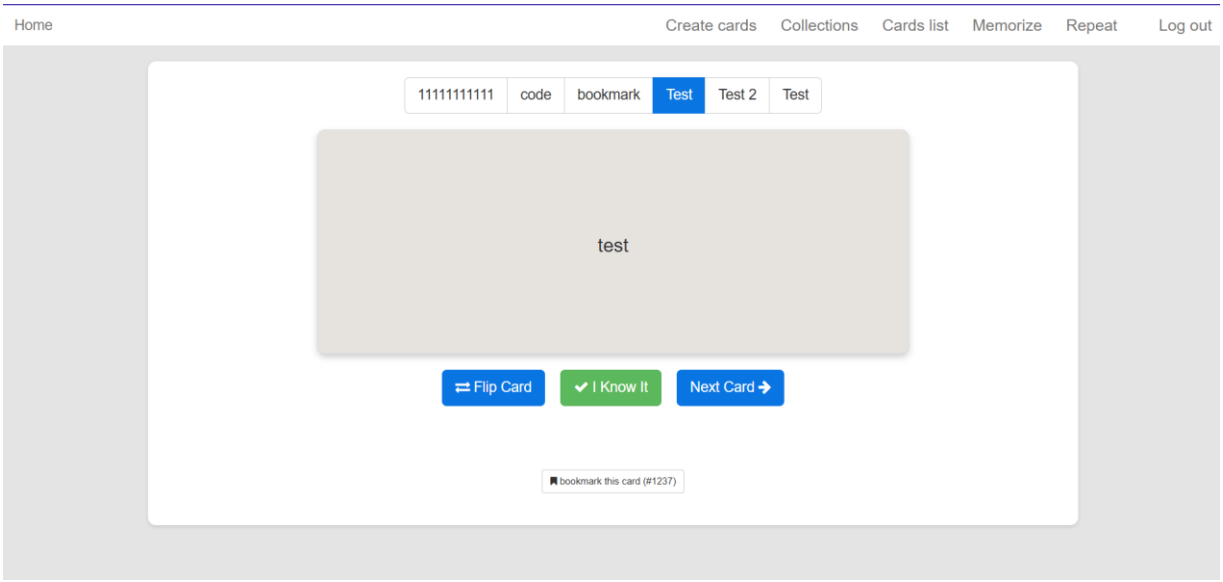

Рисунок 3.11 – Передня сторона картки

Передня сторона картки відображається коректно, забезпечуючи чіткий і зрозумілий вигляд питання або завдання для користувача. Усі кнопки на сторінці працюють належним чином: функція перевертання картки дозволяє швидко переглянути відповідь, а кнопки для позначення картки як "вивчена" або переходу до наступної картки виконують свої завдання без жодних проблем.

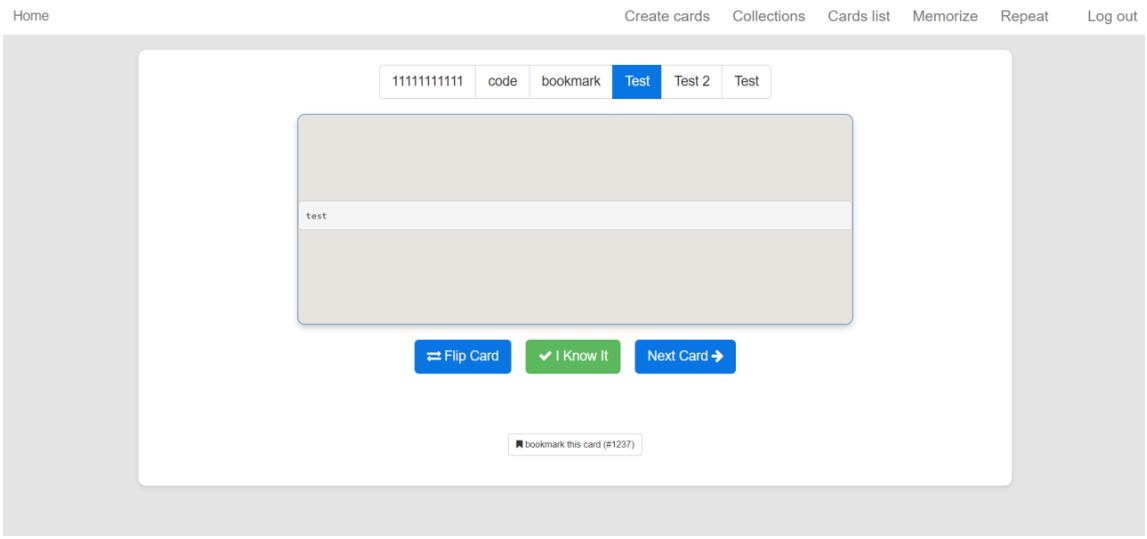

Рисунок 3.12 – Задня сторона картки

Задня сторона картки відображається коректно після натискання на відповідну кнопку, що дозволяє користувачеві без проблем переглянути відповідь або пояснення. Усі елементи функціоналу, включно з кнопками позначення картки як "вивчена" та переходу до наступної картки, залишаються доступними і працюють належним чином. Така стабільність у роботі забезпечує безперервність процесу запам'ятовування та комфорт для користувача.

На сторінці "Repeat" користувач може повторити всі картки, які він позначив як "вивчені". Функціонал цієї сторінки схожий на той, що реалізований на сторінці "Memorize", де користувач бачить випадкове питання з картки, відповідає на нього, перевертає картку, щоб перевірити правильність відповіді, і позначає картку як "знану". Однак на сторінці "Repeat" є додаткова можливість — якщо користувач забув відповідь або не впевнений у своїх знаннях, він може позначити картку як "забуту". Це дозволяє повернути картку до процесу повторення для подальшого запам'ятовування, що сприяє кращому засвоєнню матеріалу. Такий підхід забезпечує більш динамічне і персоналізоване навчання, дозволяючи користувачеві зосереджуватись на тих картках, які потребують додаткового уваги.

На сторінці "Repeat" користувач також має можливість встановити нагадування, щоб визначити, як часто він хоче повторювати вивчений матеріал. Це дозволяє налаштувати інтервал повторення карток, що сприяє ефективному запам'ятовуванню і постійному оновленню знань. Користувач може вибрати частоту повторення, що найкраще відповідає його ритму навчання, зокрема встановити нагадування на певні дні або інтервали часу. Ця функціональність допомагає підтримувати інформацію в пам'яті, запобігаючи забуванню матеріалу і забезпечуючи більш ефективне навчання.

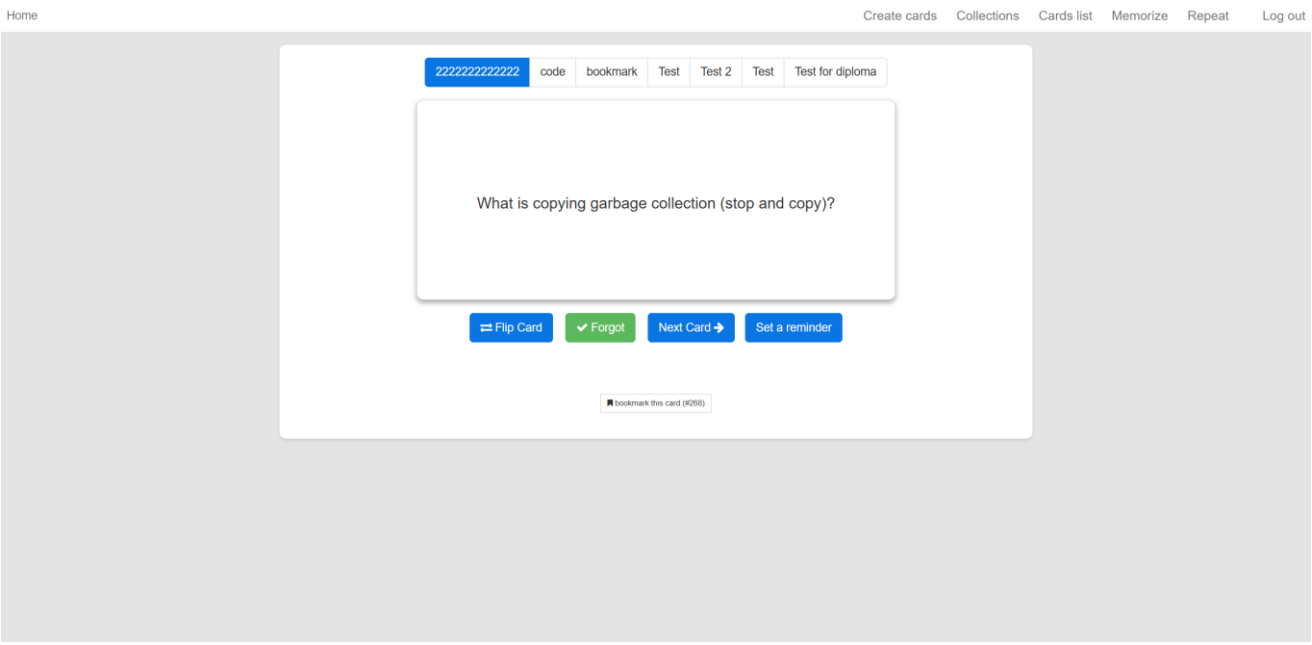

Рисунок 3.13 – Передня сторона картки для повторення

Після натискання на кнопку нагадування відкривається форма, яка дозволяє користувачеві налаштувати інтервал повторення вивченого матеріалу. У цій формі користувач може вибрати, як часто він хоче повторювати картки, наприклад, щодня, щотижня або через певні дні. Це надає гнучкість у налаштуванні інтервалів, що дозволяє адаптувати систему до індивідуальних потреб користувача. Після встановлення бажаного інтервалу, користувач може зберегти налаштування, і система буде автоматично нагадувати про необхідність повторення карток у визначені моменти.

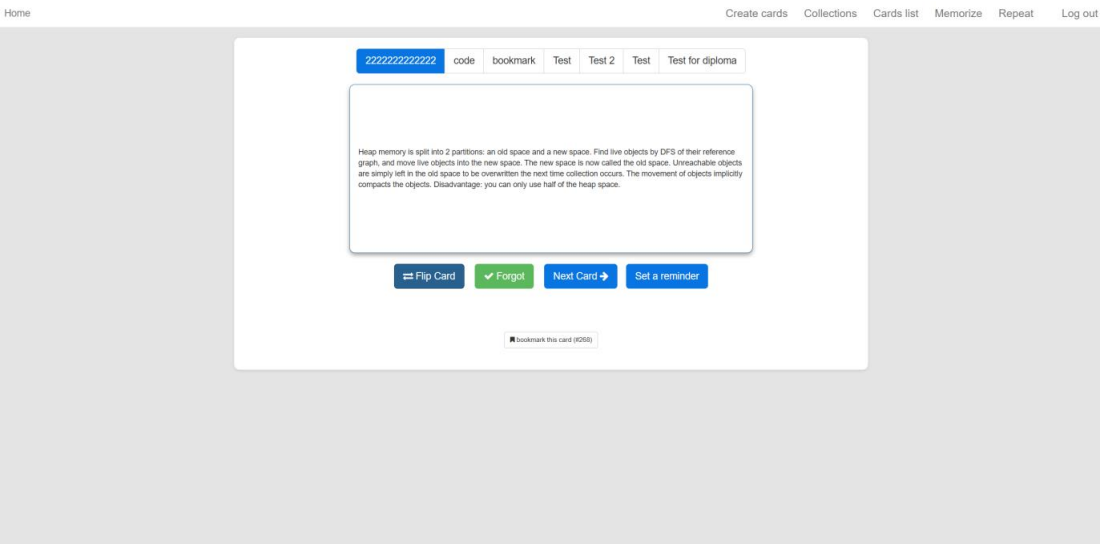

Рисунок 3.14 – Задня сторона картки для повторення

Після проведення тестування на сторінці "Repeat" ми впевнилися, що весь функціонал працює правильно. Користувач може ефективно повторювати картки, позначати їх як "забуті", і всі кнопки та інтерфейс працюють коректно. Процес повторення карток відбувається без проблем, і система належно реагує на зміни статусу карток. Це підтверджує, що функціональність сторінки реалізована стабільно і забезпечує ефективне повторення матеріалу для користувача.

3.5.2 Аналіз ефективності роботи

Програма для запам’ятовування інформації, яка базується на використанні флеш-карток та функції нагадувань, демонструє високу ефективність у покращенні довготривалого збереження знань. Особливістю програми є можливість налаштування нагадувань, які допомагають користувачам періодично повторювати матеріал, що сприяє укріпленню знань у довготривалій пам’яті.

Ефективність програми була перевірена шляхом експерименту за участі двох груп людей. Перша група вивчала матеріал лише один раз, використовуючи флеш-картки без подальших повторень. Друга група користувалася функцією

нагадувань і регулярно повторювала матеріал у встановлені інтервали. Результати експерименту показали, що учасники другої групи значно краще запам’ятали інформацію. Це підтверджує дієвість функції нагадувань у покращенні процесу запам’ятовування завдяки ефекту повторення, відомому як ефект інтервального повторення.

Таким чином, програма не лише допомагає у вивченні нового матеріалу, але й забезпечує ефективне його закріплення, використовуючи науково обґрунтовані методи оптимізації процесу навчання.

Таблиця 3.2 – Порівняльна характеристика розробленого ПЗ

	Anki	Quizlet	Memrise	Розроблена інформаційна технологія
Безкоштовний доступ до функціоналу	+	-	-	+
Можливість створювати власні колекції питань	+	+	-	+
Крос-платформеність	-	+	-	+
Можливість роботи офлайн	-	-	-	+
Можливість ставити нагадування для інтервального повторювання	-	-	-	+

3.6 Висновок до розділу 3

В результаті роботи над проектом було реалізовано програмне забезпечення для запам'ятовування інформації за допомогою інтервального повторення карток. Для розробки програми була обрана мова програмування

Python, що забезпечує зручність та гнучкість у створенні додатків завдяки широкому вибору бібліотек та фреймворків.

Програмна реалізація включала створення інтерфейсу користувача для додавання та організації карток, а також забезпечення функціональності для їх повторення на основі статусу запам'ятовування. Вся система спрямована на підвищення ефективності навчання завдяки методам інтервального повторення.

Було обрано SQLite через її простоту у використанні, можливість роботи без необхідності встановлення серверної частини, а також підтримку транзакцій та можливість інтеграції з іншими компонентами програми.

SQLite забезпечує надійне зберігання даних, що дозволяє організувати картки та колекції у вигляді таблиць, де кожна картка має свої атрибути, такі як питання, відповідь, теги, колекція тощо. Для кожної картки створюється окрема запис у базі даних, що дозволяє швидко здійснювати пошук і сортування карток за різними параметрами. Використання SQLite також дозволяє зручно оновлювати та редагувати існуючі картки без великих витрат ресурсів.

Тестування програми підтвердило її працездатність, виявивши і виправивши кілька незначних помилок. Після внесення коректівів система демонструє стабільну роботу та відповідає вимогам до продуктивності і зручності використання.

Також було проведено тестування функціональних можливостей програми, яке продемонструвало високий рівень її результативності. Учасники, які використовували функцію нагадувань для повторення матеріалу, показали значно вищі результати у тестах на запам’ятовування, ніж ті, хто вивчав інформацію лише один раз.

Таким чином, створена програма є ефективним інструментом для запам'ятовування інформації, що може бути корисним у навчанні та особистій продуктивності.

4 ЕКОНОМІЧНА ЧАСТИНА

Успішне впровадження науково-технічної розробки можливе за умови, що вона відповідає актуальним вимогам науково-технічного прогресу та враховує економічні аспекти. Важливим етапом є оцінка економічної ефективності результатів науково-дослідної роботи.

Магістерська кваліфікаційна робота, присвячена розробці та дослідженню на тему «Інформаційна технологія для запам’ятовування інформації», належить до категорії науково-технічних проектів, орієнтованих на комерціалізацію. Рішення щодо виходу розробки на ринок може бути ухвалене ще в процесі її виконання, що відкриває можливість реалізації отриманих результатів у реальних умовах.[21]

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу інформаційної технології для запам’ятовування інформації.

Для проведення технологічного аудиту використано таблицю 4.1 в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах

Таблиця 4.1 – Продовження

Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Таблиця 4.1 – Продовження

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використо-вуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використо-вуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведені до таблиці 4.3. Експертами в опитуванні були: Бойко Тетяна – людина з досвідом в викладанні, Щупак Денис – JavaScript developer, Павліченко Юрій – Python developer .

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Експерти		
	Бойко Тетяна	Щупак Денис	Павліченко Юрій
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	4	2
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	4	2	1
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	4	4	3
8. Практична здійсненність (наявність фахівців)	3	2	3
9. Практична здійсненність (наявність фінансів)	2	2	1
10. Практична здійсненність (необхідність нових матеріалів)	2	2	3
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	4	4	3
Сума балів	СБ1=39	СБ2=35	СБ3=33
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{39+35+33}{3} = 35$		

За результатами розрахунків, наведених в таблиці 4.3, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 4.2.

Згідно проведених досліджень рівень комерційного потенціалу розробки, розрахований на основі висновків експертів склав 35 балів, що, відповідно до

таблиці 4.2, свідчить про рівень комерційного потенціалу розробки вище середнього.

На початковому етапі розробка може бути реалізована шляхом розміщення програмного файлу в репозиторії GitHub із подальшим поширенням посилання та опису через професійні соціальні мережі, наприклад, LinkedIn. Після отримання перших відгуків від користувачів, рішення буде вдосконалено і розміщено на спеціалізованому сервері. Детальний опис розробки та доступ до сервера буде представлено у вигляді статті, яку можна опублікувати на таких спеціалізованих платформах, як DOU.

Основною аудиторією інформаційної технології для запам’ятовування інформації є студенти, викладачі, учні шкіл, професіонали, які потребують швидкого освоєння нового матеріалу, а також люди, які прагнуть покращити свої когнітивні здібності. Також до цільової групи можуть належати компанії, що займаються навчанням персоналу, та окремі особи, які займаються саморозвитком і підготовкою до іспитів або сертифікацій.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія для запам’ятовування інформації», під час планування, обліку і обчислення собівартості науково-дослідної роботи групуються за певними статтями.

4.2.1 Витрати на оплату праці

Основна заробітна плата кожного із дослідників $З_0$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_0 = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки програмні засоби необхідно залучити програміста з посадовим окладом 15000 грн. Кількість робочих днів у місяці складає 40, а кількість робочих днів програміста складає 22. Зведемо сумарні розрахунки до таблиця 4.4.

Таблиця 4.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	20000	909,1	25	23809
Програмний інженер	15000	681,8	40	28571
Всього				31817

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт НДР розраховуємо за формулою 4.2:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \qquad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою 4.3:

$$C_i = \frac{M_m \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.3)$$

де M_m – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо M_m = 8000 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати;

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$$C_i = \frac{8000 \cdot 1 \cdot 1{,}65}{21 \cdot 8} = 78{,}6 \text{ } грн.$$

$$З_{p1} = 78{,}6 \cdot 2 = 157{,}2 \text{ } грн.$$

Таблиця 4.5 – Величина витрат на основну заробітню плату робітників

Найменування робіт	Тривалість, год.	Розряд роботи	Погодинна тарифна ставка, грн.	Величина оплати на робітника, грн.
Збір вимог	6	1	50,3	301,8
Проектування	14	3	90,1	1261,4
Розробка	20	5	120,6	2412
Тестування	4	2	75,3	301,2
Реалізація	12	4	60,5	726
Всього				5002,4

Додаткова заробітна плата дослідників та робітників. Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою 4.4:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

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

$$З_{дод} = (31817 + 5002{,}4) \cdot \frac{11}{100\%} = 4050{,}1 \text{ грн.}$$

4.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 4.5:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймемо 22%.

$$З_н = (31817 + 5002{,}4 + 4050{,}1) \cdot \frac{22}{100\%} = 8991{,}2 \text{ грн.}$$

4.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на

проведення досліджень за темою "Інформаційна технологія для запам’ятовування інформації". Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 4.6:

$$\text{М} = \sum_{j=1}^{n} H_j \cdot \text{Ц}_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot \text{Ц}_{\text{в}j}, \quad (4.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

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

K_j– коефіцієнт транспортних витрат, ($K_j = 1{,}1 \ldots 1{,}15$);

B_j – маса відходів j-го найменування, кг;

$\text{Ц}_{\text{в}j}$ – вартість відходів j-го найменування, грн/кг.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 шт, грн.	Норма витрат, шт.	Вартість витраченого матеріалу, грн.
Пачка паперу А4	170	1	170
Ручка	45	2	80
ОЗП Kingston DDR4 32GB	1899	1	1899
Флеш пам'ять USB Kingston DataTraveler	400	1	400
Всього			2549
З врахуванням коефіцієнта транспортування			2700,8

4.2.4 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 4.5:

$$\text{А}_{\text{обл}} = \frac{\text{Ц}_{\text{б}}}{T_{\text{в}}} \cdot \frac{t_{\text{вик}}}{12}, \quad (4.7)$$

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

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

K_c – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$$А_{обл} = \frac{35000}{2} \cdot \frac{2}{12} = 2916{,}6 \text{ грн.}$$

Таблиця 4.7 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Ноутбук	40000	2	2	3333,3
Монітор	8000	2	1	666,6
Приміщення	250000	20	1	2083,3
Всього				4208,2

4.2.5 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію $В_е$ розраховуємо за формулою 4.8:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot K_{впі}}{n_i}, \quad (4.8)$$

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

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 10 грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі}$ < 1;

n_i – коефіцієнт корисної дії обладнання, n_i < 1.

$$В_е = \frac{0{,}25 \cdot 250{,}0 \cdot 10 \cdot 0{,}5}{0{,}8} = 390{,}62 \text{ грн.}$$

4.2.6 Службові відрядження

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень. Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 4.9:

$$З_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.9)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження». Приймемо 20%.

$$З_{св} = (31817 + 5002{,}4) \cdot \frac{20}{100\%} = 7363{,}8 \text{ грн.}$$

4.2.7 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 4.10:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.10)$$

де $І_в$ – норма нарахування за статтею «Інші витрати». Приймемо 50%.

$$І_в = (31817 + 5002{,}4) \cdot \frac{50}{100\%} = 18409{,}7 \text{ грн.}$$

4.2.8 Накладні (загальновиробничі) витрати

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

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 4.11:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.11)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати». Приймемо 100%.

$$Н_{нзв} = (31817 + 5002{,}4) \cdot \frac{100}{100\%} = 36819{,}4 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія для запам'ятовування інформації», розраховуємо як суму всіх попередніх статей витрат за формулою 4.12:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв} \quad (4.12)$$

$$В_{заг} = 31817 + 5002{,}4 + 4050{,}1 + 8991{,}2 + 2700{,}8 + 2916{,}6 + 4208{,}2 + 390{,}62 + 7363{,}8 + 18409{,}7 + 36819{,}4 = 122669.82 \text{ грн.}$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою 4.13:

$$ЗВ = \frac{В_{заг}}{n}, \quad (4.11)$$

де n – коефіцієнт, який характеризує етап (стадію) виконання науководослідної роботи. Приймемо 0,7.

$$ЗВ = \frac{122669.82}{0{,}7} = 175242{,}6 \textit{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

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

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи.

Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із трьох років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою 4.12:

$$\Delta П_i = \sum_1^n (\Delta Ц_o * N * Ц_o * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\upsilon}{100}\right), \quad (4.12)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ – ставка податку на прибуток. У 2024 році – 18%.

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (88 \cdot 3000 + 338 \cdot 1000) \cdot 0{,}83 \cdot 0{,}4 \cdot \left(1 - \frac{0{,}18}{100}\%\right)$$

$= 163888{,}48$ грн.

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = П2\ (88 \cdot 4000 + 338 \cdot (1000 + 4000)) \cdot 0{,}83 \cdot 0{,}4 \cdot \left(1 - \frac{0{,}18}{100}\%\right)$$

$= 531956{,}96$ грн.

$$\Delta П_3 = (88 \cdot 4000 + 338 \cdot (1000 + 4000 + 6000)) \cdot 0{,}83 \cdot 0{,}4 \cdot \left(1 - \frac{0{,}18}{100}\%\right) = 1084059{,}68 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{4.13}$$

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

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 282746,3 грн.

$$PV = 2 \cdot 175242{,}6 \ = 350485{,}2 \text{ грн.}$$

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

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \tag{4.14}$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 282746,3 грн;

$$\text{ПП} = \sum_1^T \frac{\Delta \text{П}_\text{і}}{(1+\tau)^t}\ , \tag{4.13}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$Е_{абс} = 175242,6 - 350485,2 \ = \ 425727,8 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \ , \qquad (4.14)$$

де $Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[з]{1 + \frac{425727,8}{350485,2}} - 1 = 0.4$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f \ , \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,14…0,2);

f – показник, що характеризує ризикованість вкладень, приймемо 0,25.

$$\tau_{min} = 0{,}1 + 0{,}25 = 0{,}35.$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}. \qquad (4.16)$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}4} = 2{,}5 \text{ (роки)}.$$

$\mathrm{T_{ок}} \leq 3$ років, що свідчить про комерційну привабливість науковотехнічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновок до розділу 4

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки на тему «Інформаційна технологія для запам’ятовування інформації» становить 35 балів, що свідчить про високий інтерес до цієї тематики на ринку. Такий показник перевищує середній рівень, підтверджуючи важливість проведення відповідних наукових досліджень. При оцінюванні конкурентоспроможності, узагальнений коефіцієнт свідчить, що розробка перевищує існуючі аналоги приблизно в 1,3 рази, що демонструє її інноваційність та перевагу над конкурентами.

Термін окупності становить 2,5 роки, що менше стандартного трирічного періоду, прийнятого для оцінки інвестиційної привабливості. Це свідчить про високу комерційну перспективу розробки, що може зацікавити потенційних інвесторів у фінансуванні впровадження цієї технології на ринок. Таким чином, можна зробити висновок про доцільність проведення науково-дослідної роботи на тему «Інформаційна технологія для запам’ятовування інформації» та перспективність її комерціалізації.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології для запам’ятовування інформації. Зокрема, було проведено аналіз предметної області, об’єкта дослідження та існуючих методів і технологій, що сприяють ефективному збереженню та відтворенню інформації в пам’яті.

Розроблено модель інформаційної технології, яка включає інтеграцію алгоритмів для організації, структуризації та візуалізації інформації, адаптованих до різних стилів навчання та індивідуальних потреб користувачів. Було визначено основні принципи роботи системи, включно з використанням методів активного повторення, асоціативного мислення та інтерактивних тестів.

Запропонована технологія базується на використанні сучасних алгоритмів для створення персоналізованих навчальних шляхів, які автоматично адаптуються до прогресу користувача. Особлива увага приділена гнучкості системи, що дозволяє інтегрувати мультимедійні ресурси, графіки, текстові матеріали.

Програмну реалізацію здійснено з використанням SQLite як бази даних для зберігання інформації, а також мови програмування Python для створення основних функціональних модулів. Розроблений інтерфейс дозволяє легко додавати нові дані для запам’ятовування, налаштовувати параметри повторення.

Програмне забезпечення було протестована для оцінки його впливу на ефективність запам’ятовування інформації. За результатами тестування вдалося підтвердити доцільність обраного підходу, а також визначити напрями подальшого вдосконалення, зокрема щодо його інтеграції з іншими освітніми платформами.

Розроблена інформаційна технологія відповідає сучасним вимогам до систем підтримки навчання та може бути ефективно застосована в освітній діяльності, професійному розвитку та самонавчанні. Мета роботи досягнута, а поставлені задачі виконані у повному обсязі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Методи запам’ятовування інформації різними особистостями / В.Є. Домбровська, В. С. Озеранський – Молодь в науці: дослідження, проблеми, перспективи (МН-2025) – [Електронний ресурс]. – Тип доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22838

2. Соціально-філософський аналіз особистості як цілісної системи / Макєшина Юлія. – [Електронний ресурс]. – Тип доступу: http://humstudies.com.ua/article/view/261861

3. Вплив графічної інформації на запам’ятовуваніст / Надточій Діана. – [Електронний ресурс]. – Тип доступу: https://openarchive.nure.ua/server/api/core/bitstreams/78a00d25-a86a-4c25-be4e-031a877933c9/content

4. Про індивідуалізацію навчання іншомовної мовленнєвої діяльності шляхом урахування сенсорно-перцептивних переваг особистості. / Кисельова Г.І., – [Електронний ресурс]. – Тип доступу: https://repository.sspu.edu.ua/items/eb3210af-ed29-4656-b01f-be353be1cbbf

5. Використання методів нейролінгвістики у навчанні іноземним мовам / Мартинчук О.О., Дьомочка Л.В., – [Електронний ресурс]. – Тип доступу: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/1f41902b-86f9-4d31-9d0c-6fcd8d78eabe/content

6. Визначення типу репрезентативної системи у дітей шкільного віку. / Форманюк Ю.В., – [Електронний ресурс]. – Тип доступу: http://repositsc.nuczu.edu.ua/bitstream/123456789/16418/1/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D1%97%2024.06.2022%20%D0%BC.%20%D0%9E%D0%B4%D0%B5%D1%81%D0%B0.%20%281%29.pdf#page=97

7. Використання кінетичної ментальної уяви у навчанні фізики / Рябко А. В., Сизьон О. О. – [Електронний ресурс]. – Тип доступу: https://sci-conf.com.ua/wp-

content/uploads/2024/05/EUROPEAN-CONGRESS-OF-SCIENTIFIC-ACHIEVEMENTS-20-22.05.24.pdf#page=236

8. Ефективні мнемонічні прийоми навчання іншомовної лексики / Бєляєва О.М – [Електронний ресурс]. – Тип доступу: https://repository.pdmu.edu.ua/items/92bc3e09-f79b-4c9b-8672-d12e01e699d2

9. Використання методу інтервального повторення слів в процесі вивчення англійської мови./ Баланова Т.В., – [Електронний ресурс]. – Тип доступу: http://eprints.zu.edu.ua/26807/1/49.pdf

10. Побудова концептуальної моделі інформаційної системи / Кільченко А.В. – [Електронний ресурс]. – Тип доступу: https://lib.iitta.gov.ua/id/eprint/925/

11. Застосування сучасних технологій навчання іноземної мови на основі аналізу кривої забування / Горідько Н. М. , Бабич М. Є. – [Електронний ресурс]. – Тип доступу: https://pednauk.cusu.edu.ua/index.php/pednauk/article/view/1933

12. Дидактичні принципи вивчення китайської лексики. / Кулага Д. Є.– [Електронний ресурс]. – Тип доступу: https://kmvodpmd.pnu.edu.ua/wp-content/uploads/sites/18/2024/04/2023-tezy-vyk-1.pdf#page=219

13. Douglas Crockford. JavaScript: The Good Parts, 2008. 2c.

14. Patrick Niemeyer, Jonathan Knudsen. Learning Java, 2005. 7 c.

15. Рiхтер Джеффрі**.** CLR via C#**,** 2020. 9 c.

16. Python. – [Електронний ресрс]. – Тип доступу: https://www.python.org/

17. Grant Allen, Mike Owens. The Definitive Guide to SQLite. Second Edition, 2010. 1 c.

18. Ateeq ur Rehman, Muhammad Zakarya. Exploring TinyDB for a Low Traffic Solution and Query Optimization in Sensor Networks. 2013. 1–8 c.

19. Kyle Banker, Douglas Garrett, Peter Bakkum, Shaun Verch. MongoDB in Action: Covers MongoDB version 3.0. 2016. 39 c.

20. Flask. – [Електронний ресурс]. – Тип доступу: https://pythonbasics.org/what-is-flask-python/

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

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія для запам'ятовування інформації

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 93,00% Схожість 7,00%

Аналіз звіту подібності (відмітити потрібне):

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

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

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

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

Автор роботи _____ Домбровська В.Є.

Керівник роботи _____ Озеранський В.С.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
import os
import sqlite3
from flask import Flask, request, session, g, redirect, url_for, abort, \
    render_template, flash

app = Flask(__name__)
app.config.from_object(__name__)
nameDB='cards-jwasham.db'
pathDB='db'

def load_config():
    app.config.update(dict(
        DATABASE=os.path.join(app.root_path, pathDB, nameDB),
        SECRET_KEY='development key',
        USERNAME='admin',
        PASSWORD='default'
    ))
    app.config.from_envvar('CARDS_SETTINGS', silent=True)

if __name__ == "__main__" or __name__ == "flash_cards":
    load_config()

def connect_db():
    rv = sqlite3.connect(app.config['DATABASE'])
    rv.row_factory = sqlite3.Row
    return rv


def init_db():
    db = get_db()
    with app.open_resource('data/schema.sql', mode='r') as f:
        db.cursor().executescript(f.read())
    db.commit()

def get_db():
    """Opens a new database connection if there is none yet for the
    current application context.
```

```
    """
    if not hasattr(g, 'sqlite_db'):
        g.sqlite_db = connect_db()
    return g.sqlite_db


@app.teardown_appcontext
def close_db(error):
    """Closes the database again at the end of the request."""
    if hasattr(g, 'sqlite_db'):
        g.sqlite_db.close()

@app.route('/')
def index():
    if session.get('logged_in'):
        return redirect(url_for('Home'))
    else:
        return redirect(url_for('login'))


@app.route('/cards')
def cards():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, type, front, back, known
        FROM cards
        ORDER BY id DESC
    '''
    cur = db.execute(query)
    cards = cur.fetchall()
    tags = getAllTag()
    return render_template('cards.html', cards=cards, tags=tags, filter_name="all")


@app.route('/filter_cards/<filter_name>')
def filter_cards(filter_name):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
```

```
    filters = {
        "all":      "where 1 = 1",
        "general":  "where type = 1",
        "code":     "where type = 2",
        "known":    "where known = 1",
        "unknown":  "where known = 0",
    }

    query = filters.get(filter_name)
    if(query is None):
        query = "where type = {0}".format(filter_name)
        filter_name = int(filter_name)

    if not query:
        return redirect(url_for('show'))

    db = get_db()
    fullquery = "SELECT id, type, front, back, known FROM cards " + \
        query + " ORDER BY id DESC"
    cur = db.execute(fullquery)
    cards = cur.fetchall()
    tags = getAllTag()
    return render_template('show.html', cards=cards, tags=tags,
filter_name=filter_name)


@app.route('/add', methods=['POST'])
def add_card():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO cards (type, front, back) VALUES (?, ?, ?)',
               [request.form['type'],
                request.form['front'],
                request.form['back']
                ])
    db.commit()
    flash('New card was successfully added.')
    return redirect(url_for('cards'))
```

```
@app.route('/edit/<card_id>')
def edit(card_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, type, front, back, known
        FROM cards
        WHERE id = ?
    '''
    cur = db.execute(query, [card_id])
    card = cur.fetchone()
    tags = getAllTag()
    return render_template('edit.html', card=card, tags=tags)


@app.route('/edit_card', methods=['POST'])
def edit_card():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    selected = request.form.getlist('known')
    known = bool(selected)
    db = get_db()
    command = '''
        UPDATE cards
        SET
          type = ?,
          front = ?,
          back = ?,
          known = ?
        WHERE id = ?
    '''
    db.execute(command,
               [request.form['type'],
                request.form['front'],
                request.form['back'],
                known,
                request.form['card_id']
                ])
    db.commit()
    flash('Card saved.')
```

```
    return redirect(url_for('show'))


@app.route('/delete/<card_id>')
def delete(card_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('DELETE FROM cards WHERE id = ?', [card_id])
    db.commit()
    flash('Card deleted.')
    return redirect(url_for('cards'))

@app.route('/memorize')
@app.route('/memorize/<card_type>')
@app.route('/memorize/<card_type>/<card_id>')
def memorize(card_type, card_id=None):
    tag = getTag(card_type)
    if tag is None:
        return redirect(url_for('cards'))

    if card_id:
        card = get_card_by_id(card_id)
    else:
        card = get_card(card_type)
    if not card:
        flash("You've learned all the '" + tag[1] + "' cards.")
        return redirect(url_for('show'))
    short_answer = (len(card['back']) < 75)
    tags = getAllTag()
    card_type = int(card_type)
    return render_template('memorize.html',
                           card=card,
                           card_type=card_type,
                           short_answer=short_answer, tags=tags)

@app.route('/memorize_known')
@app.route('/memorize_known/<card_type>')
@app.route('/memorize_known/<card_type>/<card_id>')
def memorize_known(card_type, card_id=None):
    tag = getTag(card_type)
```

```
    if tag is None:
        return redirect(url_for('cards'))

    if card_id:
        card = get_card_by_id(card_id)
    else:
        card = get_card_already_known(card_type)
    if not card:
        flash("You haven't learned any '" + tag[1] + "' cards yet.")
        return redirect(url_for('show'))
    short_answer = (len(card['back']) < 75)
    tags = getAllTag()
    card_type = int(card_type)
    return render_template('memorize_known.html',
                           card=card,
                           card_type=card_type,
                           short_answer=short_answer, tags=tags)


def get_card(type):
    db = get_db()

    query = '''
      SELECT
        id, type, front, back, known
      FROM cards
      WHERE
        type = ?
        and known = 0
      ORDER BY RANDOM()
      LIMIT 1
    '''

    cur = db.execute(query, [type])
    return cur.fetchone()


def get_card_by_id(card_id):
    db = get_db()

    query = '''
```

```
      SELECT
        id, type, front, back, known
      FROM cards
      WHERE
        id = ?
      LIMIT 1
    '''

    cur = db.execute(query, [card_id])
    return cur.fetchone()


@app.route('/mark_known/<card_id>/<card_type>')
def mark_known(card_id, card_type):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('UPDATE cards SET known = 1 WHERE id = ?', [card_id])
    db.commit()
    flash('Card marked as known.')
    return redirect(url_for('memorize', card_type=card_type))

@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != app.config['USERNAME']:
            error = 'Invalid username or password!'
        elif request.form['password'] != app.config['PASSWORD']:
            error = 'Invalid username or password!'
        else:
            session['logged_in'] = True
            session.permanent = True  # stay logged in
            return redirect(url_for('index'))
    return render_template('login.html', error=error)


@app.route('/logout')
def logout():
    session.pop('logged_in', None)
    flash("You've logged out")
```

```
    return redirect(url_for('index'))


def getAllTag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, tagName
        FROM tags
        ORDER BY id ASC
    '''
    cur = db.execute(query)
    tags = cur.fetchall()
    return tags


@app.route('/tags')
def tags():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    tags = getAllTag()
    return render_template('tags.html', tags=tags, filter_name="all")


@app.route('/addTag', methods=['POST'])
def add_tag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               [request.form['tagName']])
    db.commit()
    flash('New tag was successfully added.')
    return redirect(url_for('tags'))


@app.route('/editTag/<tag_id>')
def edit_tag(tag_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
```

```
    tag = getTag(tag_id)
    return render_template('editTag.html', tag=tag)


@app.route('/updateTag', methods=['POST'])
def update_tag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    command = '''
        UPDATE tags
        SET
          tagName = ?
        WHERE id = ?
    '''
    db.execute(command,
               [request.form['tagName'],
                request.form['tag_id']
                ])
    db.commit()
    flash('Tag saved.')
    return redirect(url_for('tags'))

def init_tag():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["general"])
    db.commit()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["code"])
    db.commit()
    db.execute('INSERT INTO tags (tagName) VALUES (?)',
               ["bookmark"])
    db.commit()

@app.route('/show')
def show():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
```

```
    tags = getAllTag()
    return render_template('show.html', tags=tags, filter_name="")

def getTag(tag_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    query = '''
        SELECT id, tagName
        FROM tags
        WHERE id = ?
    '''
    cur = db.execute(query, [tag_id])
    tag = cur.fetchone()
    return tag

@app.route('/bookmark/<card_type>/<card_id>')
def bookmark(card_type, card_id):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('UPDATE cards SET type = ? WHERE id = ?',[card_type,card_id])
    db.commit()
    flash('Card saved.')
    return redirect(url_for('memorize', card_type=card_type))

@app.route('/list_db')
def list_db():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    dbs = [f for f in os.listdir(pathDB) if os.path.isfile(os.path.join(pathDB, f))]
    dbs = list(filter(lambda k: '.db' in k, dbs))
    return render_template('listDb.html', dbs=dbs)

@app.route('/load_db/<name>')
def load_db(name):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    global nameDB
    nameDB=name
    load_config()
```

```
    handle_old_schema()
    return redirect(url_for('memorize', card_type="1"))

@app.route('/create_db')
def create_db():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    return render_template('createDb.html')

@app.route('/Home')
def Home():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    return render_template('Home.html')


@app.route('/init', methods=['POST'])
def init():
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    global nameDB
    nameDB = request.form['dbName'] + '.db'
    load_config()
    init_db()
    init_tag()
    return redirect(url_for('index'))

def check_table_tag_exists():
    db = get_db()
    cur = db.execute("SELECT name FROM sqlite_master WHERE type='table' AND
name='tags'")
    result = cur.fetchone()
    return result

def create_tag_table():
    db = get_db()
    with app.open_resource('data/handle_old_schema.sql', mode='r') as f:
        db.cursor().executescript(f.read())
    db.commit()

def handle_old_schema():
```

```
    result = check_table_tag_exists()
    if(result is None):
        create_tag_table()
        init_tag()

def get_card_already_known(type):
    db = get_db()

    query = '''
      SELECT
        id, type, front, back, known
      FROM cards
      WHERE
        type = ?
        and known = 1
      ORDER BY RANDOM()
      LIMIT 1
    '''

    cur = db.execute(query, [type])
    return cur.fetchone()

@app.route('/mark_unknown/<card_id>/<card_type>')
def mark_unknown(card_id, card_type):
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    db = get_db()
    db.execute('UPDATE cards SET known = 0 WHERE id = ?', [card_id])
    db.commit()
    flash('Card marked as unknown.')
    return redirect(url_for('memorize_known', card_type=card_type))

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Інформаційна технологія для запам'ятовування інформації

Виконала: студентка 2 курсу, групи 1КН-23м
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Домбровська В.Є.
(прізвище та ініціали)
Керівник: к. т. н., доц. каф. КН
Озеранський В.С.
(прізвище та ініціали)
«05» 12 2024 р.

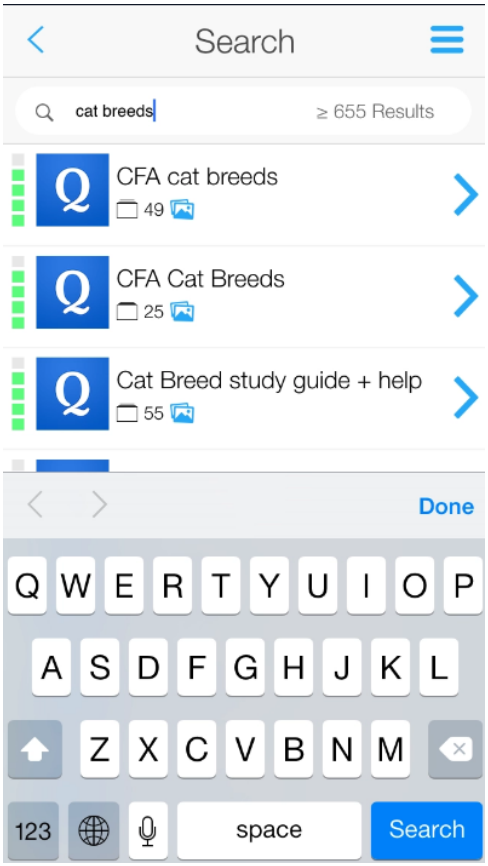

Рисунок В.1 – Приклад інтерфейсу для додатку Anki

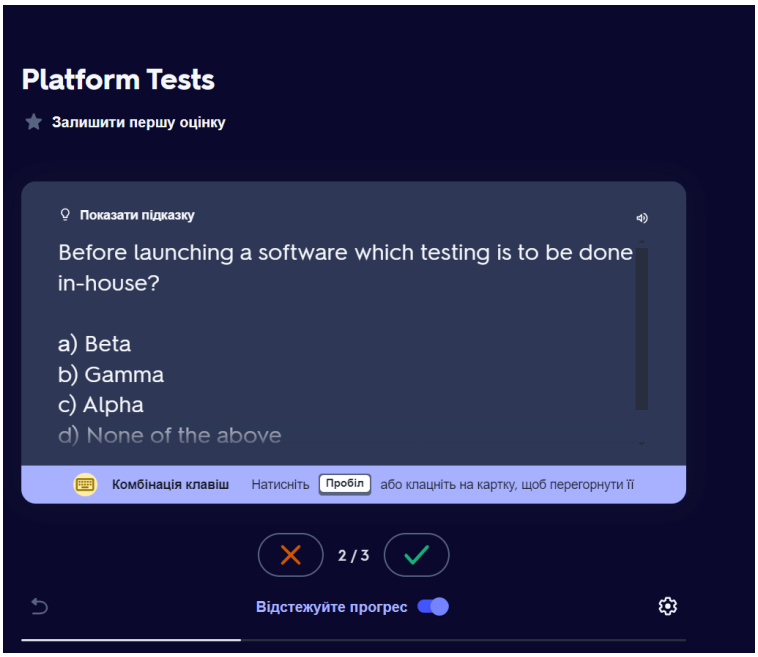

Рисунок В.2 – Приклад інтерфейсу для додатку Quizlet

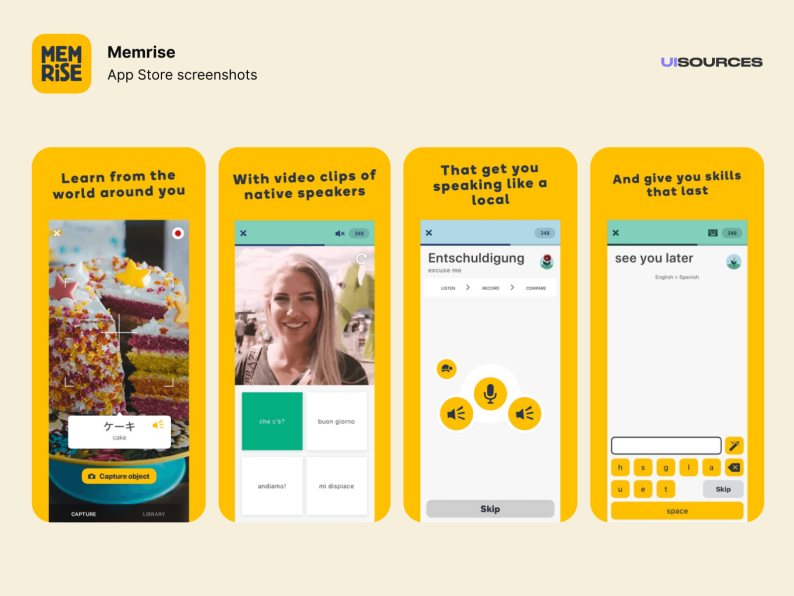

Рисунок 1.3 – Приклад інтерфейсу для додатку Memrise

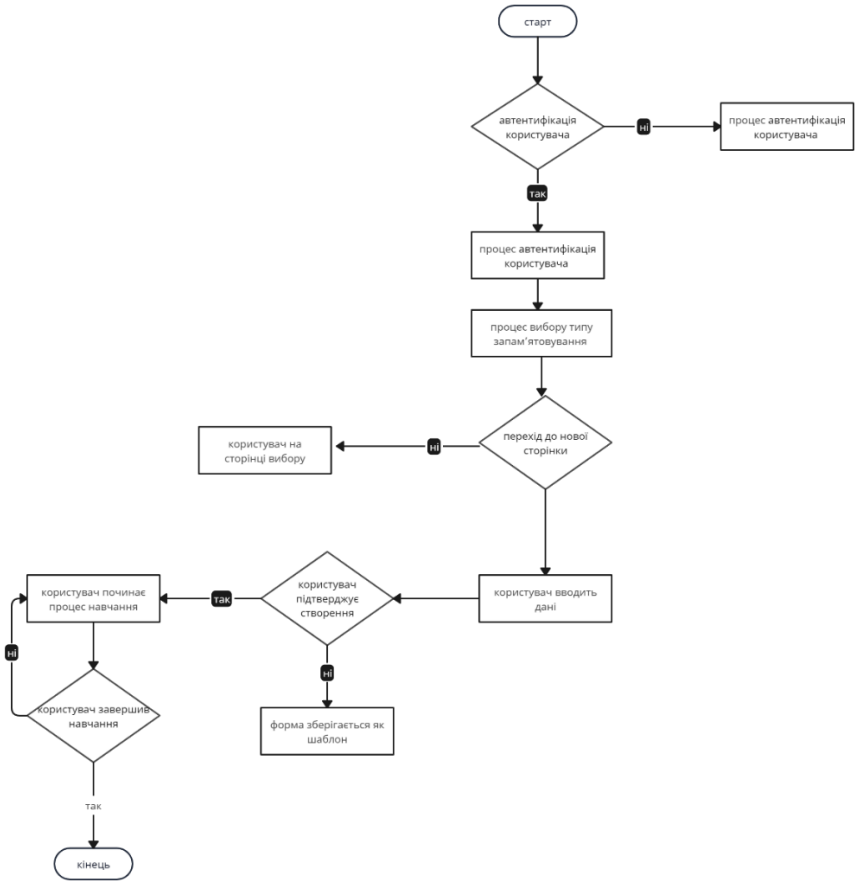

Рисунок 2.1- Схема алгоритму функціонування інформаційної технології запам'ятовування інформації

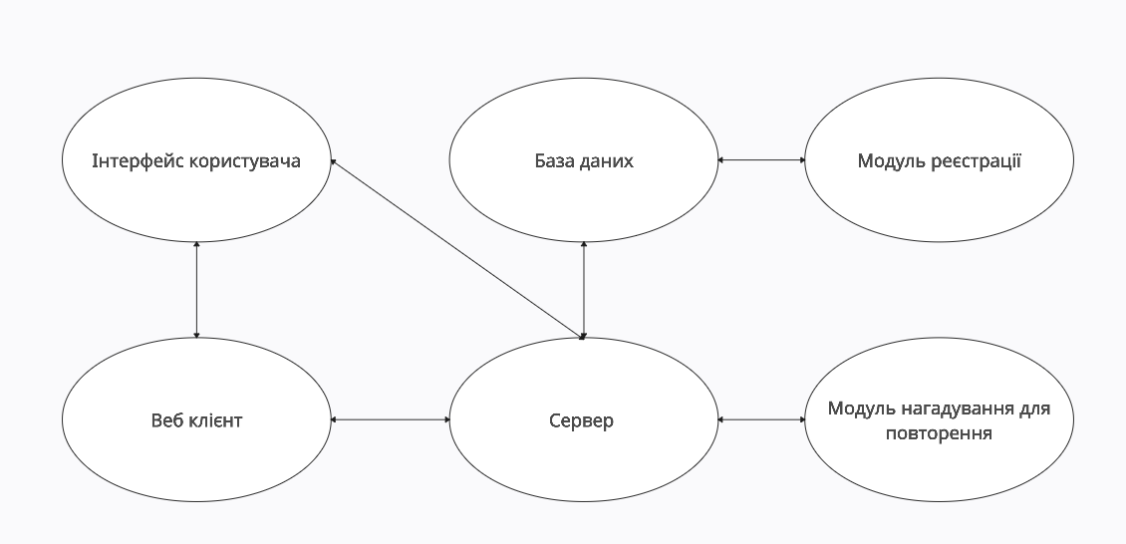

Рисунок 2.2 – Структурна схема взаємодії компонентів технології

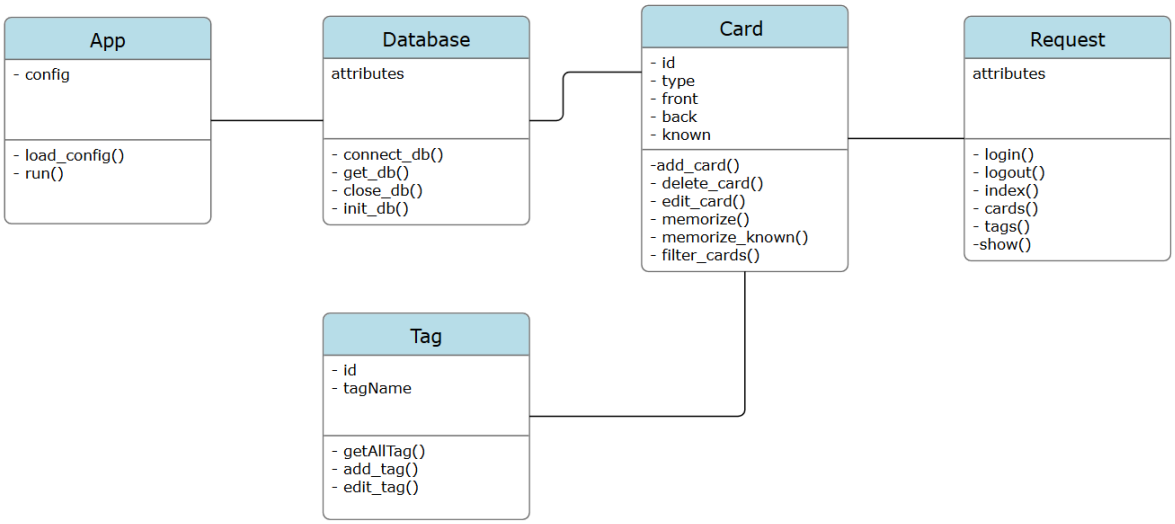

Рисунок В.5 – UML діаграма класів

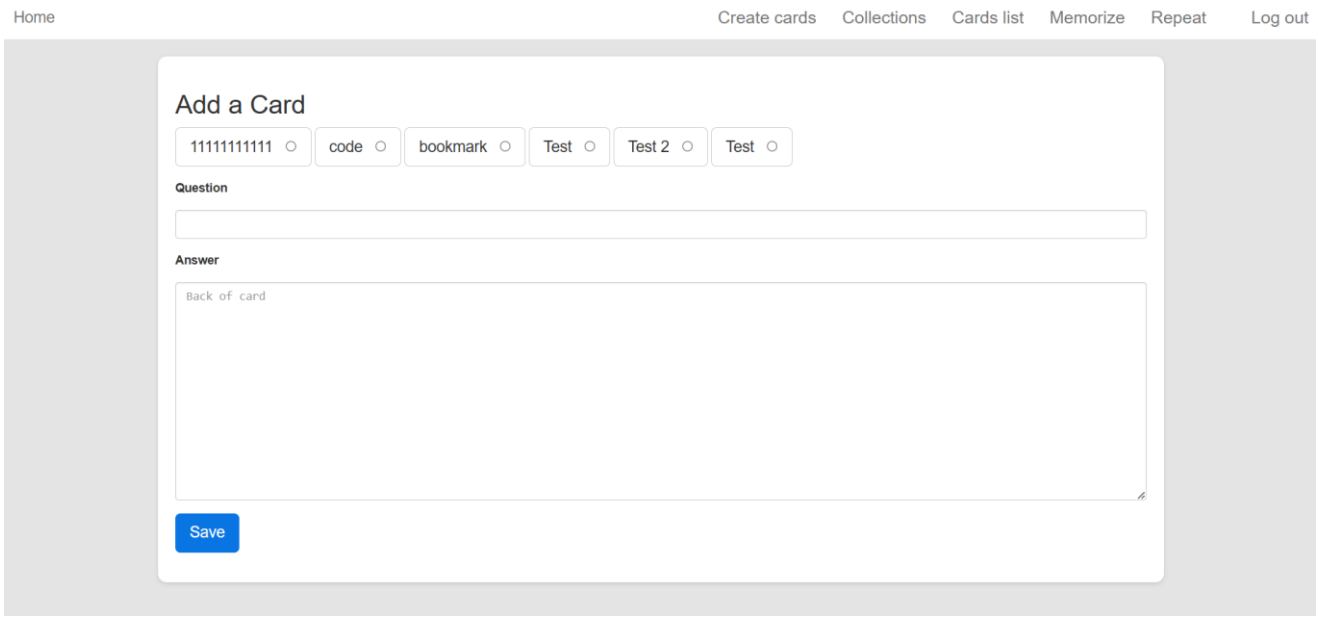

Рисунок В.6 – Форма створення картки

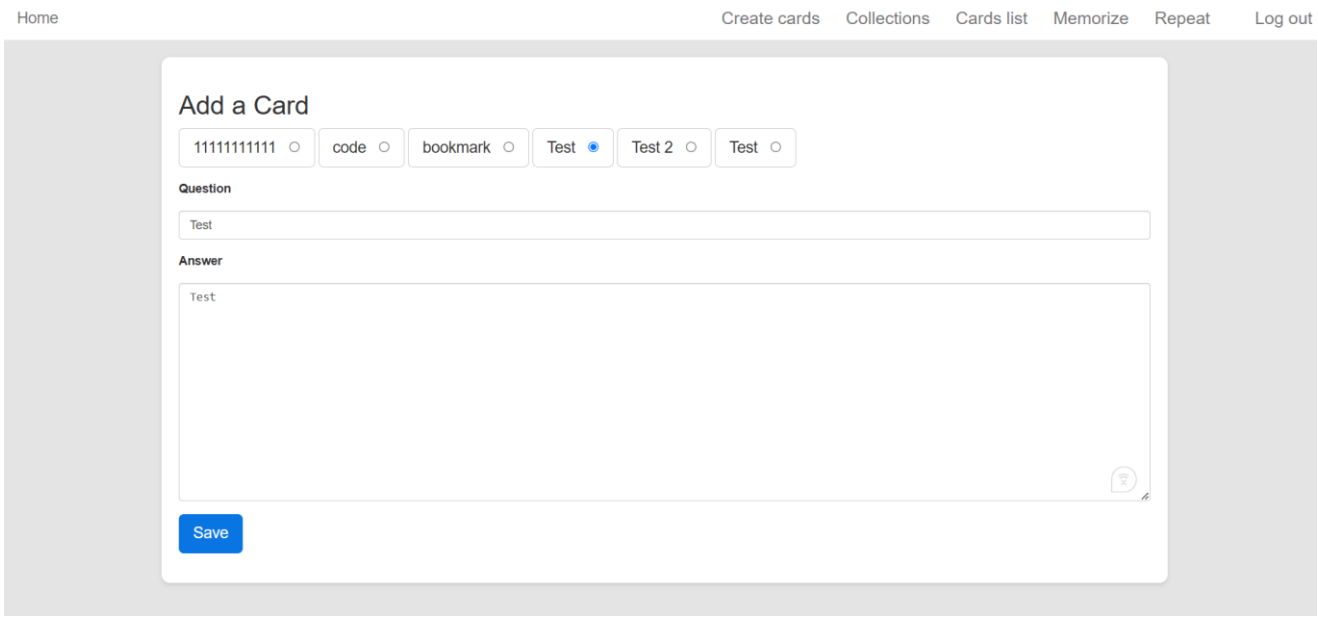

Рисунок В.7 – Заповненні поля картки

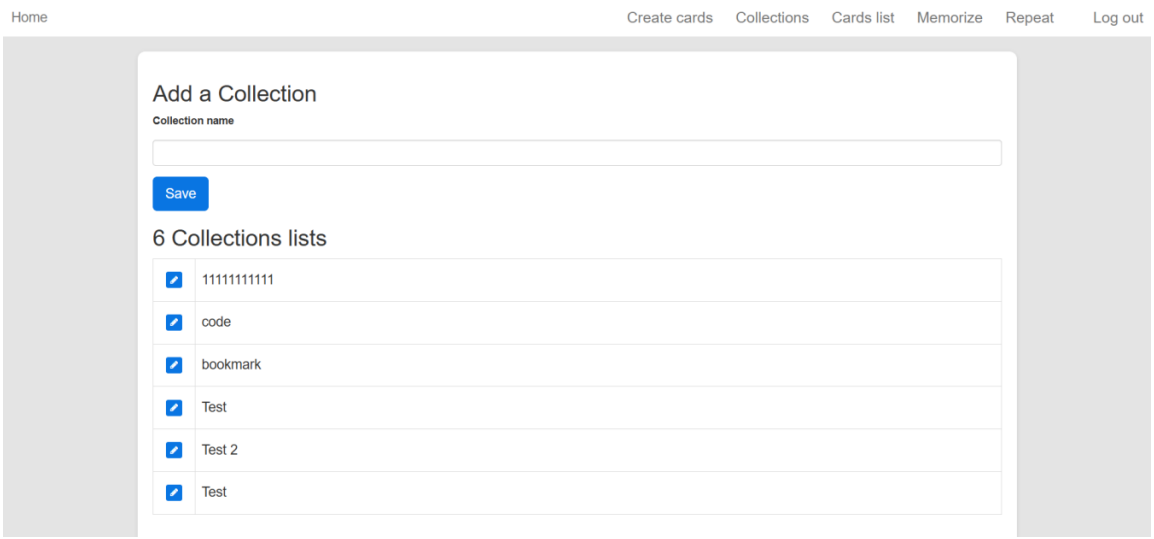

Рисунок В.8 – Форма створення колекції

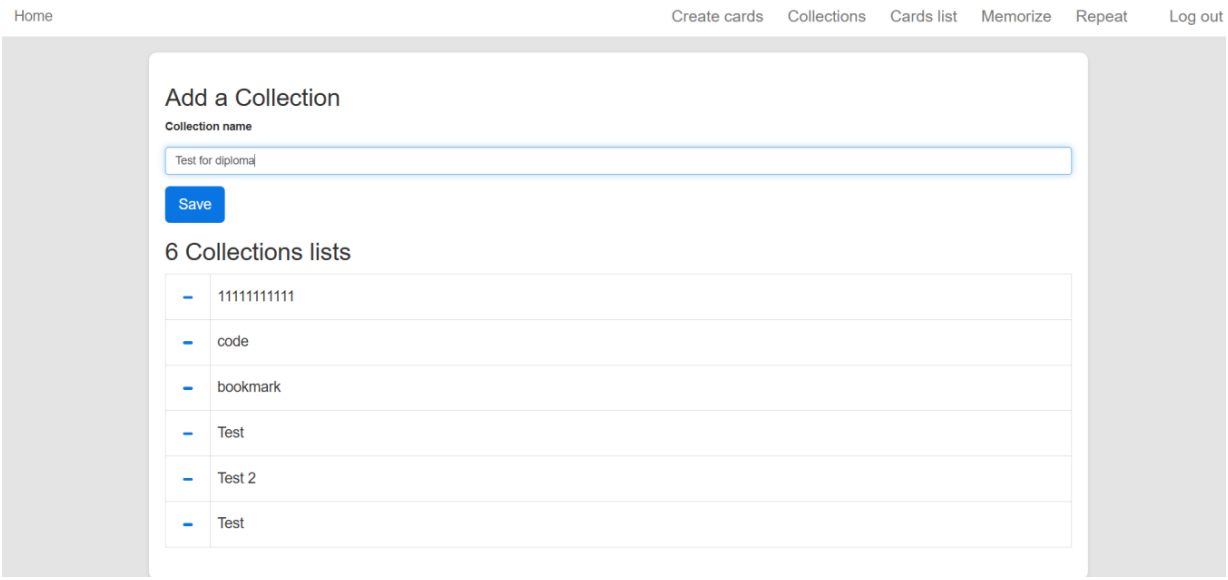

Рисунок В.9 – Заповнення поля колекції

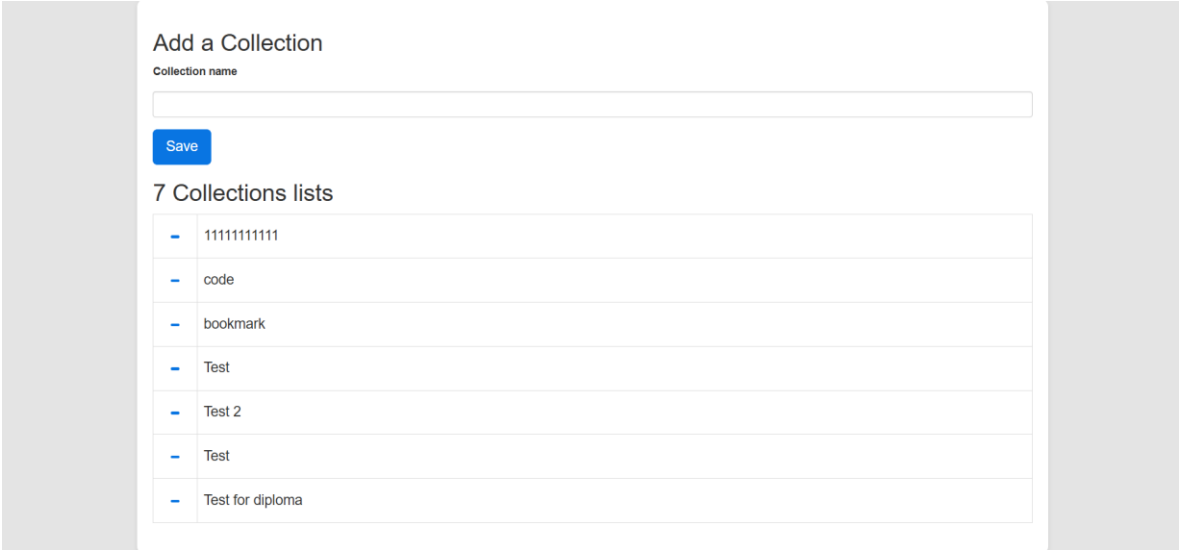

Рисунок В.10 – Приклад створеної колекції

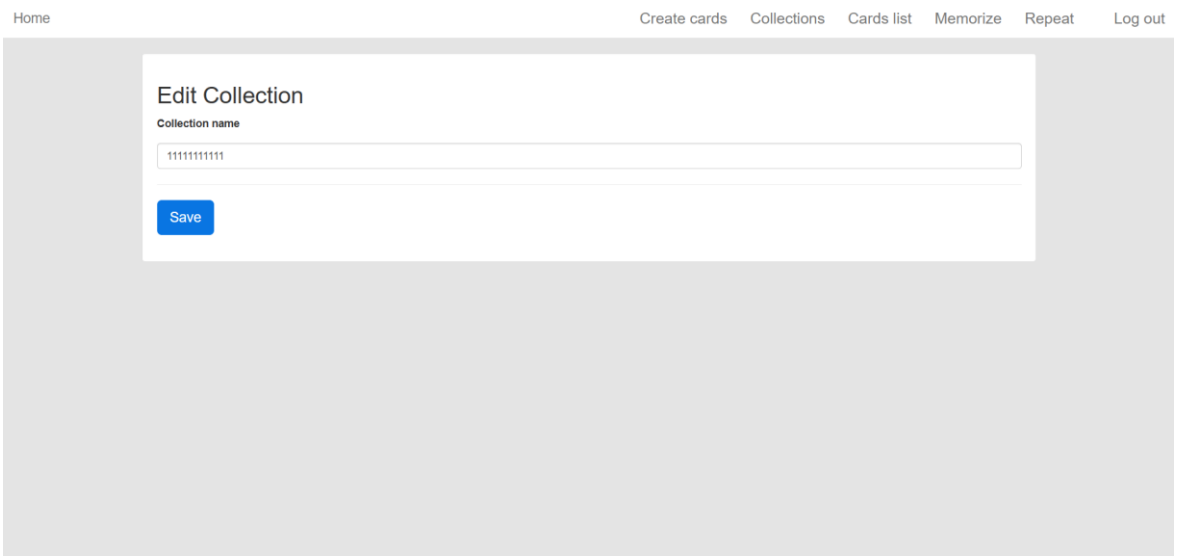

Рисунок В.11 – Форма редагування колекції

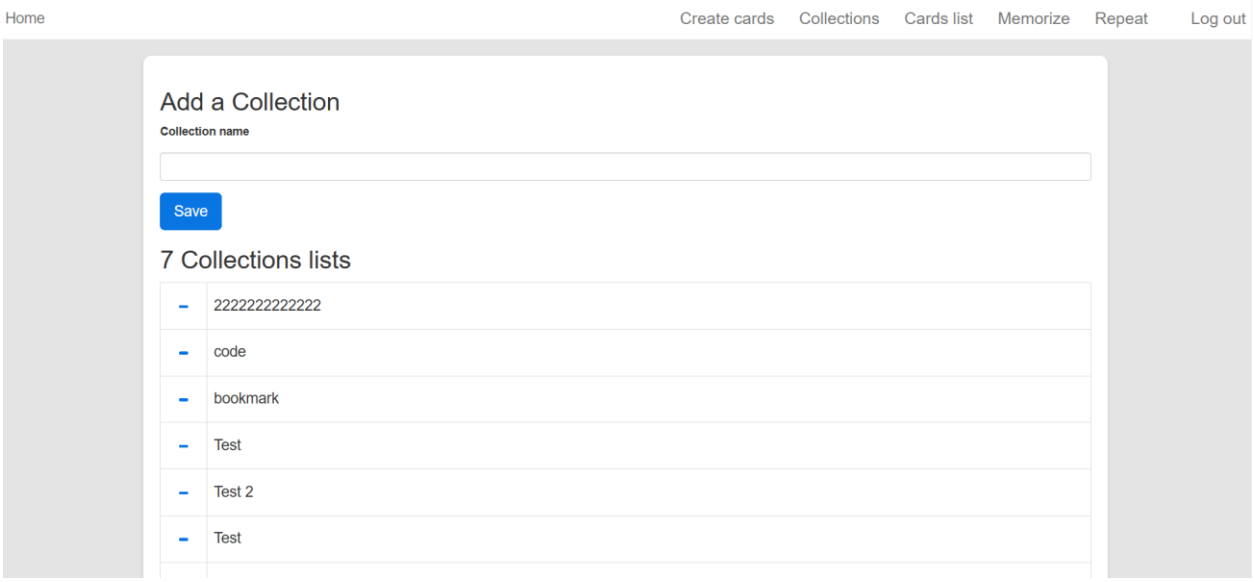

Рисунок В.12 – Відредагована колекція

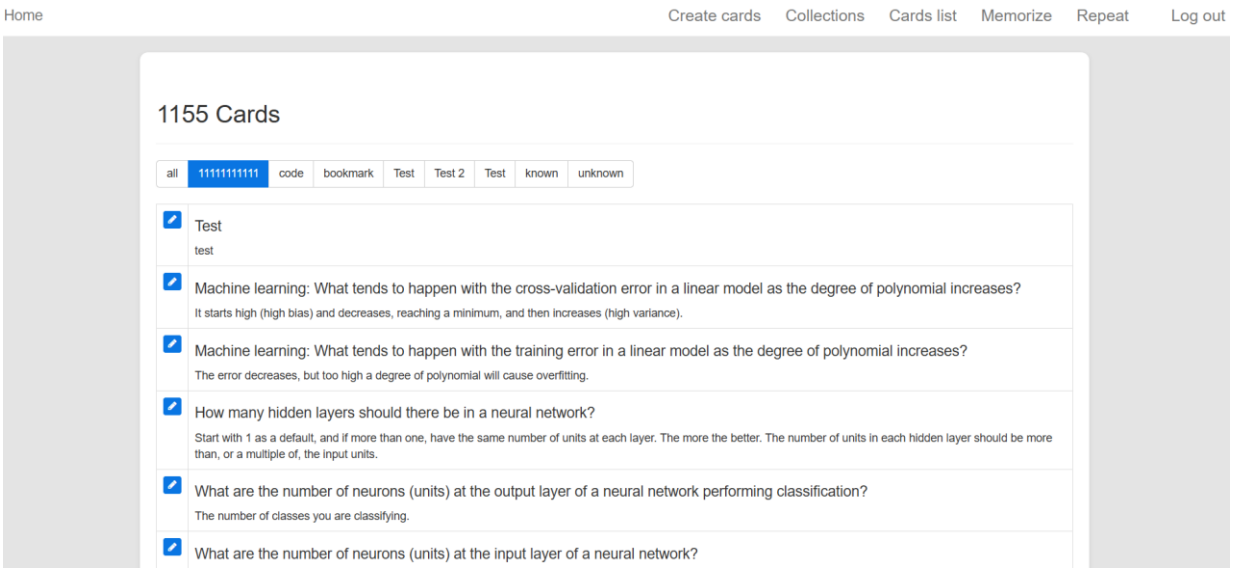

Рисунок В.13 – Сторінка з переліком карток та колекцій

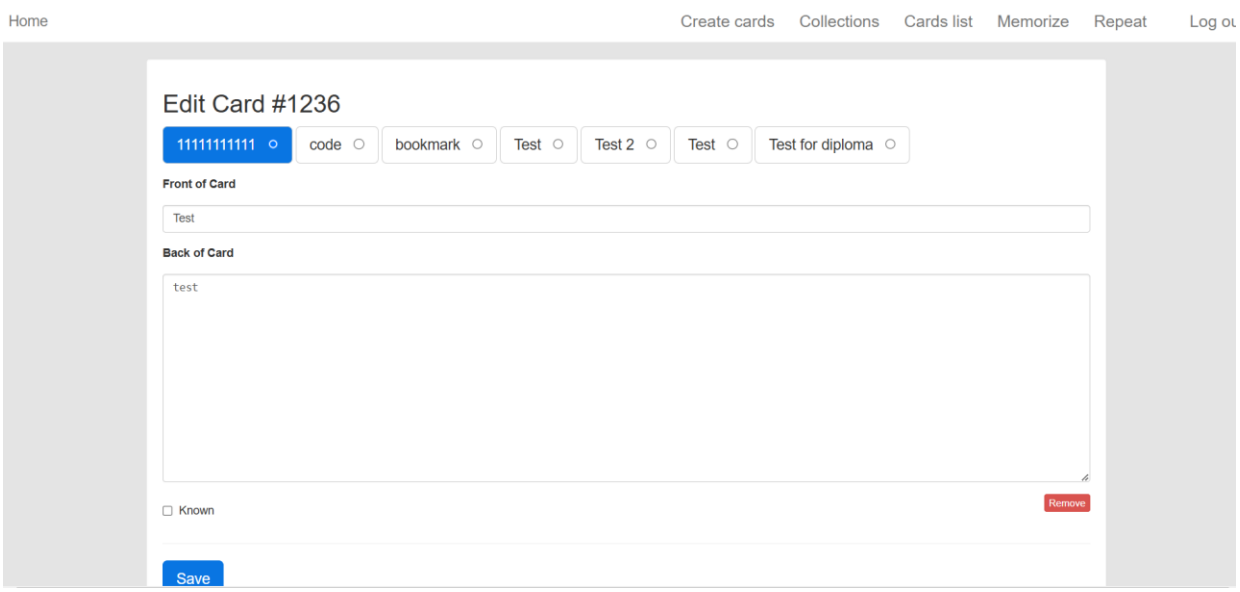

Рисунок В.14 – Форма редагування картки

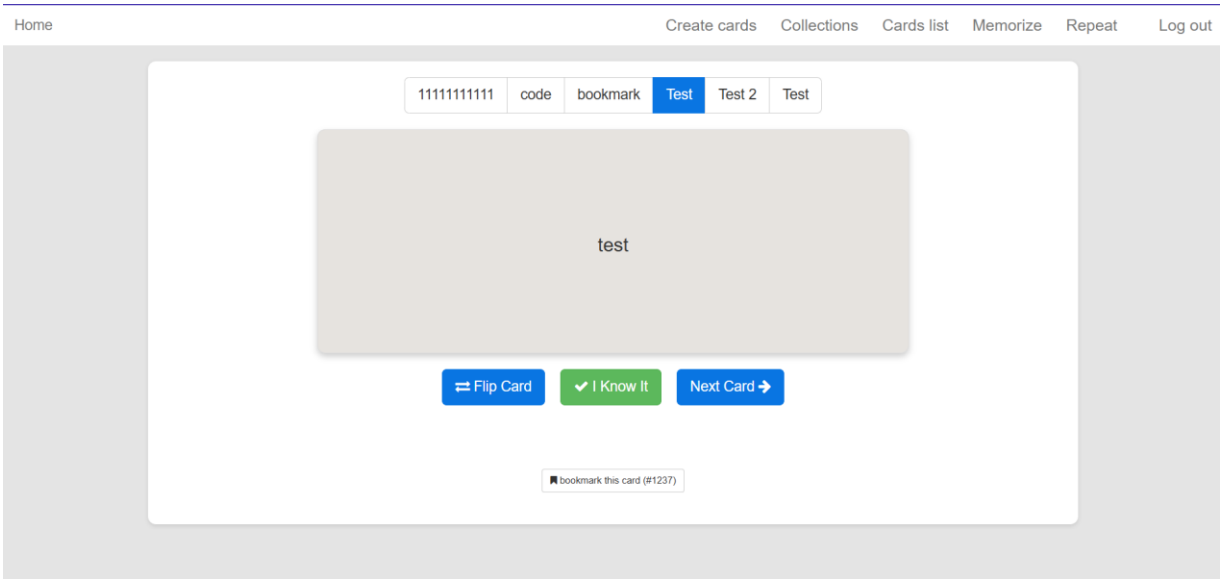

Рисунок В.15 – Передня сторона картки

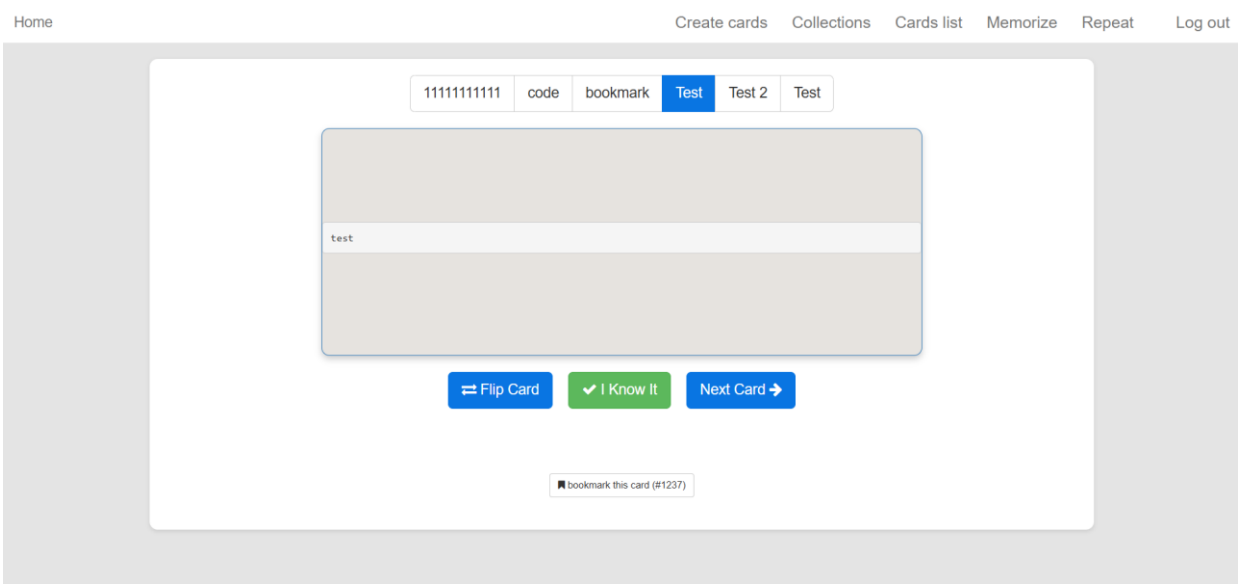

Рисунок В.16 – Задня сторона картки

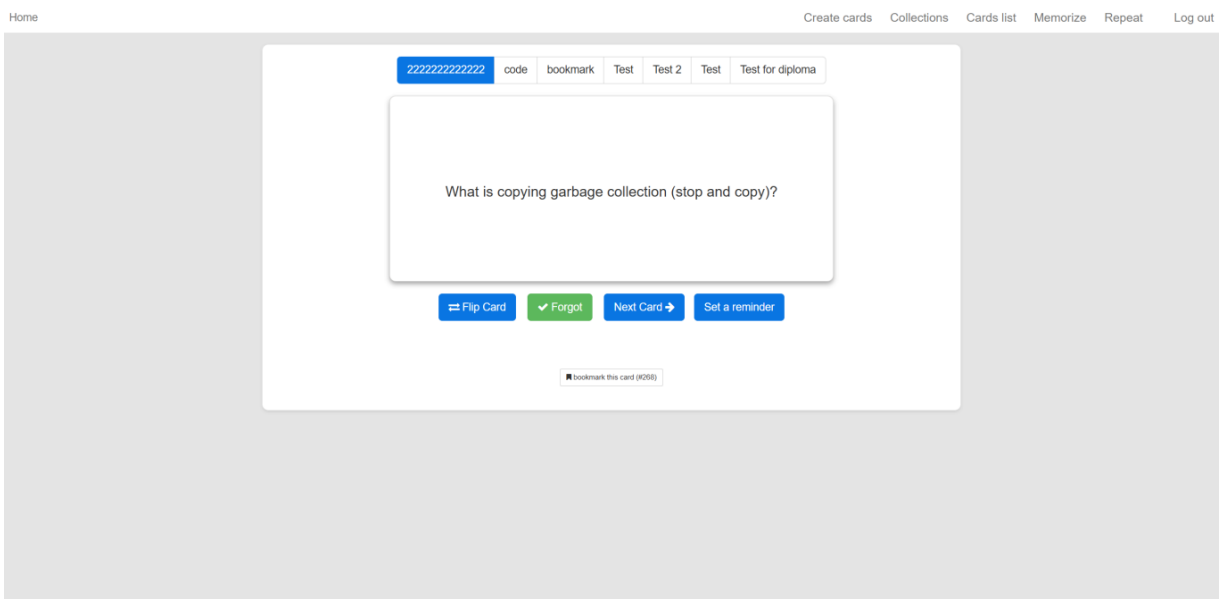

Рисунок В.17 – Передня сторона картки для повторення

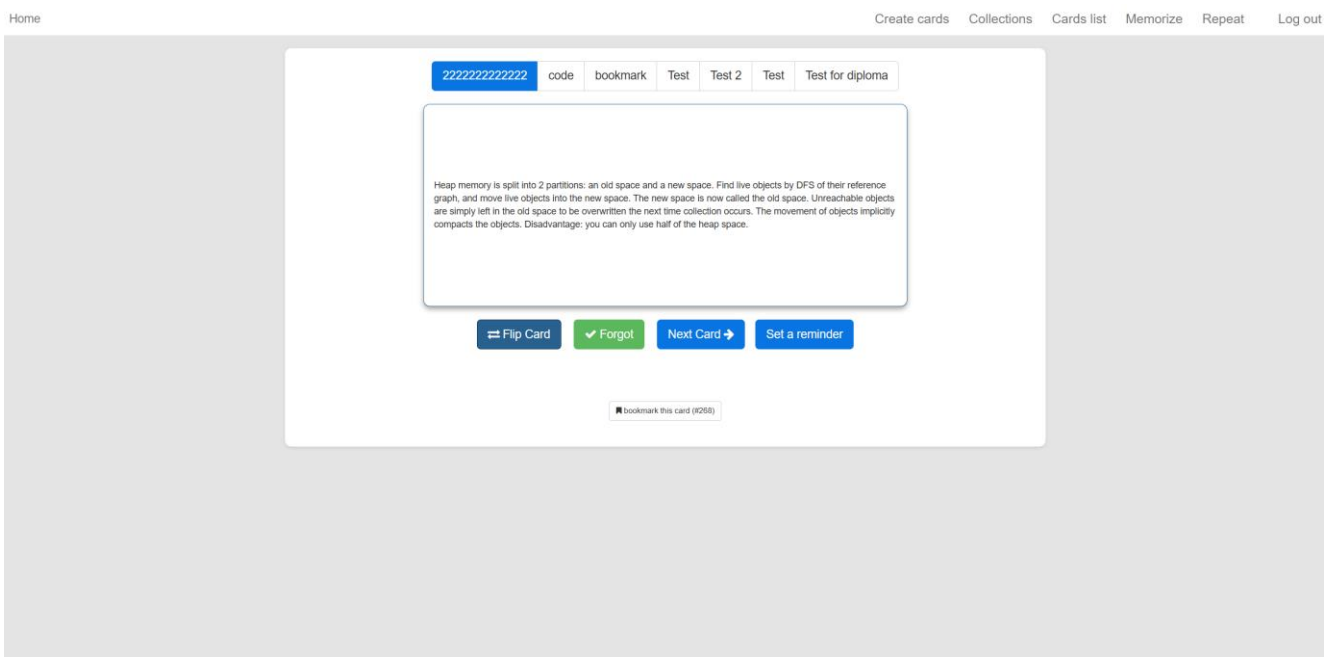

Рисунок В.18 – Задня сторона картки для повторення

Додаток Г (довідниковий)

Інструкція користувача

Щоб запустити програму потрібно відкрити PyCharm і натиснути на кнопку запуску зверху з права. Після чого в терміналі з’явиться посилання по якому користувачу треба перейти.

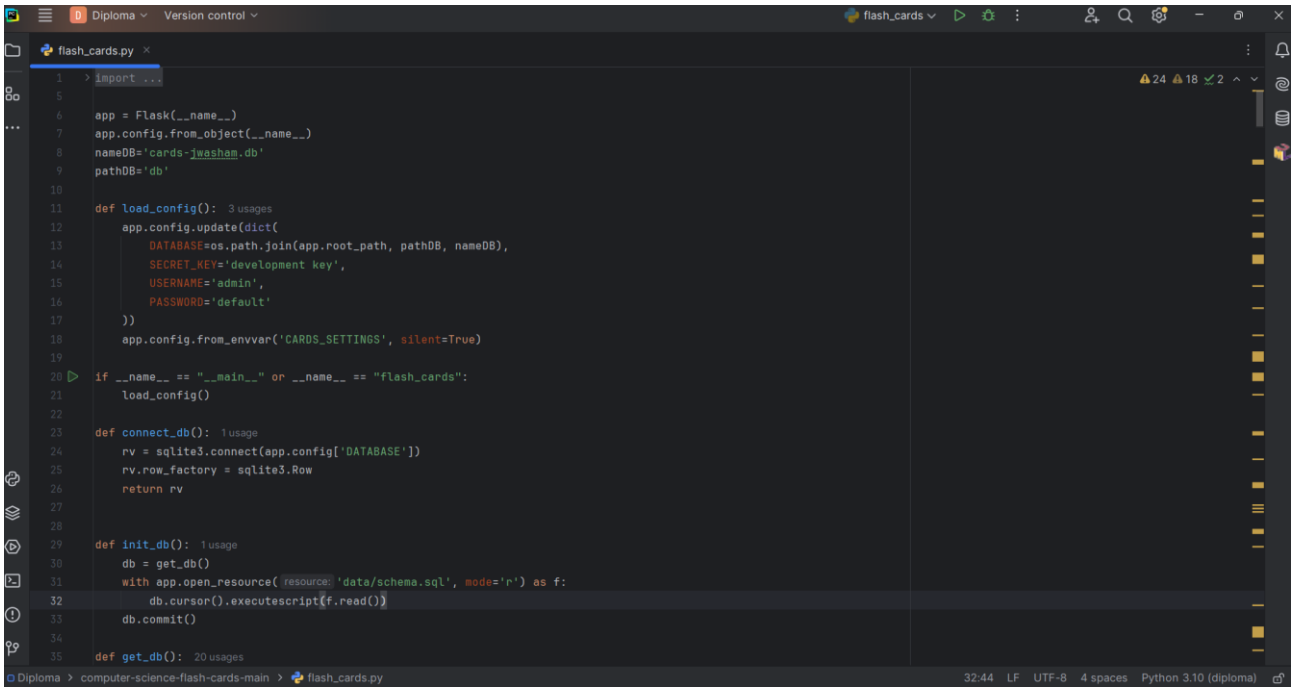

Рисунок Г.1 – Вікно додатку PyCharm

Після чого користувач перейде до гоорвної сторінки де він може обрати чи хоче він створити новий акаунт чи зайти до існуючого. Для вхолду можна використати такі дані:

- Логін – admin
- Пароль – default

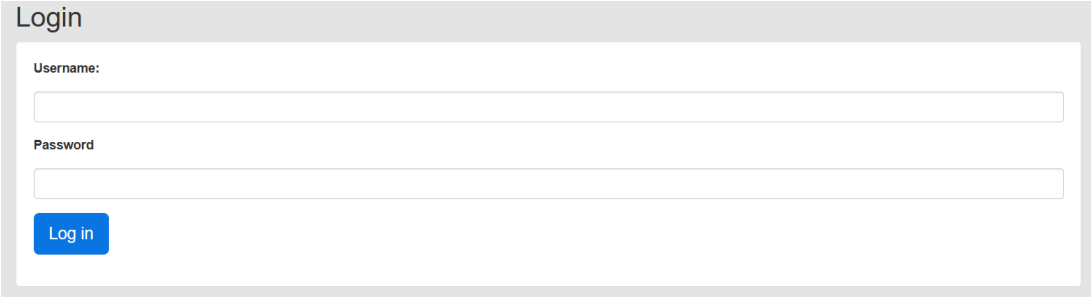

Рисунок Г.2 – Вікно входу до додатку

Щоб створити нову картку потрібно перейти на вкладку “Create cards”. Потрібно заповнити поля “Question” та “Answer”. Після чого обираємо до якої колекції буде відноситися це питання і натиснути кнопку зберегти.

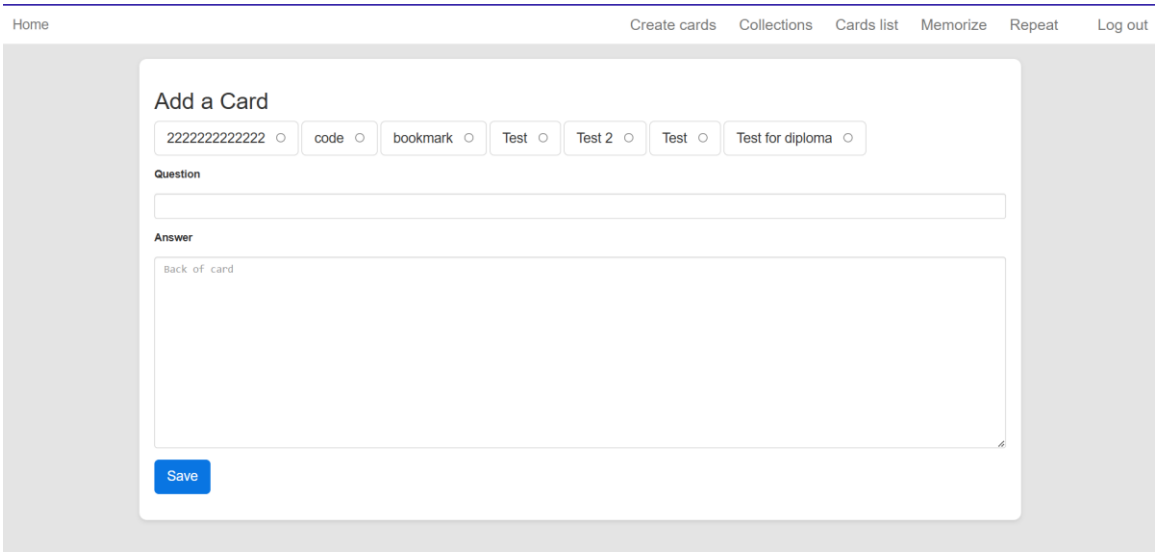

Рисунок Г.3 – Вікно створення картки

Щоб створити колекцію потрібно перейти до сторінки “Collection”. Де ми вводимо назву колекції і зберігаємо її. Вона з’явиться у загальному списку колекцій де ми зможемо її редагувати або видалити.

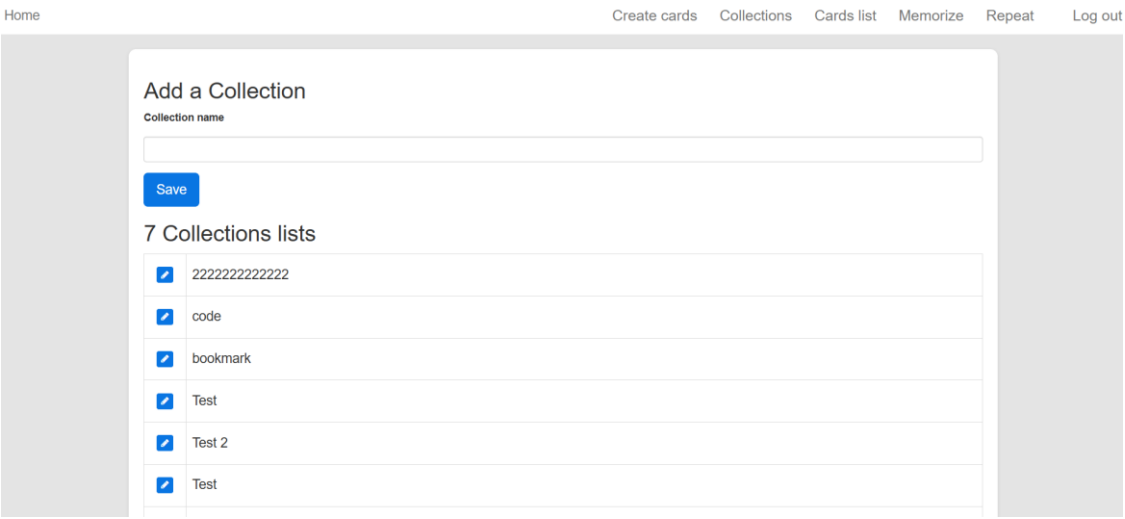

Рисунок Г.3 – Вікно створення колекції

На сторінці “Cards list” ми можемо переглянути всі створені картуи. На цій сторінці користувач може відредагувати картку або змінити колекцію до якої вона відноситься.

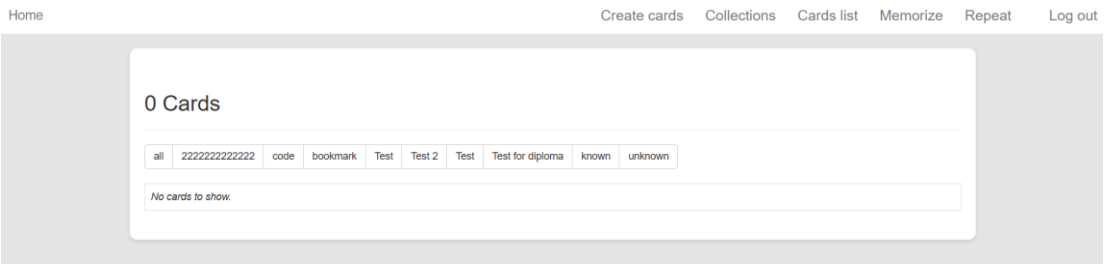

Рисунок Г.4 – Вікно зі списком всіх карток

На сторінці “Memorize” користувач може обрати колекцію для запам’ятовування після чого питання з неї будуть відображатися у вигляді картки. Спочатку користувач бачить питання і якщо він хоче себе перевірити то натискає на кнопку “Flip Card”. Картка перевертається і користувач перевіряє себе. Якщо він знає відповідь, то позначає картку як “I know it”, а якщо ні то просто її пропускає.

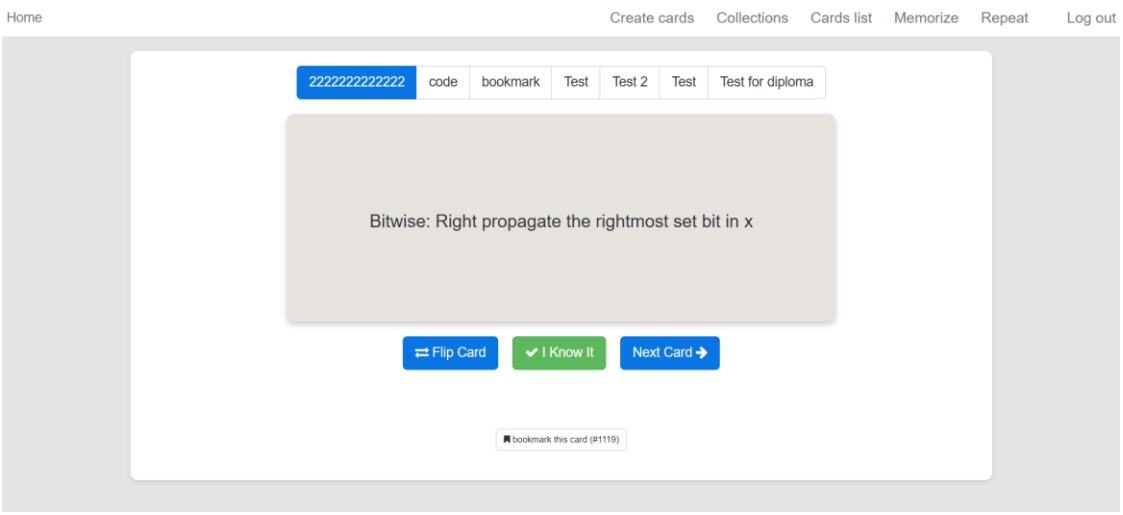

Рисунок Г.5 – Вікно з функціоналом для запам’ятовування

На сторінці “Repeat” користувач може повторити вивчений матеріал. Тут будуть всі картки які були позначенні як “I know it”. Користувач може позначити картку як “Forgot” і вона повернеться до всі невивчених карток. Також є можливість поставити нагадування для інтервального повторення.

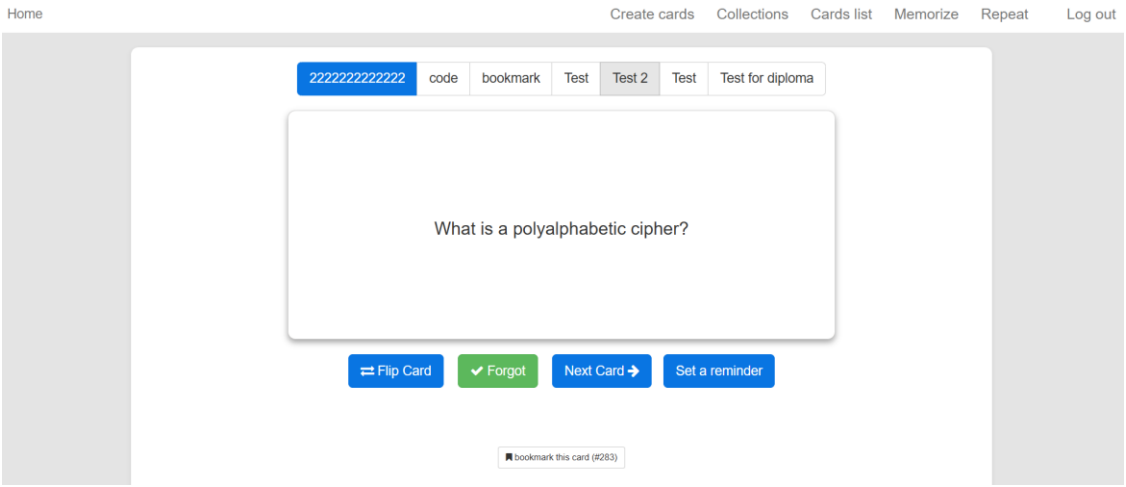

Рисунок Г.6 – Вікно з функціоналом для повторення вивченого матеріалу