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:

«WEB - орієнтована інформаційна технологія для розробки експертних систем. Частина 1»

Виконав: студент 2-го курсу, групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Левченко Н. Б.
(прізвище та ініціали)

Керівник: д.т.н., професор каф. КН
Яровий А. А.
(прізвище та ініціали)
«15» грудня 2022 р.

Опонент: д.т.н., професор каф. АПТ
Бісікало О.В.
(прізвище та ініціали)
«15» грудня 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» грудня 2022 р.

Вінниця ВНТУ - 2022 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.
14 вересня **2022 року**

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Левченко Нікіта Борисович
(прізвище, ім'я, по батькові)

1. Тема роботи WEB - орієнтована інформаційна технологія для розробки експертних систем. Частина 1

керівник роботи д.т.н., професор кафедри КН Яровий А. А.
затверджені наказом вищого навчального закладу від "14" вересня 2022 року №____
2. Строк подання студентом роботи 18 листопада 2022 року

3. Вихідні дані до роботи:
середня швидкість відповіді серверу до 1000мс, максимальний розмір бази знань < 50mb, серверна підтримка функції авторизації, максимальний час завантаження файлу бази знань в хмарне середовище 1000мс
4. Зміст текстової частини:
Вступ, аналіз сучасного стану розвитку технології надання рекомендацій, аналіз моделей надання рекомендацій та проектування WEB-орієнтованої інформаційної технології для розробки експертних систем, програмна реалізація серверної частини WEB середовища для розробки експертних систем, економічна частина, висновки, перелік використаних джерел, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Узагальнена структурна схема WEB - орієнтованої інформаційної технології для розробки експертних систем; схема циклу роботи інтерпретатора; схема алгоритму процесу реєстрації / авторизації користувача; схема алгоритму процесу створення / завантаження бази знань; схема алгоритму процесу опитування користувача; Приклад звіту автоматизованого тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		Завдання видав	Виконання прийняв
1-3	Яровий А.А., д.т.н., проф. каф. КН	14.09.2022	11.11.2022
4	Нікіфорова Л. О., к. е. н., доц. каф. ЕПВМ	19.09.2022	12.12.2022

7. Дата видачі завдання 14 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

п/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану розвитку області надання рекомендацій експертними системами. Постановка задач дослідження	14.09.2022 - 01.10.2022	
2	Розробка серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем	02.10.2022 - 16.10.2022	
3	Програмна реалізація серверної частини веб середовища для розробки експертних систем	17.10.2022 - 7.11.2022	
4	Підготовка економічної частини	08.11.2022 - 21.11.2022	
5	Апробація та/або впровадження результатів дослідження	23.11.2022 - 01.12.2022	
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.11.2022 - 14.12.2022	

Студент _____ Левченко Н.Б.
(підпис)

Керівник роботи _____ Яровий А. А.
(підпис)

АНОТАЦІЯ

УДК 004.8

Левченко Н. Б. WEB - орієнтована інформаційна технологія для розробки експертних систем. Частина 1. Магістерська кваліфікаційна робота зі спеціальності 122 – Комп’ютерні науки, освітня програма - Системи штучного інтелекту. Вінниця: ВНТУ, 2022. 92 с.

На укр. мові. Бібліогр.: 28 назв; рис.: 24; табл. 6.

Дана магістерська кваліфікаційна робота присвячена розробці серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем. Розглянуто предметну область надання рекомендацій. Здійснено аналіз систем аналогів та аналіз об’єкту проектування WEB - орієнтованої інформаційної технології для розробки експертних систем. Також здійснено аналіз різні серверні архітектури та обґрунтовано вибір для даної задачі REST архітектури.

Розглянуто та здійснено порівняльний аналіз систем, моделей, методів та технологій надання рекомендацій. Розроблено структурну організацію WEB - орієнтованої інформаційної технології для розробки експертних систем, особливо серверної частини. Спроектовано серверну частину WEB - орієнтованої інформаційної технології для розробки експертних систем, що розташована на хмарному сервісі AWS EC2, написану мовою програмування Node.js з використанням фреймворку Express.js.

Обґрунтувано вибір середовища та вибір мови програмування для WEB - орієнтованої інформаційної технології для розробки експертних систем. Розроблено алгоритм роботи та реалізовано автоматизоване тестування програмного продукту.

На основі запропонованої інформаційної технології можна створювати власні експертні системи для вирішення різних типів завдань у різних предметних областях: консультування, навчання, діагностика, тестування, проектування тощо

Ключові слова: експертні системи, бази знань, прийняття рішень

ABSTRACT

Levchenko N.B. WEB-oriented information technology for the development of expert systems. Master's thesis in the specialty 122 - Computer sciences, educational program - Artificial intelligence systems. Vinnytsia: VNTU, 2022. 92 p.

In Ukrainian language. Bibliographer: 28 titles; drawings: 24; table 6.

This master's thesis is devoted to the development of the server part of the WEB - oriented information technology for the development of expert systems. The subject area of providing recommendations is considered. An analysis of analog systems and an analysis of the design object of WEB - oriented information technology for the development of expert systems was carried out. Various server architectures were also analyzed and the choice of the REST architecture for this task was justified.

A comparative analysis of systems, models, methods and technologies for providing recommendations was considered and carried out. The structural organization of WEB - oriented information technology for the development of expert systems, especially the server part, has been developed. The server part of WEB - oriented information technology for the development of expert systems, located on the AWS EC2 cloud service, written in the Node.js programming language using the Express.js framework, was designed.

The choice of environment and choice of programming language for WEB - oriented information technology for the development of expert systems is substantiated. The work algorithm was developed and automated testing of the software product was implemented.

On the basis of the proposed information technology, you can create your own expert systems for solving various types of tasks in various subject areas: consulting, training, diagnostics, testing, design, etc.

Keywords: expert systems, knowledge bases, decision-making

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Для розробки експертних сист. ем сьогодні найчастіше використовуються оболонки експертних систем, що дозволяють наповнювати базу знань та встановлювати правила виведення. Технологія проектування експертних систем є одним з напрямків нової галузі дослідження, яка отримала найменування штучного інтелекту. Дослідження в цій області сконцентровані на розробці та впровадженні комп'ютерних програм, здатних імітувати, відтворювати ті області діяльності людини, які вимагають мислення, певної майстерності і накопиченого досвіду На сьогоднішній час багато інформаційних систем створюються для обробки значних об’ємів інформації за короткий проміжок часу. Вони виконують значний обсяг робіт, що значно спрощують роботу не тільки інженера, а й любого робітника в залежності в якій області застосовується дане забезпечення. Все більш зрозуміло, що використання експертних систем у розробці та виробництві буде продовжувати розширюватися як у галузі застосування, так і в розширенню знань. Отже, багато підприємств користуватимуться перевагами скорочення тривалості роботи, покращенням продуктивності праці, стабільнішими та надійними за своєю структурою систем. З цією метою в багатьох великих компаніях було розроблено десятки експертних системних. Експертні системи використовуються у наступних напрямках: інтерпретація, дизайн, прогнозування, діагностика, планування, моніторинг, налагодження, ремонт, інструкція та контроль. Для реалізації платформи для розробки та супроводу експертних систем за основу були взяті сучасні веб-технології. Однією з переваг веб-орієнтованої платформи є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є міжплатформовими сервісами, що є великою перевагою над подібними оболонками для розробки експертних систем. Актуальність теми дослідження полягає в поєднанні сучасних веб-технологій та напряму штучного інтелекту «експертні системи», що не лише реалізує простоту використання експертних систем кінцевими користувачами, а й сприяє розробці складних інтелектуальних систем.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету спеціальності 122 «Комп'ютерні науки» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою магістерської кваліфікаційної роботи є автоматизація процесу розробки експертних систем та підвищення швидкодії серверної обробки інформації.

Для досягнення поставленої мети необхідно розв'язати такі наступні **завдання**:

– здійснити аналіз сучасного стану розвитку засобів розробки експертних систем;

– здійснити аналіз моделей подання знань та обґрунтування їх вибору у веб-орієнтованій програмній платформі для створення експертної системи;

– розробити алгоритм роботи програмного продукту;

– виконати проектування WEB-середовища для експертних систем;

– виконати програмну реалізацію WEB -середовища для розробки експертних систем;

– виконати економічні розрахунки.

Об'єкт дослідження – процес розробки експертної системи.

Предмет дослідження – WEB-орієнтовані програмні засоби для створення експертних систем.

Методи дослідження - методи та моделі подання знань, теорія побудови експертних систем, методи інженерії знань, методи та підходи до розробки систем штучного інтелекту, методи та підходи до розробки веб-орієнтованих програмних додатків, методи об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

Розроблено інформаційну модель автоматизованої побудови експертних систем, що відрізняється від існуючих ефективними функціональними можливостями

за рахунок використання хмарних інструментів, що забезпечило покращення показників швидкодії та якості наданих рекомендацій.

Практичне значення одержаних результатів полягає у наступному:

– розроблено алгоритми створення та збереження бази знань та проведення експертизи згідно заданої бази знань, що покращило автоматизацію процесу розробки експертних систем та підвищило швидкодії серверної обробки інформації ;

– здійснено програмну реалізацію серверної частини веб орієнтованої інформаційної технології для розробки експертних систем з розширеними функціональними можливостями.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

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

Апробація результатів роботи. Результати досліджень апробовані на LI науково-технічній конференції ВНТУ (Вінниця, 2022 р.) [1, 2].

Публікації. За результатами досліджень опубліковано дві тези доповіді [1, 2].

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ ТЕХНОЛОГІЇ НАДАННЯ РЕКОМЕНДАЦІЙ

1.1 Аналіз предметної області надання рекомендацій

Класичне тлумачення поняття інтелекту було запропоноване вченим А.Тюрінгом. Тест Тюрінга — емпіричний тест, ідея якого було запропоновано Аланом Тюрінгом у статті «Обчислювальні машини і розум» (Computing Machinery and Intelligence), опублікованій 1950 року у філософському журналі «Mind». Тюрінг задався метою визначити, чи може машина мислити [3].

Стандартне звучання закону: «Якщо комп'ютер може працювати так, що людина не може визначити, з ким вона спілкується — з іншою людиною або з машиною, — вважається, що вона пройшла тест Тюрінга» [3].

Розумні, подібні до людини машини протягом багатьох десятиліть були однією з основних тем науково-фантастичних творів. З моменту зародження сучасної обчислювальної техніки розум людей займав питання: чи можна побудувати машину, яка могла б у чомусь замінити людину. Спробою створити твердий емпіричний ґрунт для вирішення цього питання і став тест, розроблений Аланом Тюрінгом [3].

Перший варіант тесту, опублікований у 1950 році, був дещо заплутаним. Сучасна версія тесту Тюрінга є наступне завдання. Група експертів спілкується із невідомою істотою. Вони не бачать свого співрозмовника і можуть спілкуватися з ним лише через якусь ізолюючу систему, наприклад, клавіатуру. Їм дозволяється ставити співрозмовнику будь-які питання, розмовляти на будь-які теми. Якщо в кінці експерименту вони не зможуть сказати, чи спілкувалися вони з людиною або з машиною, і якщо насправді вони розмовляли з машиною, можна вважати, що ця машина пройшла тест Тюрінга [3].

Інтелектуальну програму, яка може імітувати поведінку людини-експерта у вирішенні проблем, можна визначити як експертну систему. Людські знання

складаються з предметних знань/знань предметної області та знань про вирішення проблем. Функціональність експертної системи подібна до людини-експерта, яка вирішує проблему, застосовуючи свої знання з розв’язання проблем до знань, що стосуються предметної області. Отже, експертна система - це комп'ютерна програма, яка символізує знання експерта в певній області [4, 5].

Експертна система (ЕС) - це система штучного інтелекту, що використовує накопичені знання для забезпечення високоефективного рішення задач у вузькій професійній області. Експертні системи відносяться до систем підтримки прийняття рішень (СППР), заснованим на знаннях [4, 5].

Експертні системи є відносно молодим науковим напрямком, що виник в межах досліджень із штучного інтелекту в середині 70-х років XX ст. Свій розвиток він отримав завдяки суттєвим змінам, що відбулись в цей час в технології розробки та використання програмних засобів штучного інтелекту. Найбільш важливими з них стали:

– відокремлення в програмі деякої універсальної частини (логічного виведення) від частини, яка залежить від предметної області (бази знань);

– підвищення рівня взаємодії користувача з комп’ютерною програмою [4, 5].

Важливість цих змін стає більш зрозумілою, якщо звернути увагу на головну особливість процесу створення інтелектуальної програми, який полягає в виконанні послідовності таких дій:

– визначається необхідність модифікації;

– здобуваються нові знання, які дозволяють підвищити якість функціонування системи;

– нові знання перетворюються в форму, „зрозумілу” комп’ютерній системі;

– виконується модифікація знань системи.

При створенні перших інтелектуальних програм всі вищевказані дії виконував програміст. Але скоро виникла суперечність, яка полягала в тому, що для створення

досконалих програм програмісту потрібно було мати рівень знань найкращих фахівців (експертів) у відповідній проблемній області, або експертам досконало опановувати програмування.

Традиційні СППР універсальні і застосовуються для рішення унікальних проблем у різних предметних областях, а ЕС дають відповіді на питання у вузькій предметній області і роблять висновки, що міг би зробити людина-професіонал високої кваліфікації. Інтеграція традиційної СППР із ЕС утворить більш складний вид - так називану експертну систему підтримки прийняття рішень (ЕСППР).

Така система, виходячи з загальних вимог, пропонованих до ЕС, повинна пояснювати свої ради кінцевому користувачу, і, крім того, надавати йому універсальні засоби вільного моделювання. Таким чином, саме такий варіант організації системи буде розглядатися в магістерській роботі, тому що він надає найбільше раціональний підхід до одержання результативного аналізу.

Ця технологія вже успішно застосовується в деяких областях техніки й життя суспільства – органічної хімії, пошуку корисних копалин, медичній діагностиці. Перелік типових завдань, які вирішуються експертними системами, включає:

– добування інформації з первинних даних (таких як сигнали, що надходять від гідролокатора);

– діагностика несправностей (як у технічних системах, так і в людському організмі);

– структурний аналіз складних об'єктів (наприклад, хімічних сполук);

– вибір конфігурації складних багатокомпонентних систем (наприклад, розподілених комп'ютерних систем);

– планування послідовності виконання операцій, що приводять до заданої цілі (наприклад, виконувані промисловими роботами) [4, 5].

Чіткого формального визначення експертної системи, яке всіх би задовольнило, не існує - приведене вище теж досить розпливчасте. Але проте існує досить багато важливих ознак, властивих тією чи іншою мірою всім експертним системам.

1.2 Аналіз систем аналогів

Аналогами нашої технології є:

– Мала експертна система. Програма є простою експертною системою, що використовує байєсовську систему логічного висновку.

Вона призначена для проведення консультації з користувачем у будь-якій прикладній області (на яку налаштована завантажена база знань) з метою визначення ймовірностей можливих результатів та використовує для цього оцінку правдоподібності деяких передумов, що отримується від користувача [6].

Як приклад розглянемо завдання визначення ймовірності наявності різних захворювань у пацієнта. Програма в даному випадку виступає в ролі лікаря (експерта), який ставить пацієнтові питання щодо симптомів та на основі отриманих відомостей ставить діагноз. Причому бажано не мучити пацієнта зайвими питаннями, а ставити лише найважливіші, від відповіді які переважно залежить остаточне встановлення хвороби. Саме так і вчиняє ця експертна система. Вона запитує користувача оцінку істинності найважливішого свідчення, на основі відповіді коригує ймовірності результатів і переходить до наступного свідчення, вибравши знову найактуальніше. Таким чином досягається якнайшвидше отримання результату при мінімальній кількості запитів [6].

Використання байесівської системи логічного висновку означає, що інформація, що обробляється експертною системою, не є абсолютно точною, а має імовірнісний характер. Користувач не обов'язково повинен бути впевнений в абсолютній істинності чи хибності свідчення, він може відповідати на запити системи з певним ступенем впевненості. У свою чергу, система видає результати консультації у вигляді ймовірностей настання результатів.

В ході дослідження було реалізовано власну базу знань з предметної області «Вибір смартфону». Розроблена база знань у редакторі баз знань системи „Мала експертна система” зображена на рисунку 1.1.

Запитання:

– Ви багато коштів виділяєте на придбання телефону?;

– Ви багато часу проводите за телефоном?;
– Ви багато файлів/додатків зберігаєте на телефоні?;
– Ви часто робите фотографії?;
– Ви часто граєте в ігри на телефоні?;
– Ви любите власноруч налаштовувати більшість функціоналу телефону?.

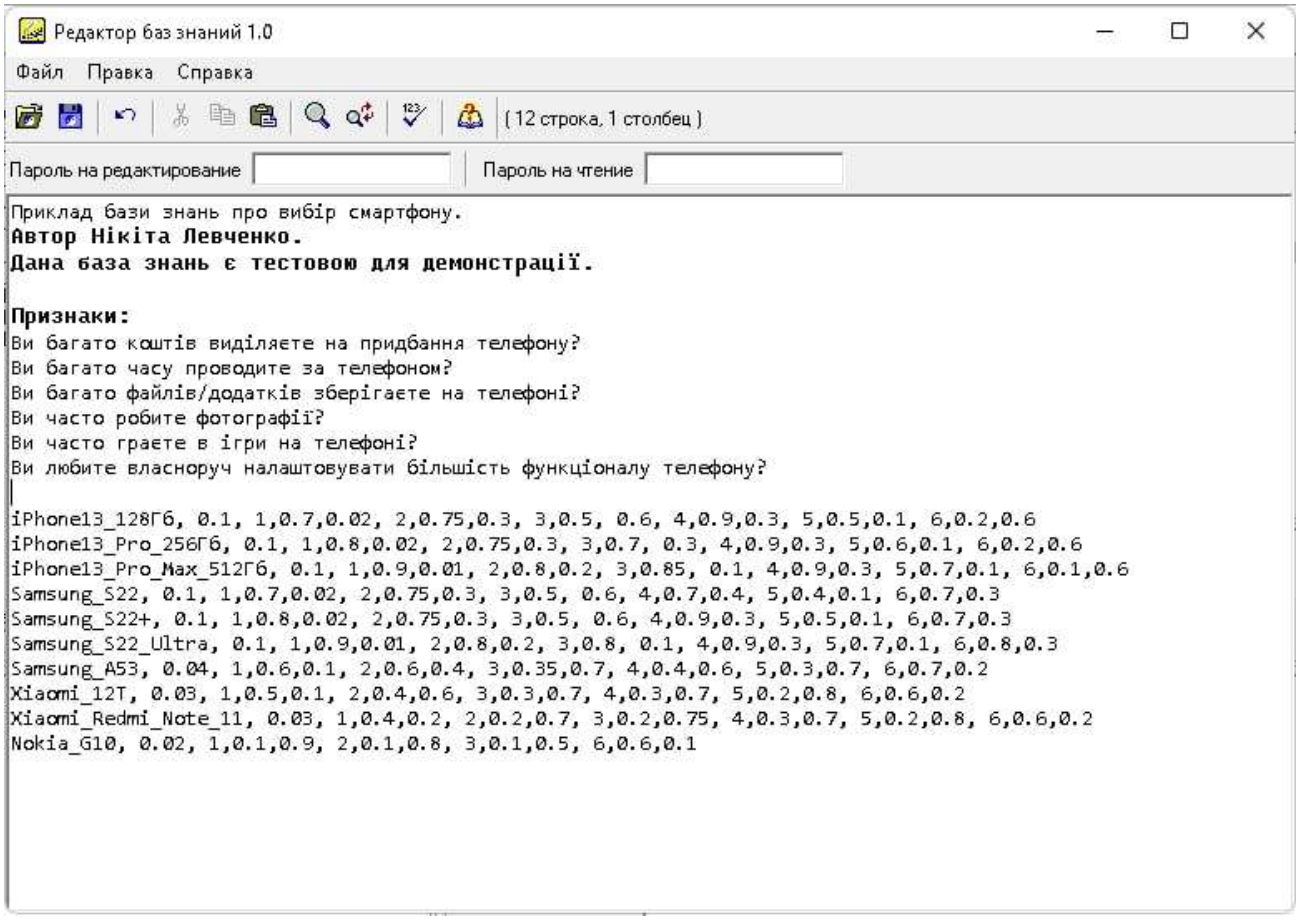


Рисунок 1.1 – База знань у редакторі баз знань системи
„Мала експертна система ”

Результат опитування у системі „Мала експертна система” згідно реалізованої бази знань зображено на рисунку 1.2

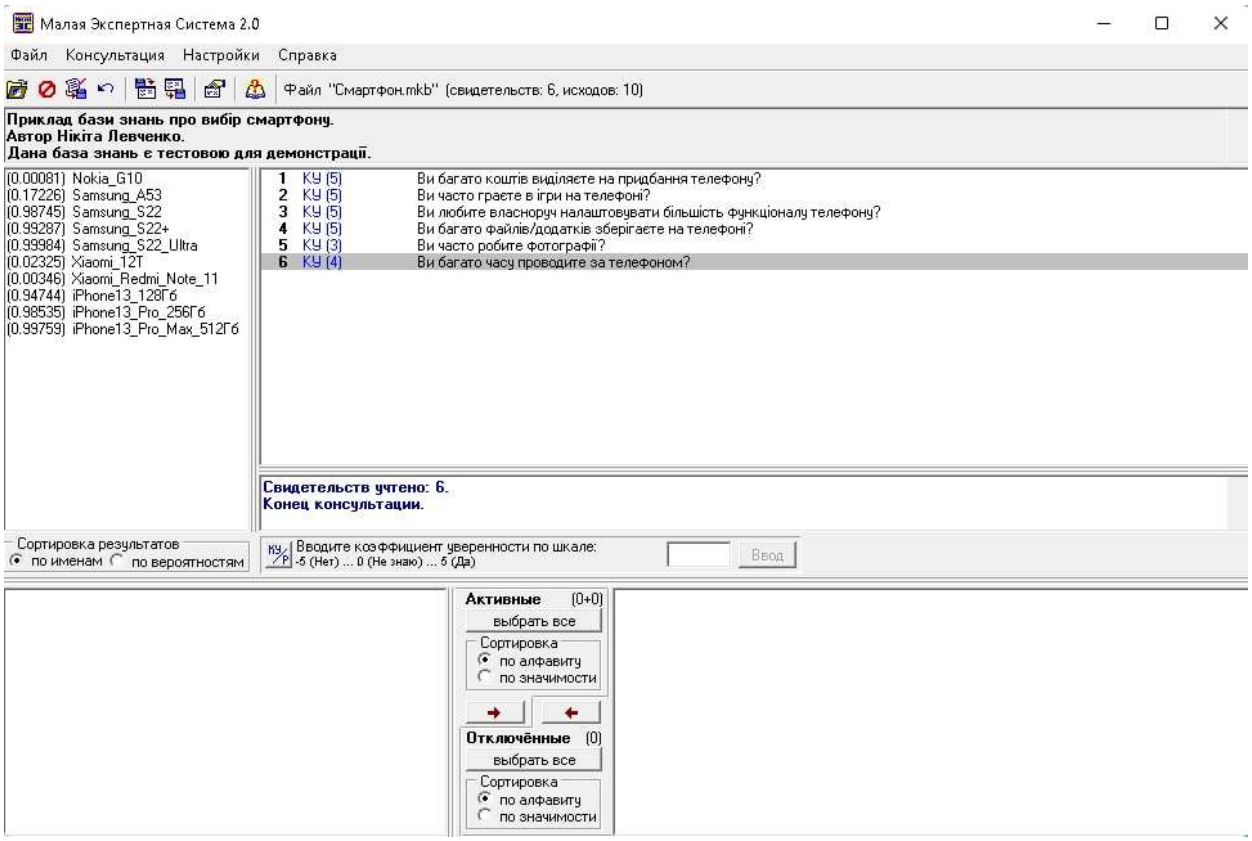


Рисунок 1.2 – Результат опитування у системі „Мала експертна система ”

Важливим плюсом програми «Мала Експертна Система» можна назвати можливість створення, редагування та використання власної бази знань. Щоб полегшити це завдання, було написано Редактор баз знань 1.0, що постачається разом із системою. [6]

Недоліком даної технології є те, що вона не доступна у веб-версії;

– Експертна система „Decision Support Expert::Shell” продукційного типу призначена для одержання консультації або рішення конкретної проблеми, яку важко формалізувати, з будь-якої предметної області, знання для якої можна виразити за допомогою правил-продукцій [7].

Відповідно до концепцій проектування і функціонування продукційної експертної системи, вона складається з оболонки і бази знань (БЗ).

База знань складається з правил, фактів, дозволених значень з антецедентів (лівих частин правил), а також питань для цих дозволених значень [7].

Оболонка складається з механізму логічного висновку, підсистеми пояснень отриманого результату, інтерфейсу з користувачем. Інтерфейс користувача включає введення/коректування вхідних даних, включаючи дані бази знань, підсистему поповнення знань [7].

Таким чином, до складу оболонки продукційної експертної системи „Decision Support Expert::Shell” входять:

- база знань;
- механізм логічного виведення;
- інтерфейс користувача;
- підсистема пояснень;
- підсистема поповнення знань;
- синтаксичний аналізатор;
- підсистема введення/коректування даних.

В ході дослідження було реалізовано власну базу знань з предметної області «Транспорт».

Після завантаження новоствореної БЗ до експертної системи, починаємо проводити процес опитування, що підтверджує працездатність та відповідність тематиці даної бази знань. Запитання:

- Розпочати опитування?;
- В яке місто Ви хотіли б дістатися?;
- Скільки коштів Ви виділяєте на поїздку?;
- Скільки людей подорожуватиме?;
- Скільки у Вас часу на поїздку?;
- Вам необхідне спальне місце?.

Результат опитування у системі „Decision Support Expert::Shell” згідно реалізованої бази знань зображено на рисунку 1.3

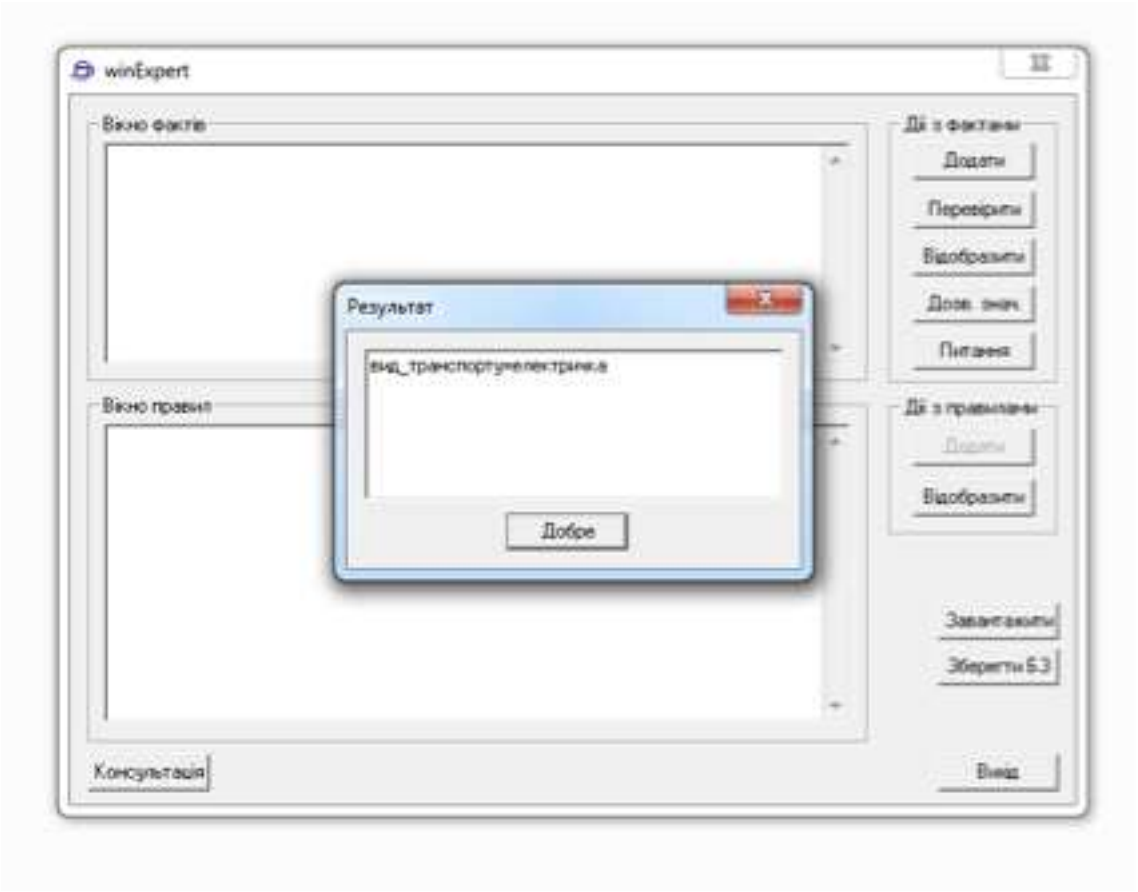


Рисунок 1.3 – Результат опитування у системі „Decision Support Expert::Shell”

„Decision Support Expert::Shell” має візуальний інтерфейс, який не відповідає сучасним критеріям дизайну програмних продуктів. Заповнення бази знань відбувається дуже не зручним способом – заповнення текстового документу правилами вручну і для цього користувачу потрібно знати окремий синтаксис даної оболонки. Використовується застарілий підхід створення експертних систем, а саме продукційна модель подання знань. Також дана система не має можливості працювати на інших операційних платформах.

– Gimet – веб-орієнтована програмна платформа для створення ЕС. В даній платформі застосовано семантичну, а також продукційну моделі подання знань [8].

Основними модулями веб-орієнтованої платформи для створення експертних систем «GIMET» є такі: модуль авторизації, модуль логічного введення-виведення, модуль створення та наповнення бази знань, сховище баз знань, модуль консультації, інтерфейс користувача (безпосередньо графічний інтерфейс). Модуль

авторизації відповідає за створення облікового запису користувача його особистого кабінету та можливість користування системою відповідно до отриманих повноважень (рольовий підхід). Модуль консультації відповідає за створення консультаційної сесії між користувачем та веб-ресурсом із спеціалізованою експертною системою [8].

Недоліком даної технології є те, що вона має недостатній функціонал.

Разом з тим, аналіз ринку програмних продуктів переконливо свідчить про чітку тенденцію до значного поширення веб-орієнтованих засобів як складних та масштабних програмних комплексів, що пов’язано із стрімким розвитком технологій, практичністю та економічною доцільністю їх застосування, ефективним застосуванням ресурсів. Зважаючи на вищевказане, актуальною є задача використання сучасних програмних технологій у сфері розробки спеціалізованих експертних систем, зокрема, розробки веб-орієнтованої програмної платформи для створення ЕС, що забезпечить пришвидшення процесу створення ЕС, надійність функціонування, зручність для користувача, тощо. Метою досліджень є підвищення ефективності роботи експертів, інженерів зі знань та програмістів при розробці спеціалізованих ЕС, а також користувачів при використанні ЕС.

Найперспективнішим аналогом є веб-орієнтованої платформи для створення експертних систем «GIMET», тому що вона є єдиним WEB-орієнтованим аналогом.

1.3 Аналіз об’єкту проектування WEB - орієнтованої інформаційної технології для розробки експертних систем

Основною метою розробки WEB - орієнтованої інформаційної технології для розробки експертних систем є надання користувачеві можливості зручного створення бази знань будь-якої складності та подальше її використання для надання користувачу рішення у поставленій проблемі.

Вимоги до WEB - орієнтованої інформаційної технології:

- реєстрація користувачів та створення облікових записів;
- наявність персонального кабінету;

– можливість користувачу створювати власті експертні системи;

– можливість збереження баз знань у хмарних сервісах.

Додаток створений як веб-застосунок – розподілений застосунок, в якому клієнтом виступає браузер, а сервером – веб-сервер. Так як клієнти можуть підключатись до системи з мобільних пристроїв та з своїх домашніх комп'ютерів або ноутбуків, потужність яких може бути незадовільною для виконання складних обчислень та обробки великої кількості даних, а в страхуванні по іншому не виходить, було прийняте рішення розробки «тонкого клієнта». Це означає що всі обчислення зосереджуються на веб-сервері.

На серверах EC2 встановлена операційна система Linux, що дуже схожа на образ CentOS7.

Варто розуміти що підключення до серверу відбувається завдяки протоколу SSH, тому всі зміни та налаштування відбуваються за допомогою терміналу

Для зберігання файлів баз знань використовується Amazon Simple Storage Service.

Amazon Simple Storage Service (Amazon S3) – це сервіс зберігання об'єктів, що пропонує найкращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних. Клієнти будь-якої величини та з будь-якої промислової галузі можуть зберігати та захищати необхідний обсяг даних для практично будь-якого прикладу використання. Наприклад, для озер даних, хмарних додатків та мобільних додатків. Вигідні класи сховища та прості у використанні інструменти адміністрування дозволяють оптимізувати витрати, організувати дані та точно налаштувати обмеження доступу відповідно до потреб бізнесу чи законодавчих вимог [9].

Для зберігання решти даних використовується Amazon RDS PostgreSQL.

PostgreSQL стала найкращою реляційною базою даних з відкритим вихідним кодом для багатьох корпоративних розробників і стартапів, яка працює над провідними бізнес-програмами та мобільними додатками. Amazon RDS дозволяє легко налаштовувати, керувати та масштабувати розгортання PostgreSQL у хмарі. За допомогою Amazon RDS ви можете за лічені хвилини розгортати масштабовані

розгортання PostgreSQL з економічно ефективним апаратним забезпеченням із змінним розміром. Amazon RDS керує складними та трудомісткими адміністративними завданнями, такими як встановлення та оновлення програмного забезпечення PostgreSQL; управління зберіганням; реплікація для високої доступності та пропускної здатності читання; і резервні копії для аварійного відновлення [10].

Amazon RDS для PostgreSQL надає вам доступ до можливостей знайомої системи баз даних PostgreSQL. Це означає, що код, програми та інструменти, які ви вже використовуєте сьогодні у своїх існуючих базах даних, можна використовувати з Amazon RDS. Наразі Amazon RDS для PostgreSQL підтримує PostgreSQL 9.6, 10, 11, 12, 13 і 14 [10].

Розглянемо два основних підходи при розробці серверної частини додатку. На верхньому рівні SOAP обмежує структури ваших повідомлень, тоді як REST — це архітектурний підхід, орієнтований використання HTTP як транспортного протоколу. Специфіка SOAP – це формат обміну даними. З SOAP це завжди SOAP-XML, який є XML, що включає:

– Envelope (конверт) – кореневий елемент, який визначає повідомлення та простір імен, використаний у документі;

– Header (заголовок) – містить атрибути повідомлення, наприклад: інформація про безпеку або мережеву маршрутизацію;

– Body (тіло) – містить повідомлення, яким обмінюються програми;

– Fault – необов'язковий елемент, який надає інформацію про помилки, що сталися під час обробки повідомлень. І запит, і відповідь мають відповідати структурі SOAP [11].

Специфіка REST — використання HTTP як транспортний протокол. Він має на увазі найкраще використання функцій, що надаються HTTP - методи запитів, заголовки запитів, відповіді, заголовки відповідей і т.д.

Формат обміну повідомленнями. У SOAP ви використовуєте формат SOAP XML для запитів та відповідей. У REST такого фіксованого формату немає. Ви можете обмінюватись повідомленнями на основі XML, JSON або будь-якого іншого

зручного формату. JSON є найпопулярнішим серед використовуваних форматів [11].

Визначення послуг. SOAP використовує Web Services Description Language (WSDL) — мову опису веб-сервісів та доступу до них, засновану на мові XML. REST немає стандартної мови визначення сервісу. Незважаючи на те, що WADL був одним із перших запропонованих стандартів, він не дуже популярний. Більш популярним є використання Swagger або Open API [11].

Транспорт. SOAP не накладає жодних обмежень на тип транспортного протоколу. Ви можете використовувати або протокол HTTP, або MQ. REST має на увазі найкраще використання транспортного протоколу HTTP [11].

Простота реалізації. RESTful веб-сервіси, як правило, набагато простіше реалізувати, ніж веб-сервіси на основі SOAP. REST зазвичай використовує JSON, який легше аналізувати та обробляти. На додаток до цього, REST не потребує визначення служби для надання веб-служби. Однак у випадку SOAP вам необхідно визначити свій сервіс з використанням WSDL, і при обробці та аналізі повідомлень SOAP-XML виникають великі витрати.

Враховуючи вищевказані переваги і недоліки будемо використовувати REST архітектуру.

Дана задача є актуальною, зважаючи на прогрес у розвитку інтелектуальних комп’ютерних технологій які забезпечують вирішення специфічних завдань (консультування, навчання, діагностування, тестування, проектування тощо). Вхідними даними для розробки експертної системи можуть будь-які дані.

Вимоги до серверної частини технології:

- операційна система – Linux;
- хостинг сервіс – AWS EC2;
- база даних – PostgreSQL;
- протокол взаємодії з клієнтом – HTTP;
- швидкість відповіді на будь-який запит <1000 мс;
- мова програмування Node.js;

– фреймворк Express.js.

1.4 Висновок до розділу 1

В даному розділі було проаналізовано предметну область у сфері надання рекомендацій експертними системами.

Здійснено аналіз технологій аналогів на прикладі Expert Shell, Мала експертна система, Gimet та проведено поверхневий порівняльний аналіз. Визначено вимоги до WEB-орієнтованої інформаційної технології.

Здійснено аналіз об’єкту проектування WEB - орієнтованої інформаційної технології для розробки експертних систем. Також здійснено аналіз різних серверних архітектур та обґрунтовано вибір для даної задачі REST архітектури

Виходячи з попередніх аналізів предметної області надання рекомендацій, програмних засобів для створення баз знань, дало можливість визначити основні проблеми та недоліки. Визначено критичні негативні параметри у програм аналогів та поставлені цілі для їх усунення.

2 АНАЛІЗ МОДЕЛЕЙ НАДАННЯ РЕКОМЕНДАЦІЙ ТА ПРОЕКТУВАННЯ WEB-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ

2.1 Порівняльний аналіз систем надання рекомендацій

Основні сфери застосування систем надання рекомендацій пов'язані з підтримкою прийняття управлінських рішень у таких напрямках бізнесу, як кредитування й оцінка ризиків, маркетинговий аналіз, прогнозування фінансових ринків, моделювання функціональних складових менеджменту (фінанси, виробництво, людські ресурси), розв'язання прикладних соціологічних задач (моделі формування і зміни рейтингів політиків), управління бюджетними ресурсами і економічне моделювання, виявлення незаконного використання кредитних карток. Наприклад, в банківській сфері прийняття рішень стосовно надання кредиту може прийматися комп'ютеризовано на основі обробки даних клієнтів [12,13].

Системи підтримки прийняття рішень (СППР) — інформаційні системи, які використовують обладнання, програмне забезпечення, дані, базу моделей і роботу менеджера з метою підтримки всіх стадій прийняття рішень у процесі аналітичного моделювання. Програмні засоби включають комплекс різних алгоритмів підтримки рішень, базу моделей, базу даних, допоміжні та керівну програми, яка забезпечує процес прийняття рішень з урахуванням специфіки проблеми. Орієнтовані на операційне управління СППР застосовуються для виконання науково-дослідних робіт, в управлінні кадрами, виробництвом тощо [12,13].

Експертні системи - це клас комп’ютерних програм, які пропонують рекомендації, проводять аналіз, виконують класифікацію, дають консультації і ставлять діагноз. Вони орієнтовані на розв’язування задач, вирішення яких вимагає проведення експертизи людиною-спеціалістом. На відміну від програм, що використовують процедурний аналіз, експертні системи розв’язують проблеми у

вузькій предметній площині (конкретній ділянці експертизи) на основі логічних міркувань. Такі системи часто можуть знайти розв’язок задач, які неструктуровані і неточно визначені. Вони через використання евристик компенсують відсутність структурованості, що корисно в ситуаціях, коли недостатня кількість необхідних даних або часу виключає можливість здійснення повного аналізу [14].

Основою експертної системи є сукупність знань, яка структурується для спрощення процесу прийняття рішення. Для спеціалістів в галузі штучного інтелекту термін “знання“ означає інформацію, що потрібна програмі для того, щоб вона вела себе інтелектуально. Ця інформація приймає форму фактів або правил. Факти і правила не завжди правдиві або неправильні, інколи існує деяка міра неправильності в достовірності факту або точності правила. Якщо сумнів виражається явно, то він називається коефіцієнтом впевненості [14].

Рекомендаційні системи — підклас системи фільтрації інформації, яка будує рейтинговий перелік об'єктів (фільми, музика, книги, новини, веб-сайти), яким користувач може надати перевагу. Для цього використовується інформація з профілю користувача. Рекомендаційні системи порівнюють однотипні дані від різних людей і розраховують список рекомендацій для конкретного користувача. Деякі приклади їх комерційного та некомерційного використання наведені в статті про колаборативну фільтрацію. Для розрахунку рекомендацій використовується граф інтересів. Рекомендаційні системи — зручна альтернатива пошуковим алгоритмам, оскільки дозволяють виявити об'єкти, які не можуть бути знайдені останніми. Цікаво, що рекомендаційні системи часто використовують пошукові машини для індексації незвичайних даних [14].

Експертна система відрізняється від інших прикладних програм наявністю наступних ознак. Моделює не стільки фізичну (або іншу) природу певної проблемної області, скільки механізм мислення людину стосовно до розв'язку завдань у цій проблемній області. Це суттєво відрізняє експертні системи від систем математичного моделювання або комп'ютерної анімації. Не можна, звичайно, сказати, що програма повністю відтворює психологічну модель фахівця в цій предметній області (експерта), але важливо, що основна увага все-таки приділяється

відтворенню комп'ютерними засобами методики рішення проблем, яка застосовується експертом, -тобто виконанню деякої частини завдань так само (або навіть краще), як це робить експерт [14].

Система, крім виконання обчислювальних операцій, формує певні міркування й висновки, ґрунтуючись на тих знаннях, якими вона розташовує. Знання в системі представлені, як правило, на деякій спеціальній мові й зберігаються окремо від власне програмного коду, який і формує висновки й міркування. Цей компонент програми прийнято називати базою знань.

2.2 Аналіз моделей, методів і технологій надання рекомендацій

Розглянемо основні підходи до створення рекомендаційних систем. В класичних рекомендаційних системах найбільш поширеними є контент-орієнтована та колаборативна. Схема вищезгаданих підходів показана на рисунку 2.1.

Рисунок 2.1 - Схема контент-орієнтованої та колаборативної фільтрацій

Контент-орієнтована фільтрація. Суть цього підходу полягає в тому, що ми зіставляємо користувачів з контентом або товарами, які їм подобалися або були ними

куплені. Тут важливі атрибути користувачів та продуктів. Наприклад, для рекомендацій до фільмів ми використовуємо такі ознаки як режисер, актори, тривалість фільму, жанр і т.д., щоб знайти схожість між фільмами. Крім того, ми можемо отримати такі характеристики, як оцінка настроїв та оцінки TF-IDF з описів фільмів та оглядів. (Оцінка TF-IDF відображає, наскільки важливим є слово для документа в наборі документів). Мета контент-орієнтованих методів – створити «профіль» для кожного користувача та кожного предмета [15].

Розглянемо приклад рекомендації новинних статей користувачам. Припустимо, ми маємо 100 статей і словник розміру N. Спочатку ми обчислюємо оцінку TF-IDF для кожного слова в кожній статті. Потім ми будуємо 2 вектори:

Вектор предмета: це вектор довжини N. Він містить значення 1 для слів, які мають високу оцінку TF-IDF у цій статті, інакше значення 0. Приклад векторів показано на рисунку 2.2 [15].

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	kings	lazy	love	quick
0	0.00	0.00	0.60	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.49	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.00

Рисунок 2.2 - Приклад двух векторів предмета

Вектор користувача: знов вектор розмірністю 1xN. Для кожного слова ми зберігаємо можливість появи слова в статтях, які вжив користувач. Зверніть увагу, що вектор користувача ґрунтується на атрибутах елемента (у даному випадку це оцінка слів TF-IDF) [15].

Побудувавши ці «профілі», ми обчислюємо схожість між користувачами та предметами. Предмети повинні бути рекомендовані користувачеві, якщо: 1) вони мають найбільшу схожість з користувачем або 2) мають велику подібність до інших елементів, прочитаних користувачем. Є кілька способів зробити це. Давайте подивимося на 2 поширені методи:

Косинусна схожість, формула (2.1):

$$\text{cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}, \quad (2.1)$$

Щоб рекомендувати предмети, які найбільш схожі на ті, якими цікавився користувач, ми обчислюємо косинусну схожість між статтями, які прочитав користувач, та іншими статтями. Найбільш схожі будуть рекомендовані. Отже, це подоба предмет-предмет [15].

Косинусна схожість найкраще підходить, коли ваші ознаки є високорозмірними, особливо в області пошуку інформації та аналізу тексту.

Щоб обчислити схожість між користувачем та предметом, ми просто беремо косинусну схожість між вектором користувача та вектором предмета.

Подібність Жаккара, також відоме як перетин над об'єднанням, формула (2.2).

$$\text{J}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}, \quad (2.2)$$

Використовується для подібності предмет-предмет. Ми порівнюємо вектори елементів один з одним та повертаємо найбільш схожі предмети [15].

Подібність Жакара корисна лише тоді, коли вектори містять бінарні значення. Якщо у них є ранжування або рейтинги, які можуть набувати більше двох можливих значень, подібність Жаккара не застосовується [15].

Сучасні рекомендаційні системи можна класифікувати за методами фільтрації, а саме:

– колаборативна (спільна) фільтрація;
– контентна (вмісту) фільтрація.

Колаборативна фільтрація (collaborative filtering) використовує відомі оцінки групи користувачів для прогнозування невідомих переваг іншого активного

користувача. Ці методи відповідно створюють рекомендації тільки на основі виявлених попередніх взаємодій між користувачами та елементами. Кожна нова рекомендація залежить від знань про поведінку користувача. Вважається, що даних про минулі взаємодії достатньо для виявлення подібних користувачів та/або подібних елементів і відповідно здійснення прогнозів на основі оцінок встановленої подібності [16].

А методи колаборативної фільтрації містять три основних підходи:

– підхід, що ґрунтується на даних (memory/heuristic-based);
– підхід, що ґрунтується на моделях (model-based);
– гібридний підхід.

Ключовою перевагою підходів колаборативної фільтрації є те, що вони зовсім не покладаються на аналіз вмісту елементів чи характеристики користувачів. Для пошуку подібності, вироблення рекомендацій не потрібно опрацьовувати сам елемент, щоб зрозуміти його. Чим активніше користувачі взаємодіють з елементами, тим точнішими стають нові рекомендації.

Контентна фільтрація (content-based filtering, CBF) використовує інформацію про властивості предметів. Такими властивостями, наприклад, для фільмів, можуть бути жанр, кіностудія, провідний актор, режисер тощо. Ідея контентної фільтрації полягає у тому, що предметам із подібним контентом користувачі надають подібні переваги. Фільтрацію вмісту можна використати у таких системах, де наперед передбачається наявність описових даних [16].

Фільтрація за демографічними показниками (демографічна фільтрація, Demographic filtering) – надає рекомендації на основі демографічного (наприклад, віку, професії) профілю користувача [16].

Рекомендовані продукти можна створювати для різних демографічних ніш, поєднуючи рейтинги користувачів у цих нішах.

Фільтрація на основі знань (Knowledge-based) – пропонує продукти на основі висновків про потреби та вподобання користувачів, вибір предметів та основу для рекомендацій [16].

Переваги:

– незалежність користувача – не залежить від інших користувачів;
– нові елементи можуть бути легко включені;
– прозорість – пояснення, зрозумілі.

Крім колаборативної та контент-орієнтованої досить поширеними також є інші методи фільтрації. Зокрема, важливим класом систем надання рекомендацій є експертні системи .

Експертні системи орієнтовані на вирішення широкого кола завдань в неформалізованих областях, рішення задачі розпізнавання образів у таких областях передбачає складання описів об'єктів і правил, що визначають по цих описів приналежність об'єктів до тих чи інших класах. Процедури застосування таких правил до будь-яких об'єктів в експертних системах підпорядковуються різним стратегіям. Найбільш часто застосовуються стратегії прямого чи зворотного виведення [17,18].

Пряме виведення є одним із двох основних методів міркування при використанні правил виведення (в галузі штучного інтелекту) і можуть бути описані логічно, як повторне застосування modus ponens – Якщо - то. Пряме виведення - популярна стратегія міркування експертних систем, системи правил для бізнесу і виробництва. Протилежним до методу прямого виведення є метод зворотного виведення [17,18].

Метод прямого виведення починає з наявних даних і використовує правила виведення для отримання додаткових даних (наприклад, з користувача), доки мета не буде досягнута. Механізм логічного виведення, що використовує пряме виведення, шукає серед правил виведення перше правило, у якого антецедант (частина якщо) набув логічного значення "істина". Якщо таке правило знайдено, то можна перейти до виведення його висновку (частина тоді) і додавання нової інформації до даних механізму виведення [17,18].

Механізм виведення буде ітераційно повторювати цей процес, доки мета не буде досягнута.

Оскільки саме дані визначають, які правила будуть вибирані і використані, цей метод відноситься до методів керованих даними, на відміну від зворотного виведення, що відноситься до методів, керованих метою. Пряме виведення часто використовується в експертних системах, таких як CLIPS.

Однією з переваг методу прямого виведення над зворотнім є те, що прийом нових даних може призвести до отримання нових висновків. Це робить механізм виведення краще пристосованим до динамічних ситуацій, в яких умови, швидше за все, зміняться [17,18].

Зворотне виведення (або зворотне міркування) є метод отримання висновку, який працює в зворотному напрямку від мети. Він використовується в автоматизованому доведенні теорем, машинному виведенні та інших напрямках штучного інтелекту, також спостерігається у приматів [17,18].

В теорії ігор застосування зворотного виведення для під-ігор, з метою знайти рішення гри, називається зворотною індукцією. У шахах даний метод називається ретроспективним аналізом, він використовується для створення баз таблиць для ендшпіля у комп'ютерних шахах.

Зворотне виведення здійснюється у логічному програмуванні за допомогою SLD-резолюції. Обидва правила базуються на правилі виведення modus ponens. Це один з двох найбільш часто використовуваних методів міркування при роботі з правилами виведення та логічними наслідками, є протилежним до прямого виведення. Системи зворотного виведення, наприклад, Пролог, зазвичай використовують стратегію пошуку в глибину [17,18].

Зворотне виведення починається з переліку цілей (або гіпотез) і працює в зворотному напрямку від висновку до антецеденту, щоб побачити, чи доступні дані, які будуть підтримувати будь-який з цих висновків.

Механізм логічного виведення, що використовує зворотне виведення, шукає серед правил виведення перше правило, у якого висновок (частина Тоді) відповідає поставленій меті. Якщо невідомо, чи набуває антецедент (частина Якщо) цього правила логічного значення "істина", тоді антецедент цього правила додається до

списку цілей. (для того, щоб мета підтвердилася, необхідно також отримати дані для підтвердження цього нового правила) [17,18].

Зверніть увагу, що цілям завжди відповідають висновки, у яких пізніше антецеденти розглядаються як нова мета. Зрештою, антецедентам повинні відповідати відомі факти (вони, як правило, визначаються як висновки, у яких завжди істинний антецедент). Таким чином, правилом виведення, яке використовується є modus ponens [17,18].

Оскільки саме список цілей визначає, які правила вибирати і використовувати, цей метод називається методом, керованим метою, на відміну від прямого виведення, що є методом, керованим даними. Зворотне виведення часто використовується в експертних системах.

Процес створення експертних систем значно змінився за останні роки. Завдяки появі спеціальних інструментальних засобів побудови експертних систем значно скоротились терміни та зменшилась трудомісткість їх розроблення. Одним із інструментальних засобів, що використовуються під час створення експертних систем є мови програмування, орієнтовані на їх створення.

Мови програмування, такі як Пролог, Knowledge Machine і Eclipse підтримують метод зворотного виведення в своїх механізмах виведення [19].

Управляючий компонент (інтерпретатор продукцій). Цей компонент визначає порядок застосування правил і визначає, чи є ще факти, які можуть бути змінені у разі продовження консультації. Управляючий компонент виконує чотири функції:

1. Зіставлення – зразок правила порівнюється з наявними фактами.

2. Вибір – якщо в конкретній ситуації можуть бути застосовані декілька правил, то з них вибирається одне, найбільш відповідне за заданим критерієм.

3. Активізація – якщо зразок правила при зіставленні збігається з будь-якими фактами з робочої пам'яті, то правило активізується.

4. Дія – робоча пам'ять змінюється, в неї додаються висновки правила, що активізувалось. Якщо в правій частині правила міститься вказання на будь-яку дію, то воно виконується (як наприклад в системах забезпечення безпеки інформації) [20].

Інтерпретатор продукції працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити серед них ті, умови яких збігаються з відомими на даний момент фактами з робочої пам'яті. Інтерпретатор визначає порядок застосування правил. Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, потім цикл повторюється спочатку. У одному циклі може спрацювати тільки одне правило. Якщо декілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, яке і спрацює в даному циклі. Інформація з робочої пам'яті послідовно зіставляється з умовами правил для виявлення збігу. Сукупність відображених правил складає так звану конфліктну множину. Для вирішення конфлікту інтерпретатор має критерій, за допомогою якого він вибирає єдине правило, після чого воно спрацьовує. Спрацьовування полягає в занесенні фактів, з яких слідує висновок правила, в робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо у висновок правила входить назва будь-якої дії, то воно виконується, наприклад, подається звуковий сигнал, запускається двигун, виконується процедура і т.д. [20]. На рисунку 2.3 наведена схема, що ілюструє цикл роботи інтерпретатора.

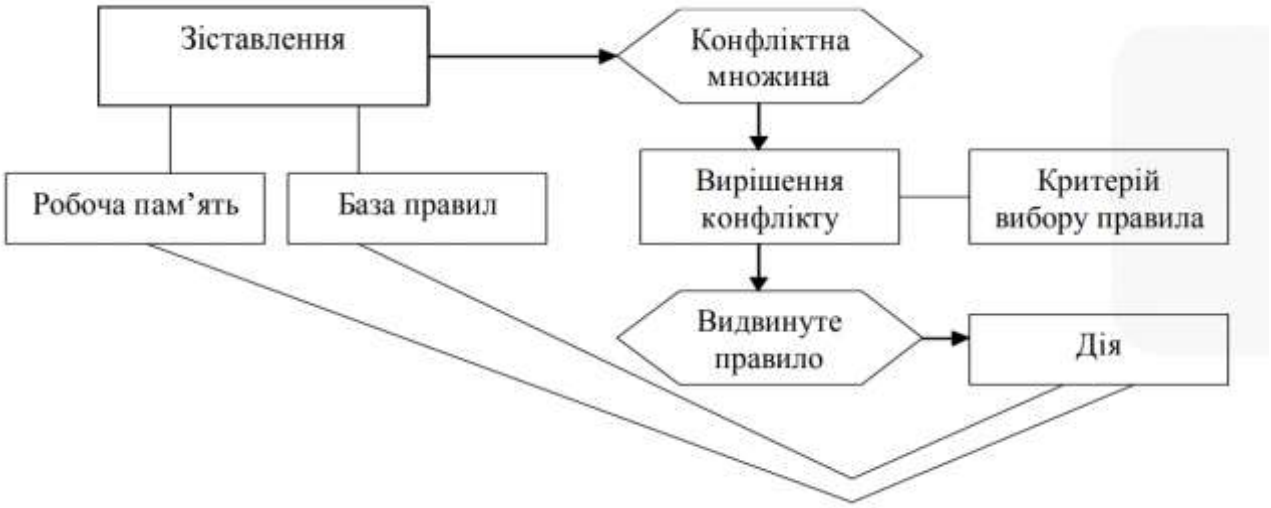


Рисунок 2.3 – Схема циклу роботи інтерпретатора

Нові дані, введені в систему правилом, що спрацювало, можуть в свою чергу змінити критерій вибору правила.

2.3 Структурна організація серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем

А тепер розглянемо класичну структурну організацію ЕС. Однією з основних структурних організацій є продукційна модель.

Продукційна модель – це така модель, яка основана на правилах, що дозволяють подати знання у вигляді речення типу: ЯКЩО (умова), ТО (дія), або системами з висновком, що використовує зіставлення за зразком. Під умовою розуміється деяке речення-зразок, за яким здійснюється пошук в базі знань, а під дією – дії, що виконуються при успішному результаті пошуку (вони можуть бути проміжними, які виступають далі як умови, і термінальними або цільовими, які завершають роботу системи). Обчислювальний процес в продукційних системах істотно відрізняється від класичної фоннеймановської схеми, прийнятої в більшості комп'ютерів, в якій і програми, і дані при обчисленнях знаходяться в одній області пам'яті. Програма може перетворювати саму себе як дані, змінюючи тим самим послідовність своїх дій [4].

У противагу цьому продукційна система майже не має процедурних компонентів і практично повністю управляється даними, тобто є дескриптивною.

Продукційна система включає три основних складових: базу правил, робочу пам'ять і механізм виведення. Для підтримки роботи системи і реалізації інтелектуальної взаємодії з користувачем в неї входять і підсистема надбання знань, і засоби спілкування на природною мовою, і підсистема пояснень. Структурна схема експертної системи зображена на рисунку 2.4 [4].

Рисунок 2.4 - Структурна схема експертної системи

Розглянемо структурну організацію серверної частини WEB – орієнтованої інформаційної технології для розробки експертних систем. Узагальнену структурну схему WEB - орієнтованої інформаційної технології для розробки експертних систем зображено на рисунку 2.5.

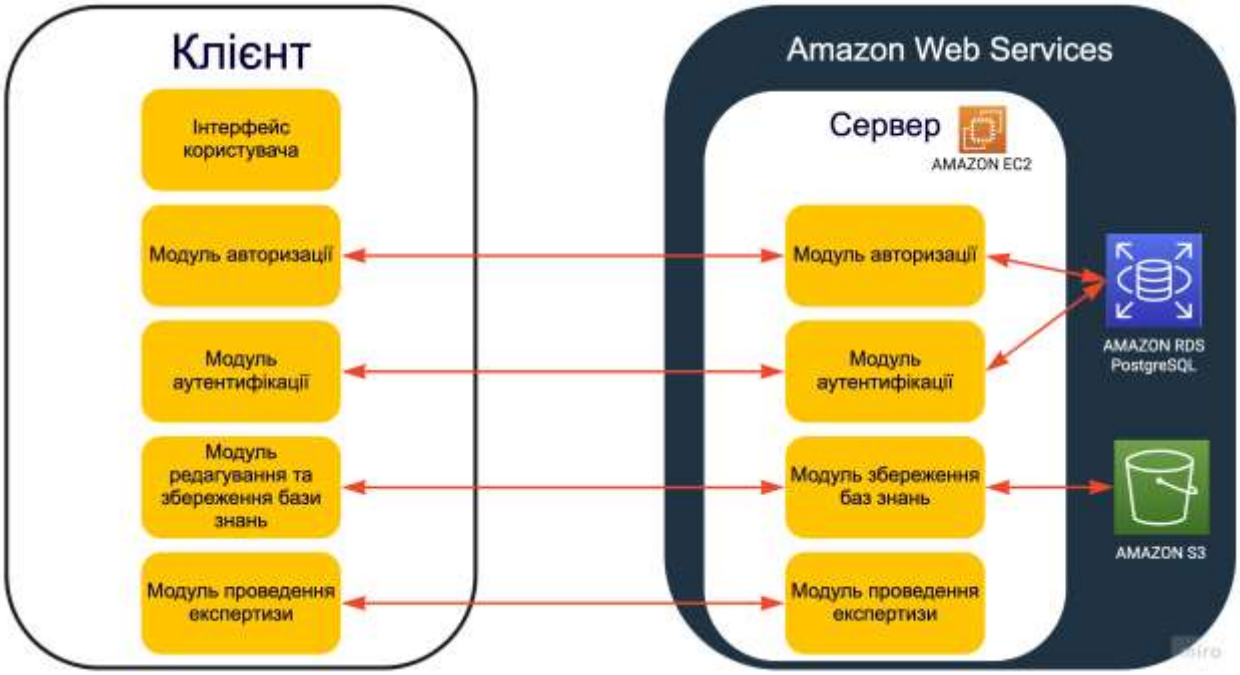


Рисунок 2.5 - Узагальнена структурна схема WEB - орієнтованої інформаційної технології для розробки експертних систем

Клієнт та сервер спілкуються шляхом обміну окремими повідомленнями. Повідомлення, надіслані клієнтом, називаються запитами, а повідомлення, надіслані сервером на запит, називаються відповідями.

HTTP - це протокол без збереження стану, тобто сервер не зберігає ніяких даних під час обміну повідомленнями. Кожен запит (англ. Request) відправляється на сервер, який аналізує його і повертає відповідь (англ. Response) [21].

2.4 Проектування серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем

Розглянемо діаграму послідовності взаємодії клієнта і серверу, що зображена на рисунку 2.6.

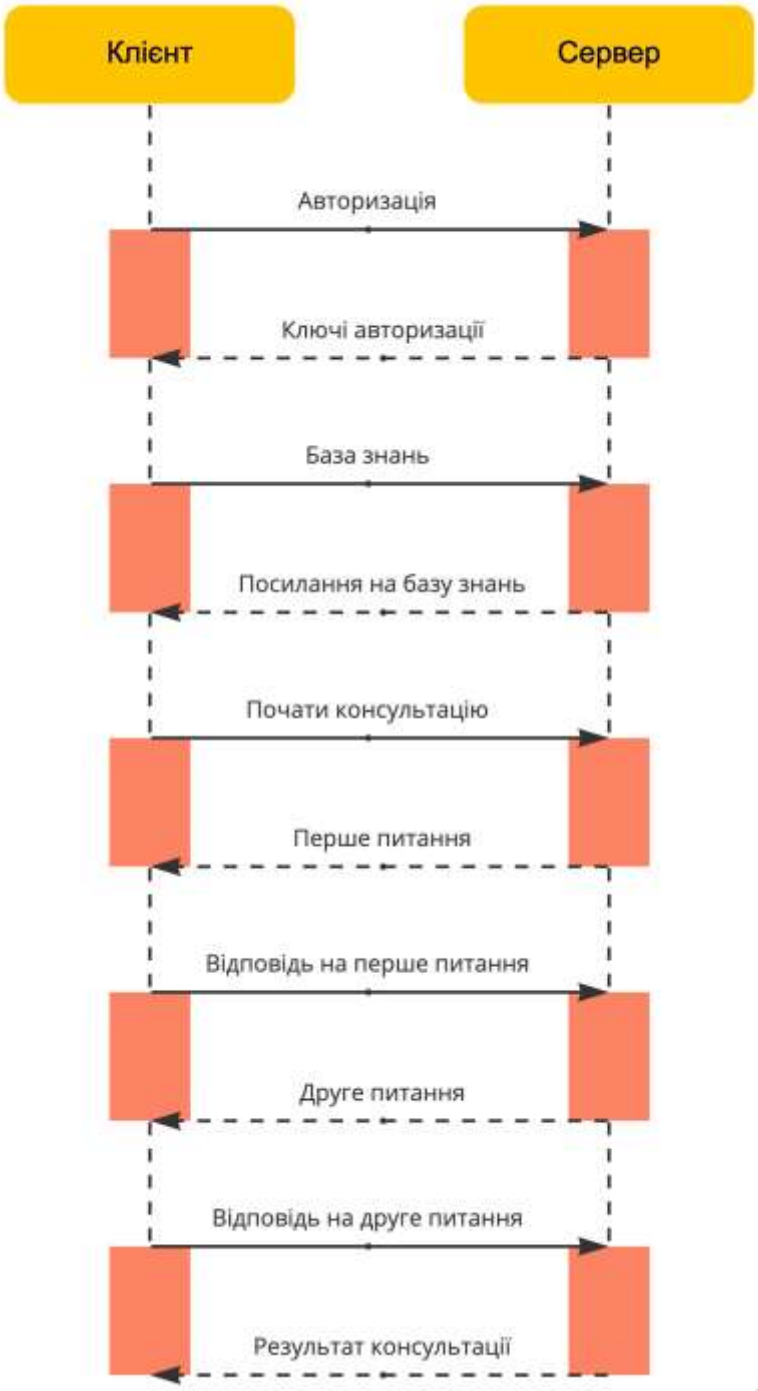


Рисунок 2.6 - Діаграма послідовності взаємодії клієнта і серверу

У якості серверу використовується Amazon EC2 інстанс.

Інстанс у хмарних обчисленнях – це серверний ресурс, що надається сторонніми хмарними сервісами [22].

Сервіс Amazon EC2 пропонує широкий вибір типів інстансів, оптимізованих для різних прикладів використання. Типи інстансів включають різні комбінації таких компонентів, як ЦПУ, пам'ять, сховище та мережеві можливості, що дозволяє вибрати відповідний набір ресурсів для програм. Кожен тип інстансу включає один або кілька розмірів інстансів, що дозволяє масштабувати ресурси відповідно до вимог цільового робочого навантаження [23].

Інстанси Amazon EC2 T2 – це інстанси з продуктивністю, що підвищується, які забезпечують базовий рівень продуктивності ЦПУ з можливістю його підвищення [23].

Безлімітні інстанси T2 можуть підтримувати високу продуктивність ЦПУ до того часу, поки цього вимагає робоче навантаження. Більшість робочих навантажень загального призначення безлімітні інстанси T2 забезпечують достатню продуктивність без додаткової плати. Якщо потрібно виділити більше ресурсів ЦПУ на інстанс протягом тривалого часу, це можна зробити з оплатою за фіксованим тарифом: 5 центів за годину роботи віртуального ЦПУ [23].

Базовий рівень продуктивності та можливість його перевищення визначаються кредитами ЦПУ. Інстанси T2 регулярно одержують певну кількість кредитів, яка залежить від розміру інстансу. Вони накопичують кредити ЦПУ під час простою та споживають їх в активному стані. Інстанси T2 добре підходять для різних робочих навантажень загального призначення, включаючи мікросервіси, інтерактивні програми з низькою затримкою, малі та середні бази даних, віртуальні робочі столи, середовища розробки, складання та тестування, репозиторії коду та прототипи продуктів [23].

Можливості:

– Процесор Intel Xeon Scalable із частотою до 3,3 ГГц (Haswell E5-2676 v3 або Broadwell E5-2686 v4);

– Високочастотні процесори Intel Xeon;

– Постійний базовий рівень продуктивності та ЦП з можливістю підвищення продуктивності (регулюється кредитами ЦП);

– Економічний тип інстанс загального призначення. Доступний на рівні безкоштовного користування;

– Збалансоване співвідношення обчислювальних, мережевих ресурсів та ресурсів пам'яті.

Основною метою розробки платформи для створення експертних систем є надання користувачеві можливості зручного створення бази знань будь-якої складності та подальше її використання для надання користувачу рішення у поставленій проблемі.

Вимоги до серверної частини:

– Мова програмування Node.js;

– Фреймворк Express.js;

– Швидкодія відповіді серверу <1000 мс;

– Можливість реєстрації користувачів та створення облікових записів;

– Можливість користувачу створювати власті експертні системи;

– Доступність ЕС для проходження консультації будь-яким користувачем.

2.5 Висновок до розділу 2

Отже, в 2 розділі наведено порівняльний аналіз систем надання рекомендацій, зокрема розглянуто найбільш поширенні системи надання рекомендацій, а саме системи підтримки прийняття рішень, експертні системи, рекомендаційні системи. Серед них найбільш перспективними для даної розробки є експертні системи.

Здійснено аналіз моделей, методів і технологій надання рекомендацій - зокрема розглянуто основні підходи до створення рекомендаційних систем, а саме контент-орієнтована та колаборативна фільтрації та експертні системи.

Розглянуто структурну організацію WEB - орієнтованої інформаційної технології для розробки експертних систем, зокрема серверну частину.

Виконано проектування та визначено основні вимоги серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ WEB СЕРЕДОВИЩА ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ

3.1 Обґрунтування вибору середовища для створення WEB - орієнтованої інформаційної технології для розробки експертних систем

При виборі середовища розробки для веб-додатку імітаційного моделювання обчислювальних систем розглянуто наступні середовища:

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з передустановленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безоплатно.WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг [24].

Основні можливості:

- Модифікація файлів .css, .html, .js з одночасним переглядом результатів (Live Edit, в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»);
- Підтримка HTML5;
- Підтримка JSDoc;
- Підтримка Node.js;

– Можливості Zen Coding і Emmet;

– Зневадження коду на JavaScript;

– Віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації;

– Інтеграція з системами управління версіями Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю побудови списку змін і відкладених змін;

– Інтеграція з системами відстеження помилок.

У версіях від WebStorm 5 для CoffeeScript є навігація за кодом, автодоповнення, рефакторинг, підсвічування синтаксису і перевірка на помилки [24].

Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для розробників на мові Java, засоби для управління сирцевими кодами, візуальні побудовники GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плаґінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP [25].

Випущена на умовах Eclipse Public License, Eclipse є вільним програмним забезпеченням. Він став одним з перших IDE під GNU Classpath і без проблем працює під IcedTea [25].

Eclipse це фреймворк для розробки модульних кросплатформових застосунків із низкою особливостей:

– можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);

– крос-платформова;

– модульна, призначена для подальшого розширення незалежним розробниками;

– з відкритим сирцевим кодом;

– розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland [25].

Eclipse написана на Java, тому є платформо-незалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, яка розробляється окремо для більшості поширених платформ. Бібліотека SWT використовує графічні засоби платформи (ОС), що забезпечує швидкість і звичний зовнішній вигляд інтерфейсу користувача [25].

Гнучкість Eclipse забезпечується за рахунок модулів, що підключаються, завдяки чому можлива розробка не тільки на Java, але і на інших мовах, таких як C/C++, Perl, Groovy, Ruby, Python, PHP, ErLang та інших.

Для середовища Eclipse існує цілий ряд вільних і комерційних модулів. Спочатку середовище було розроблене для мови Java, але в нині існують численні розширення для підтримки інших мов, як наприклад:

– C/C++ — CDT Eclipse's C/C++ Development Tooling(англ.);

– Perl — модуль EPIC, Eclipse Perl Integration(англ.);

– PHP — PDT PHP Development Tools(англ.);

– JavaScript — JSEclipse Javascript plugin for the Eclipse environment(англ.);

– Python — Pydev, Python Development Environment(англ.);

– Ruby — RDT, Ruby Development Tools(англ.) [25].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи

додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server) [26].

Розглянувши усі наведені середовища розробки було прийнято рішення використовувати IntelliJ WEBStorm. В цьому середовищі інтегровані Git, Gulp, Mocha які використовуються в розробці веб-додатку для імітаційного моделювання обчислювальних систем. Тому, це середовище розробки є найбільш зручним.

3.2 Обґрунтування вибору мови програмування

Для створення серверної частини програми була обрана платформа Node.js, яка являє собою середовище виконання коду на JavaScript. Вона реалізована на основі рушія JavaScript Chrome V8, котрий перетворює виклики на мові JavaScript в машинний код, без необхідності його інтерпретувати [27].

Дана платформа була обрана з наступних причин. По-перше, вона дозволяє швидко перейти від frontend розробки, тобто розробки клієнт частини, до backend розробки, тобто розробки серверної частини, при цьому не вивчаючи нових технологій, оскільки використовується одна мова програмування – JavaScript [27].

По-друге, в Node.js є її пакетна екосистема npm, яка є найбільшою екосистемою бібліотек з відкритим вихідним кодом. npm постачає пакети, які можна встановити в проект і вони вирішують більшість відомих і частих проблем, тим самим роблячи розробку більш швидкою та ефективною [27].

Також Node.js дозволяє написати власні модулі, тобто незалежні частини програми, які можна використовувати в подальшому в інших проектах. Окрім цього, вона має набір вбудованих модулів, котрі можна використовувати без будьякої ініціалізації. Можливості Node.js можна легко розширити використовуючи

різноманітні бібліотеки та фреймворки, яких на разі існує досить багато. Найбільш популярні серед них: Koa, Socket.io, Micro, Next.js, Meteor, Express [27].

Ще одними важливими перевагами Node.js є її асинхронність і подієво-орієнтований підхід, який на відміну від потокового програмування засновано на якихось подіях. Це можуть бути дії користувача, або, наприклад, подія надходження в програму мережевого пакету тощо. Схематично дана модель зображена на рисунку 3.1. В даній моделі один потік виконання обробляє події (наприклад, новий HTTP запит), що надходять в чергу, одна за одною, виконуючи відповідні обробники подій [27].

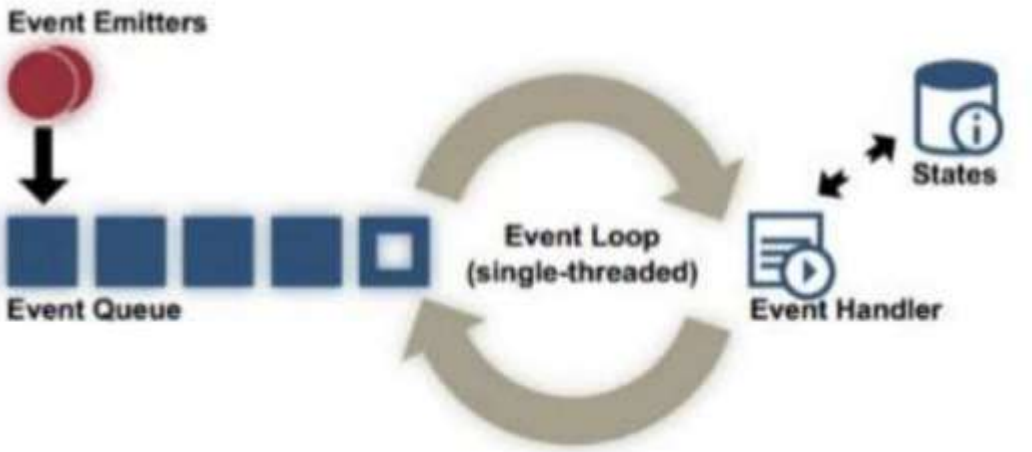


Рисунок 3.1 – Схема подієво-орієнтованої архітектури Node.js

Архітектура середовища виконання складається з таких компонентів:

– рушій для мови JavaScript Chrome V8;

– бібліотеки для крос-платформенного введення-виведення та багатопотоковості libuv;

– бібліотеки для DNS (c-ares);

– бібліотеки для криптографії OpenSSL;

– бібліотеки для компресії даних zlib [27].

Схематично дану архітектуру зображено на рисунку 3.2

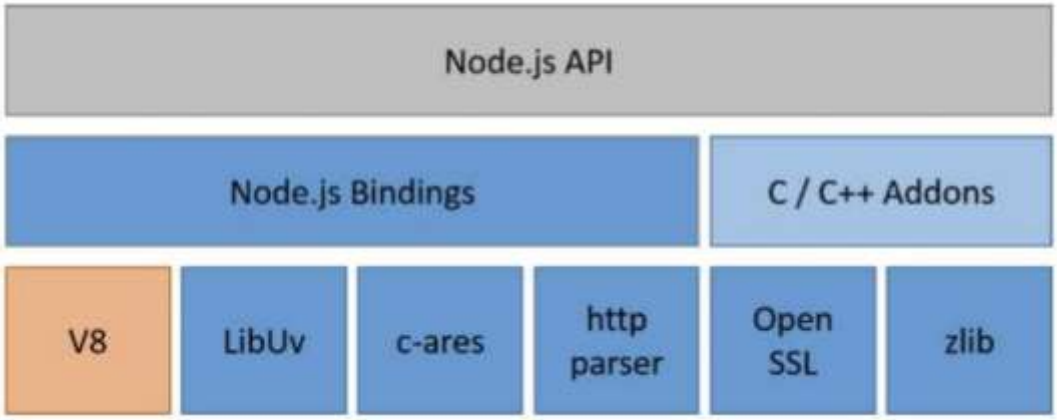


Рисунок 3.2– Схема архітектури програмного середовища виконання Node.js

В якості фреймворку було обрано Express.js, так як він швидко освоюється і вважається одним із найпростіших фреймворків Node.js. До того ж він використовується досить давно, при цьому не втрачаючи своєї популярності.

Особливістю даного фреймворку можна визначати невеликий обсяг базового функціоналу. Таким чином, розробник отримує в своє розпорядження легкий та швидкий інструмент, який за необхідності можна розширити та розвинути за рахунок зовнішніх модулів підключених до проекту. При цьому немає ніяких обмежень, ні кількісних, ні функціональних.

Тож в результаті, даний фреймворк надає розробнику можливість бути вільним у виборі засобів вирішення тієї чи іншої проблеми. Хоча це і має певний мінус, оскільки передбачає більше роботи з боку програміста при виборі і організації модулів, але такий підхід фактично означає, що кожне застосування буде унікальним, оскільки немає універсальних рішень.

3.3 Розробка алгоритму роботи програмного продукту

Процес реєстрації / авторизації користувача. У даній частині у користувача є можливість створити акаунт за його поштою для користування системою або вказати дані для авторизації у існуючий акаунт. Включає в собі перевірку наявності користувача з переданими даними авторизації у базі даних та його створення або

авторизації. Схема алгоритму процесу реєстрації / авторизації користувача показана на рисунку 3.3.

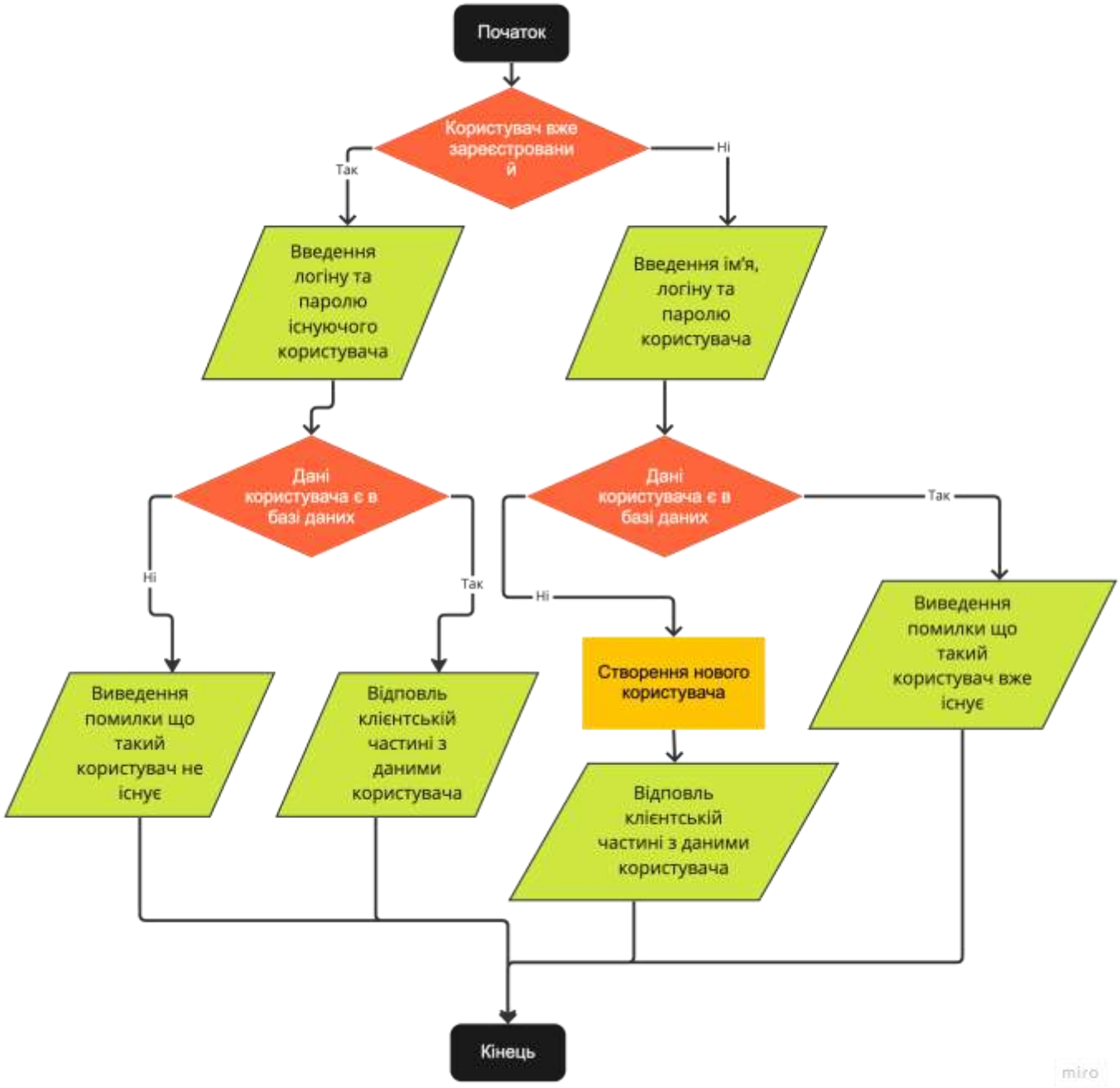


Рисунок 3.3 – Схема алгоритму процесу реєстрації / авторизації користувача

Процес створення / завантаження бази знань. У цій частині у користувача є можливість створити новий файл бази знань за допомогою графічного інтерфейсу, завантажити файл бази знань з його персонального пристрою або редагувати вже існуючу у базі даних базу знань. Після редагування у користувача також є можливість зберегти зміни бази знань або експортувати файл на персональний пристрій для

редагування у інших текстових редакторах. Схема алгоритму процесу створення / завантаження бази знань показана на рисунку 3.4.

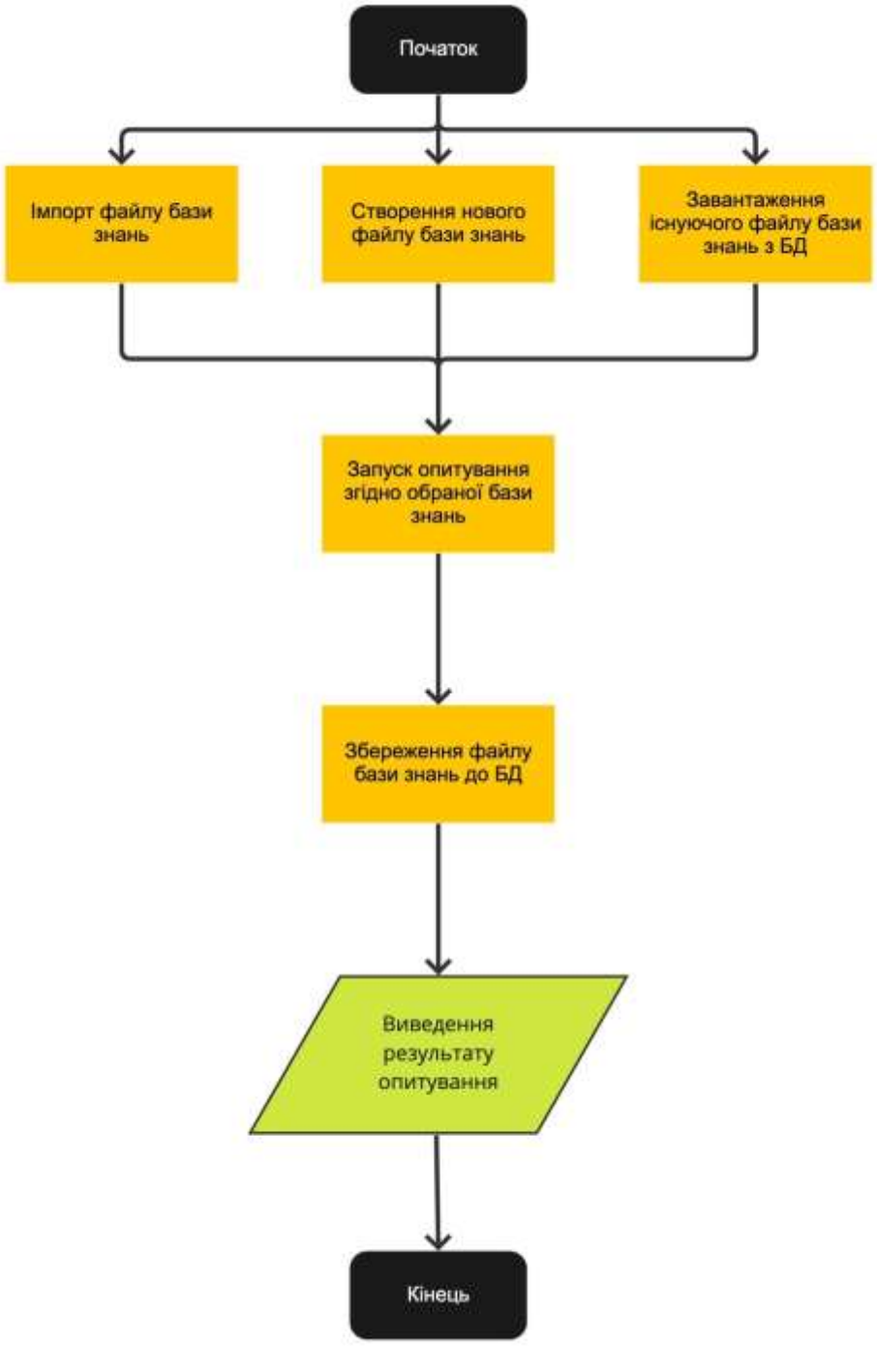


Рисунок 3.4 – Схема алгоритму процесу створення / завантаження бази знань

Процес опитування користувача. Для кращого розуміння процесу обробки бази знань була розроблена блок-схема алгоритму роботи програми (рис. 3.5). Даний алгоритм показує головний процес роботи з базою знань.

Наведений алгоритм починається з надання програмі початкових даних які ініціалізують початок виконання консультаційної сесії з експертом. Дані являють собою базу знань (набір питань та варіантів відповідей та можливих результатів експертизи). Користувач, що проходить консультацію має можливість обрати відповідь лише з наданих у базі знань варіантів, тому є досить важливим, щоб база знань, створена інженером знань, була побудована таким чином щоб спростити пошук відповіді при експертизі. На кожній ітерації циклу він бере по одному уточнюючому запитанню та якщо умови запитання не суперечать зібраним відповідям виводить на екран питання та відповіді на нього. Далі очікується вибір користувача та обробляється натискання кнопки з обраною відповіддю. Інакше, якщо умови запитання конфліктують з уже зібраними висновками - питання просто ігнорується.

Після отримання чергової відповіді від користувача система зберігає її у висновок та перевіряє чи немає результату, що підходить за вже зібраними висновками опитування. Алгоритм повторює дану ітерацію доки не знайде відповіть за висновками у базі знань або у ній не закінчаться питання. Якщо результату не знайдено – алгоритм переходить до наступної ітерації циклу. Якщо питання закінчились - виводиться повідомлення що результату за введеними відповідями немає.

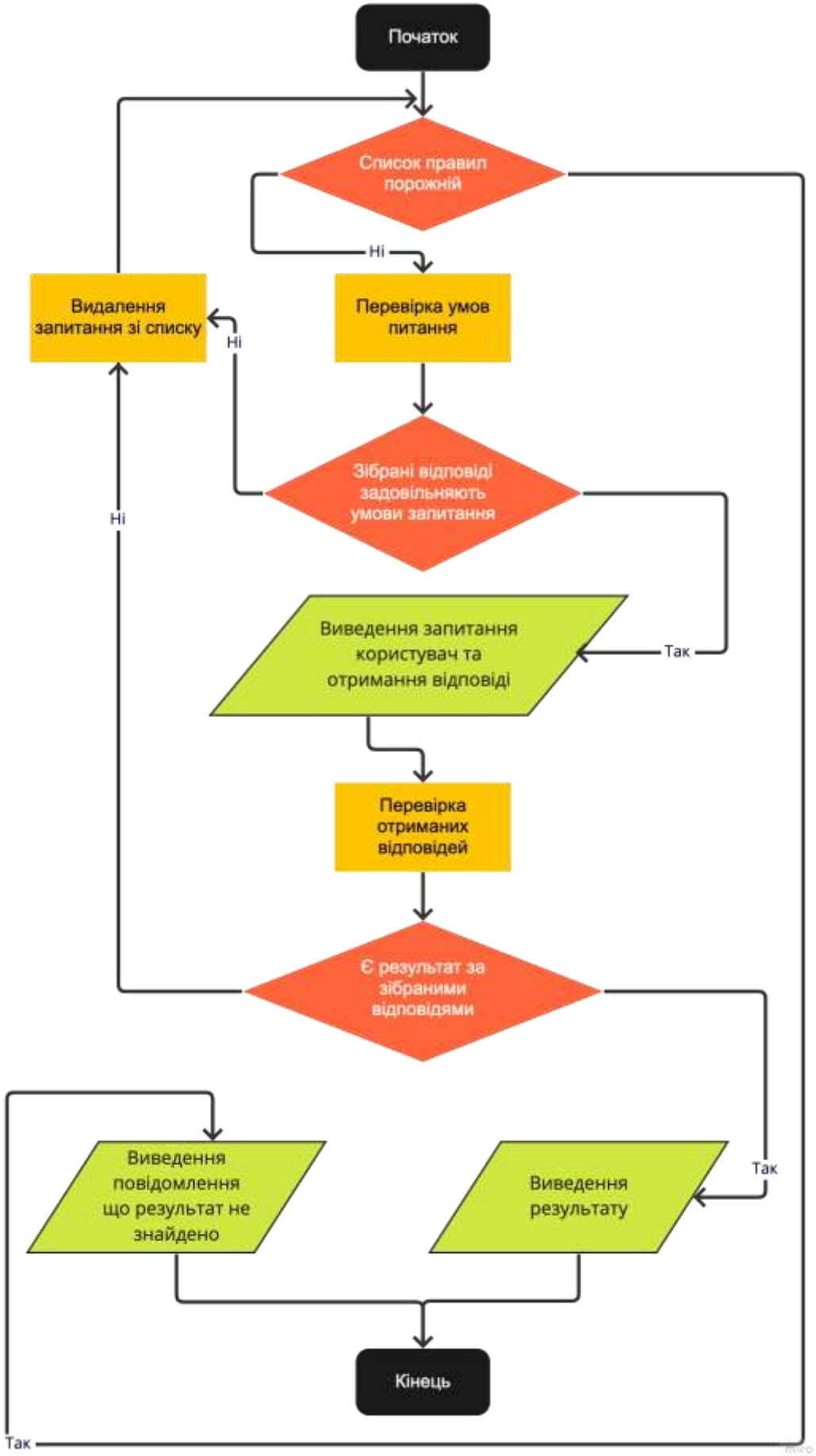


Рисунок 3.5 – Схема алгоритму процесу опитування користувача

3.4 Тестування програмного продукту і аналіз результатів

Сервісно-орієнтована серверна архітектура поширена серед сучасних технологічних компаній і вимагає надійного API для підтримки зростаючих бізнес-вимог.

Як результат, якісне тестування API є дуже важливим для забезпечення швидкої розробки без втрати впевненості в якості коду.

Одним із чудових інструментів для допомоги як у ручному, так і в автоматичному тестуванні API є Postman — спільна платформа для тестування та розробки API [28].

Postman — це програма, яка полегшує тестування API і надає:

– Зручний інтерфейс для взаємодії з API;

– Можливість налаштування тестових даних, створення HTTP-запиту з настроюваним корисним навантаженням і встановлення значень у відповіді;

– Запустити і налаштувати колекцію запитів за допомогою вбудованого функціоналу або через командний рядок.

Postman дозволяє як ручним, так і автоматизованим інженерам із забезпечення якості створювати та підтримувати структуру тестування API.

Хоча Postman має свої обмеження, він також надає численні переваги:

– Легко налаштувати : завантажте та тестуйте API за лічені хвилини;

– Мінімальне знання API : UI Postman робить взаємодію з API легкою та простою;

– Спільна робота : можливість створювати, публікувати та діліться своєю колекцією з членами команди. Подібно до проектів з відкритим вихідним кодом, наборами тестів Postman можна ділитися та робити внески іншим [28].

Ми створимо загальні «сценарії користувача» — тести типових шляхів, якими користуються користувачі нашого API, і додамо твердження для очікуваних даних щодо цих відповідей API.

Крім того, у життєвому циклі розробки програмного забезпечення команди Agile наша колекція Postman може служити іншим цілям, окрім простих сценаріїв користувача:

– Автоматизуйте регресію API;

– Надайте набір інструментів API для допомоги інженерам під час розробки;

– Інтеграція з CI/CD (Jenkins).

Тепер ми напишемо спеціальний набір тестів, який дозволить нам запитувати дану кінцеву точку з бажаним корисним навантаженням і підтверджувати дані у відповіді JSON.

Щоб досягти цього, ми будемо редагувати такі елементи кожного запиту Postman:

– Сценарій: унікальне ім’я, опис поведінки користувача;

– Сценарій попереднього запиту: розділ для налаштування тестових даних для запиту (Postman використовує JavaScript);

– Тести: Твердження, які мають бути зроблені щодо відповіді API; У Postman є кілька готових параметрів, які полегшують звичайні твердження.

Рисунок 3.6 – Приклад запиту Postman

Коли у нас локально завантажено Postman і ми розуміємо, як мають бути структуровані запити Postman (у нашому випадку вони називаються «сценарії користувача»), ми можемо розпочати створення нашої першої колекції — набору запитів, які налаштовують тестові дані та підтверджують значення відповіді JSON .

Спочатку ми хочемо зрозуміти серверні ресурси, які ми хочемо перевірити — нижче наведено тестовий REST API, який ми будемо використовувати для тестування нашої колекції.

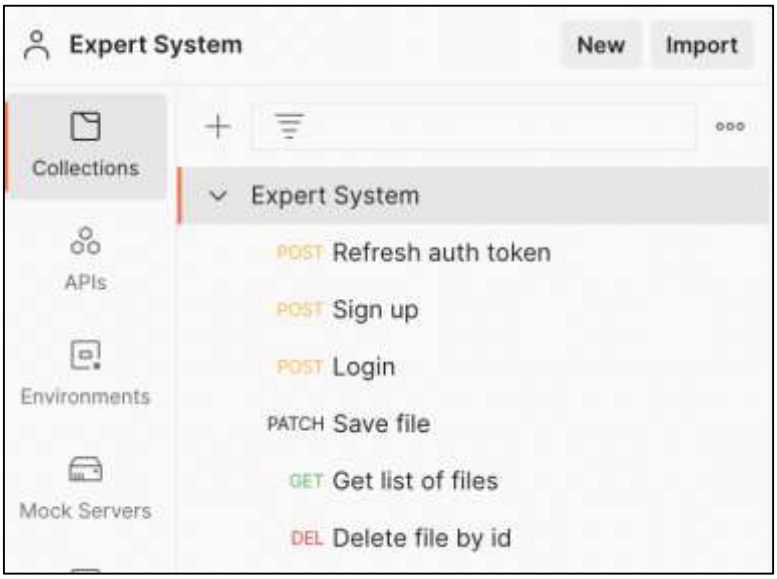


Рисунок 3.7 – Кінцеві точки

Коли кінцеві точки відомі, згідно з рисунку 3.7, ми хочемо структурувати їх таким чином, щоб відображати справжню взаємодію з користувачем.

У нашому випадку ми структуруємо їх таким чином:

Сценарій 1: як користувач, я хочу авторизуватись та запитувати список баз знань, що мені належать та завантажити одну з них для редагування. Приклад створеного сценарію показано на рисунку 3.8;

Попередній скрипт: немає;

Тести:

Підтвердити, що всі кінцеві точки повертають 200;

Підтвердити, що всі кінцеві точки виконались швидше за 1000 мс;

Приклад реалізованого скрипту для автоматизованого тестування показано на рисунку 3.9.

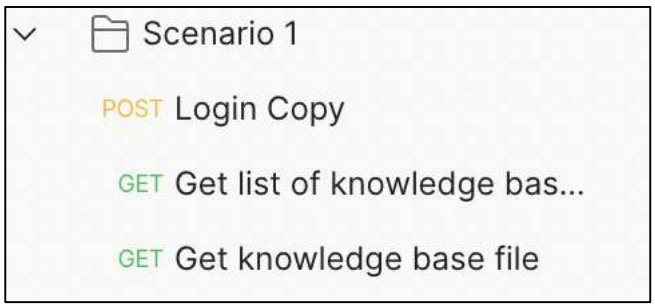


Рисунок 3.8 – Приклад створеного сценарію

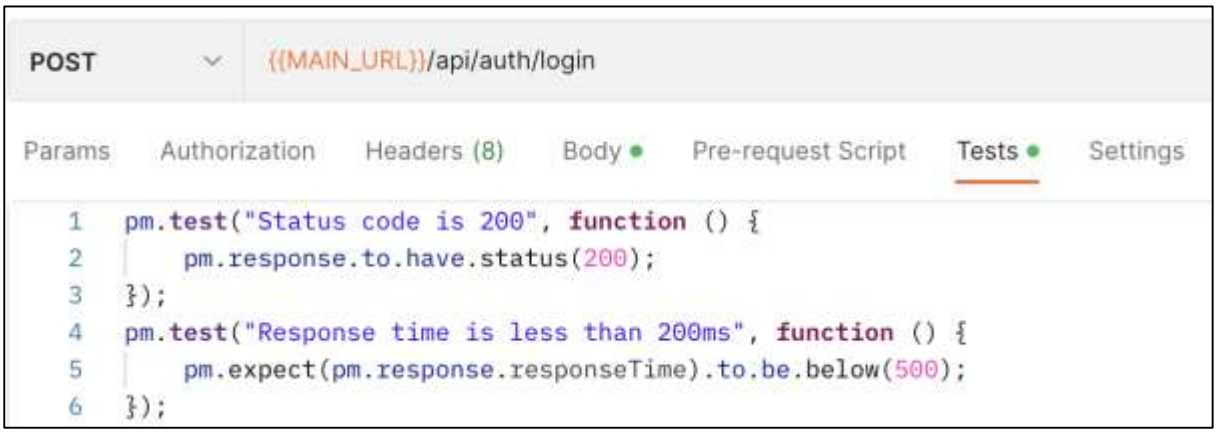


Рисунок 3.8 – Приклад реалізованого скрипту для автоматизованого тестування

Далі за допомогою інтерфейсу запускаємо тестування сценарію та отримуємо звіт по кожному з запитів. Приклад звіту автоматизованого тестування зображено на рисунку 3.9.

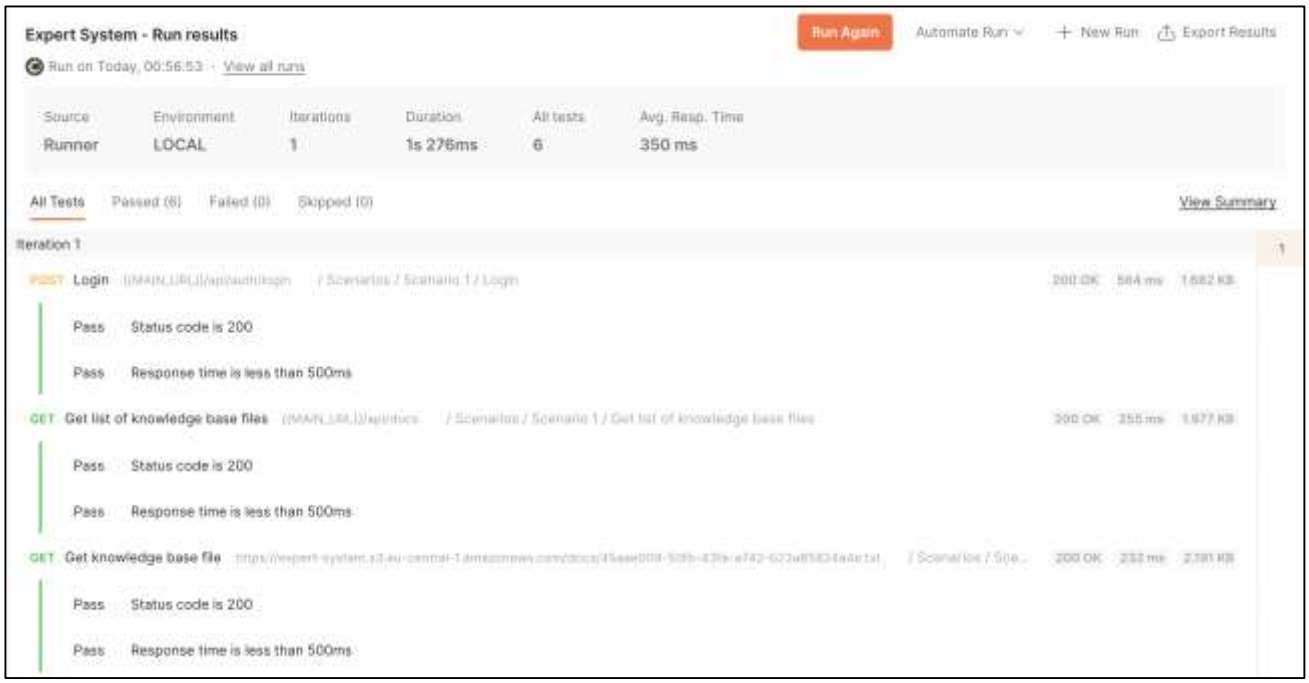


Рисунок 3.9 – Приклад звіту автоматизованого тестування

Середня швидкість відповіді серверу аналогу 420 мс. Отже на основі результатів автоматизованого тестування констатуємо, що середнє значення відповіді серверу становить 350 мс, що свідчить про покращення швидкодії відповіді сервера порівняно із системою аналогом.

3.5 Висновок до розділу 3

В даному розділі було розглянуто популярні середовища розробки, зокрема JetBrains WebStorm, Eclipse та Visual Studio. Обґрунтовано вибір середовища для створення серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем. Обґрунтовано вибір мови програмування та фреймворку.

Розроблено алгоритм роботи програмного продукту. Визначено основні алгоритми: алгоритм процесу реєстрації / авторизації користувача, алгоритм процесу створення / завантаження бази знань, алгоритм процесу опитування користувача. Також наведено схеми основних алгоритмів роботи.

Виконано автоматизацію тестування програмного продукту за допомогою платформи Postman та аналіз результатів показав, що середня швидкість відповіді серверу 350 мс.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково- технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу WEB-орієнтованої інформаційної технології для розробки експертних систем, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів, якими можуть бути провідні викладачі випускової або спорідненої кафедри чи інші відомі фахівці. Не рекомендується залучати експертами керівника магістерської кваліфікаційної роботи та завідувача відповідної випускової кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за дванадцяти критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за п’ятибальною шкалою)					
	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах

Продовження таблиці 4.1

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 нижча за ціни аналогів	Ціна продукту значно нижча 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

Продовження таблиці 4.1

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
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються 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 5ти років. Термін окупності інвестицій більший 5ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 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 потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	2	4	3
2. Ринкові переваги (наявність аналогів)	3	2	4
3. Ринкові переваги (ціна продукту)	4	3	2
4. Ринкові переваги (технічні властивості)	4	3	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	4
6. Ринкові перспективи (розмір ринку)	3	3	2
7. Ринкові перспективи (конкуренція)	3	4	2
8. Практична здійсненність (наявність фахівців)	4	3	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	2	3
11. Практична здійсненність (термін реалізації)	4	2	3
12. Практична здійсненність (розробка документів)	3	2	4
Сума балів	38	33	36
Середньоарифметична сума балів $СБ_c$	$\text{СБ}_\text{С} = 1\frac{\sum_{i=1}^{3}\text{СБ}_i}{3} = 35{,}7$		

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

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Отже, згідно таблиці 4.3, розроблена WEB-орієнтована інформаційної технологія для розробки експертних систем має науково-технічний рівень вищий середнього. Було досягнуто за рахунок розширення функціональних можливостей розробки порівняно з аналогічними розробками.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної, дослідно-конструкторської, конструкторсько-технологічної роботи, створенням дослідного зразка і здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- спецустаткування для наукових (експериментальних) робіт;
- програмне забезпечення для наукових (експериментальних) робіт;
- витрати на роботи, які виконують сторонні підприємства, установи і організації;
- інші витрати;
- накладні (загальновиробничі) витрати.

4.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам,

креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \times t_i}{T_p}, \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень; $М_{пі}$ – місячний посадовий оклад конкретного дослідника, грн; t_i – кількість днів роботи конкретного дослідника, дн.; $Т_р$ – середня кількість робочих днів в місяці, $Т_р$=21…23 дні. Проведені розрахунки бажано звести до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	35 000	1750	38	66 500
Науковий співробітник	25 000	1250	38	47 500
Всього				114 000

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата дослідників та робітників

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \times \frac{Н_{дод}}{100\%}, \quad (4.2)$$

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

$$З_д = (114\,000 * 12\ \% / 100\ \%) = 13\,680 \text{ (грн.)}.$$

4.2.2 Відрахування на соціальні заходи

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

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

$$Н_з = (З_о + З_р + З_{дод}) \times \frac{Н_{зп}}{100\%}, \quad (4.3)$$

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

$$Н_з = (114\,000 + 13\,680) \cdot 22\ \% / 100\ \% = 28\,089{,}60 \text{ (грн.)}.$$

4.2.3 Амортизація обладнання, програмних засобів та приміщень

До статті «Амортизація обладнання, програмних засобів та приміщень» відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві.

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

$$А_{обл} = \frac{Ц_б}{Т_в} \times \frac{Т_{вик}}{12}, \quad (4.4)$$

$$А_{обл} = \frac{90\,000}{2} \times \frac{1.9}{12} = 15\,000,$$

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; $Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці.

Таблиця 4.5 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп`ютер (MacBook Pro 14-inch, 2021)	90 000	2	1.9	15 000
Приміщення	1 342 000	20	1.9	22 366,6
Офісне обладнання	45 000	4	1.9	3 750
Всього				41 116,6

4.2.4 Паливо та енергія для науково-виробничих цілей

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень. Стаття формується у разі виконання енергоємних наукових досліджень за методом прямого внесення витрат і досягає значної питомої ваги у собівартості досліджень.

Витрати на силову електроенергію ($В_е$) розраховують за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \times t_i \times Ц_е \times К_{впі}}{\eta_{i=1}}, \quad (4.5)$$

$$B_e = \sum \frac{0{,}096 \times 320 \times 28{,}969 \times 0{,}9}{\eta_{i=1}} = 800{,}93,$$

$$B_e = \sum \frac{0{,}036 \times 320 \times 28{,}969 \times 0{,}9}{\eta_{i=1}} = 300{,}35,$$

$$B_e = \sum \frac{0{,}024 \times 80 \times 28{,}969 \times 0{,}9}{\eta_{i=1}} = 50{,}05,$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год; $Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії); $К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$; η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки необхідно звести до таблиці.

$$Ц_е = \left(Ц_{опт} + Ц_{розп} + Ц_{пост}\right)\left(1 + \frac{ПДВ}{100\%}\right), \quad (4.6)$$

$$Ц_е = (3{,}99466 + 0{,}19947 + 19{,}947)\left(1 + \frac{20\%}{100\%}\right) = 28{,}969$$

де $Ц_{опт}$ – середня оптова ціна електроенергії, яка визначається оператором ринку (без ПДВ), грн за 1 кВт·год; $Ц_{розп}$ – вартість розподілу електроенергії окремою енергорозподільчою компанією (без ПДВ), грн за 1 кВт·год; $Ц_{пост}$ – вартість

постачання електроенергії від енергорозподільчої компанії до конкретного споживача (без ПДВ), грн за 1 кВт·год. ПДВ – величина податку на додану вартість, %, у 2022 році ПДВ=20%.

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп`ютер (MacBook Pro 14-inch, 2021)	0.096	320	800,93
Освітлення	0.024	80	50,05
Всього			850,98

4.2.5 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» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \times \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати».

$$I_в = 114\,000 * 55\% / 100\% = 62\,700 \text{ (грн.)}$$

4.2.6 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 науково-технічну інформацію та рекламу та ін.

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

$$В_{нзв} = (З_о + З_р) \times \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

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

$$H_{нзв} = 114\,000 * 111\% / 100\% = 126\,540 \text{ (грн.)}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.9)$$

$$B_{заг} = 114\,000 + 13\,680 + 28\,089{,}60 + 41\,116{,}6 + 850{,}98 + 62\,700 + 126\,540 = 386\,977{,}18 \text{ грн.}$$

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

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.10)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії: науково-

дослідних робіт, то η=0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9.

ЗВ = 386 977,18 / 0,5 = 773 954,36 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

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. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 4-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

– абсолютного економічного ефекту (чистого дисконтованого доходу);
– внутрішньої економічної дохідності (внутрішньої норми дохідності);
– терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення інформаційної технології для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \qquad (4.11)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу; N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки; $Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$; Цб – вартість програмного продукту у році до впровадження результатів розробки; ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від

покращення його певних характеристик; λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333. Р – коефіцієнт, який враховує рентабельність продукту; ϑ – ставка податку на прибуток, у 2021 році ϑ =18%.

Припустимо, що при прогнозованій ціні 1000 грн. за програмний продукт, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 500 грн. Кількість проданих підписок на програмний продукт також збільшиться: протягом першого року – на 3000 шт., протягом другого року – на 5000 шт., протягом третього року на 7000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*500 + (1000 + 500)*3000)* 0,8333* 0,36) * (1 – 0,18) = 1 106 955,72 грн.

$\Delta П_2$ = (0*500 + (1000 + 500)*(3000 + 5000)* 0,8333* 0,36) * (1 – 0,18) = 2 951 881,92 грн.

$\Delta П_3$ = (0*500 + (1000 + 500)*(3000 + 5000 + 7000)* 0,8333* 0,36) * (1 – 0,18) = 5 534 778,6 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 9 593 616,24 грн.

4.3.2 Розробка чи суттєве вдосконалення інформаційної технології для використання масовим споживачем.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-дослідної розробки:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$ збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної роботи, грн; Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної роботи, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15; t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (1 106 955,72 / (1 + 0,1)^1) + (2 951 881,92 / (1 + 0,1)^2) + (5 534 778,6 / (1 + 0,1)^3) = 1 006 323,38 + 2 439 571,83 + 4 158 361,08 = 7 604 256,29 грн.

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

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

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$ = 2…5, але може бути і більшим;

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

PV = 2 * 773 954,36 = 1 547 908,72 грн.

Тоді абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - \text{PV}, \quad (4.14)$$

$$E_{\text{абс}} = 7\,604\,256{,}29 - 1\,547\,908{,}72 = 6\,056\,347{,}57 \text{ грн.}$$

Оскільки $E_{\text{абс}} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_{\text{в}}$. Для цього використаємо формулу:

$$E_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1, \quad (4.15)$$

$\text{Т}_{\text{ж}}$ життєвий цикл наукової розробки, роки.

$$E_{\text{в}} = \sqrt[3]{(1 + 6\,056\,347{,}57 \div 1\,547\,908{,}72)} - 1 = 0{,}699$$

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

$$\tau = d + f, \quad (4.16)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,23…0,25); f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}25 + 0{,}05 = 0{,}29.$$

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:

$$T_{ок} = \frac{1}{E_в}, \quad (4.17)$$

$$T_{ок} = 1 / 0{,}699 = 1{,}43 \text{ р.}$$

Оскільки < 3-х років, а саме термін окупності рівний 1,43 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновок до розділу 4

Економічна частина даної роботи містить розрахунок витрат на розробку WEB-орієнтованої інформаційної технології для розробки експертних систем, сума яких складає 773 954,36 гривень. Було прогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є високо конкурентоспроможним.

ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки серверної частини WEB-орієнтованої інформаційної технології для розробки експертних систем.

У першому розділі магістерської роботи було здійснено аналіз предметної області надання рекомендацій та аналіз об’єкту проектування WEB - орієнтованої інформаційної технології для розробки експертних систем. Здійснено аналіз технологій аналогів на прикладі Expert Shell, Мала експертна система, Gimet, що дало можливість визначити основні проблеми та недоліки. Найперспективнішим аналогом було визначено веб-орієнтовану платформу для створення експертних систем «GIMET». Недоліком аналогу є недостатній фунціонал та недостатня швидкодія, а звідси і випливає мета даної роботи – розробка серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем та підвищення швидкодії серверної обробки інформації.

У другому розділі магістерської кваліфікаційної роботи було наведено порівняльний аналіз систем надання рекомендацій, зокрема розглянуто найбільш поширенні системи надання рекомендацій, а саме системи підтримки прийняття рішень, експертні системи, рекомендаційні системи. Проаналізовано різні моделі, методи та технології надання рекомендацій та моделей подання знань, зокрема контент-орієнтована та колаборативна фільтрації та експертні системи. Розроблено структуру та визначено основні вимоги до серверної частини WEB - орієнтованої інформаційної технології для розробки експертних систем.

У третьому розділі розглянуто плюси та мінуси популярних середовищ розробки та обґрунтовано її вибір. У результаті було розроблено серверну частину WEB - орієнтованої інформаційної технології для розробки експертних систем, створену мовою програмування Node.js із застосуванням безкоштовної бібліотеки Express.js. Було розроблено алгоритм роботи програмного продукту та наведено схеми основних алгоритмів роботи. Було проведено автоматизоване тестування API за допомогою платформи Postman. Аналіз результатів тестування показує, що сервер

відповідає поставленим вимогам. Тобто мета магістерської кваліфікаційної роботи досягнута – середня швидкість відповіді 350 мс, що свідчить про покращення швидкодії відповіді сервера порівняно із системою аналогом, в якого швидкість відповіді серверу 420 мс.

У четвертому розділі було виконано розрахунок витрат на розробку та виготовлення нового технічного рішення, сума яких складає 773 954,36 гривень. Було прогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який потенційно може отримати виробник від реалізації розробленого технічного рішення, знайдено термін окупності витрат виробника та економічний ефект для споживача при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розробка у виробництві та використання дешевша за аналог і є високо конкурентоспроможною. Період окупності складе близько 1,43 роки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шептяков, І.; Левченко, Н.; Ярова, О.; Мельник, О. ANALYSIS OF SOFTWARE TOOLS FOR EXPERT SYSTEMS DEVELOPING. – LI Науково-технічна конференція підрозділів ВНТУ, Ukraine, May. 2022. Available at: https://conferences.vntu.edu.ua/index.php/all-fbtegp/all-fbtegp-2022/paper/view/15460/13008

2. Шептяков, І.; Левченко, Н.; Ярова, О.; Яровий, А. ОСОБЛИВОСТІ ПРОЕКТУВАННЯ WEB-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ. – LI Науково-технічна конференція підрозділів ВНТУ, Ukraine, May. 2022. Available at: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2022/paper/view/15498/13024

3. Тест Тюрінга - Вікіпедія [Електронний ресурс]. - Доступний з <https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%A2%D1%8E%D1%80%D1%96%D0%BD%D0%B3%D0%B0>.

4. Експертні системи. Частина 1. Навчальний посібник. – Вінниця:ВНТУ, 2006.– 114 с.

5. Експертні системи. Частина 2 : навчальний посібник / Яровий А. А., Арсенюк І. Р., Месюра В. І. – Вінниця : ВНТУ, 2017. – 106 с.

6. Мала експертна система [Електронний ресурс]. - Доступний з http://bourabai.ru/alg/mes2.htm

7. Яровий А.А., Яровий А.М., Малик Н.О. Свідоцтво про реєстрацію авторського права на твір No 24444. Комп'ютерна програма "Оболонка експертної системи продукційного типу для підтримки процесу прийняття рішень "Decision Support Expert:Shell" / Дата реєстрації ДДІВ України 13.05.2008.

8. Веб-орієнтована програмна платформа для створення експертних систем «GIMET» / Яровий А., Маначинський О., Супрун Р., Попова І. : Збірник праць XI Міжнародної науково-практичної конференції [Інтернет-Освіта-Наука (ІОН-2018)], (Вінниця, 22-25 травня 2018 р.) – Вінниця, ВНТУ, 2018. – с. 61-63.

9. Amazon Simple Storage Service [Електронний ресурс]. Доступний з https://aws.amazon.com/s3/?nc1=h_ls

10. Amazon RDS PostgreSQL [Електронний ресурс]. Доступний з https://aws.amazon.com/rds/postgresql/

11. REST та SOAP [Електронний ресурс]. Доступний з https://habr.com/ru/post/483204/

12. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Системи штучного інтелекту: навчальний посібник [Рекомендовано МОНУ]. 2-е видання за ред. В.В. Пасічника – Львів: "Магнолія 2006", 2013. – 279 с.

13. Рекомендаційна система [Електронний ресурс]. - Доступний з https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

14. П. Джексон Экспертные системы. – М.: ИД «Вильямс», 2001. – 609 с.

15. Контент-орієнтовна фільтрація [Електронний ресурс]. Доступний з https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BD%D1%82-%D1%84%D1%96%D0%BB%D1%8C%D1%82%D1%80

16. Колаборативна фільтрація [Електронний ресурс]. Доступний з https://en.wikipedia.org/wiki/Collaborative_filtering

17. Системи прийняття рішень [Електронний ресурс]. - Доступний з https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BF%D1%96%D0%B4%D1%82%D1%80%D0%B8%D0%BC%D0%BA%D0%B8_%D1%80%D1%96%D1%88%D0%B5%D0%BD%D1%8C

18. Підходи до створення рекомендаційних систем [Електронний ресурс]. Доступний з https://mathchi.medium.com/recommendation-systems-8999834e444

19. Пролог [Електронний ресурс]. Доступний з https://www.wiki.uk-ua.nina.az/%D0%9F%D1%80%D0%BE%D0%BB%D0%BE%D0%B3.html

20. Інтерпретатор продукцій [Електронний ресурс]. Доступний з https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0

_%D0%B2%D0%B8%D1%81%D0%BD%D0%BE%D0%B2%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

21. HTTP [Електронний ресурс]. Доступний з https://uk.wikipedia.org/wiki/HTTP

22. Хмарні інстанси [Електронний ресурс]. Доступний з https://aws.amazon.com/ru/what-is/cloud-instances/

23. Amazon EC2 [Електронний ресурс]. Доступний з https://aws.amazon.com/ec2/

24. JetBrains WebStorm [Електронний ресурс]. Доступний з https://www.jetbrains.com/webstorm/

25. Eclipse [Електронний ресурс]. Доступний з https://en.wikipedia.org/wiki/Eclipse_(software)

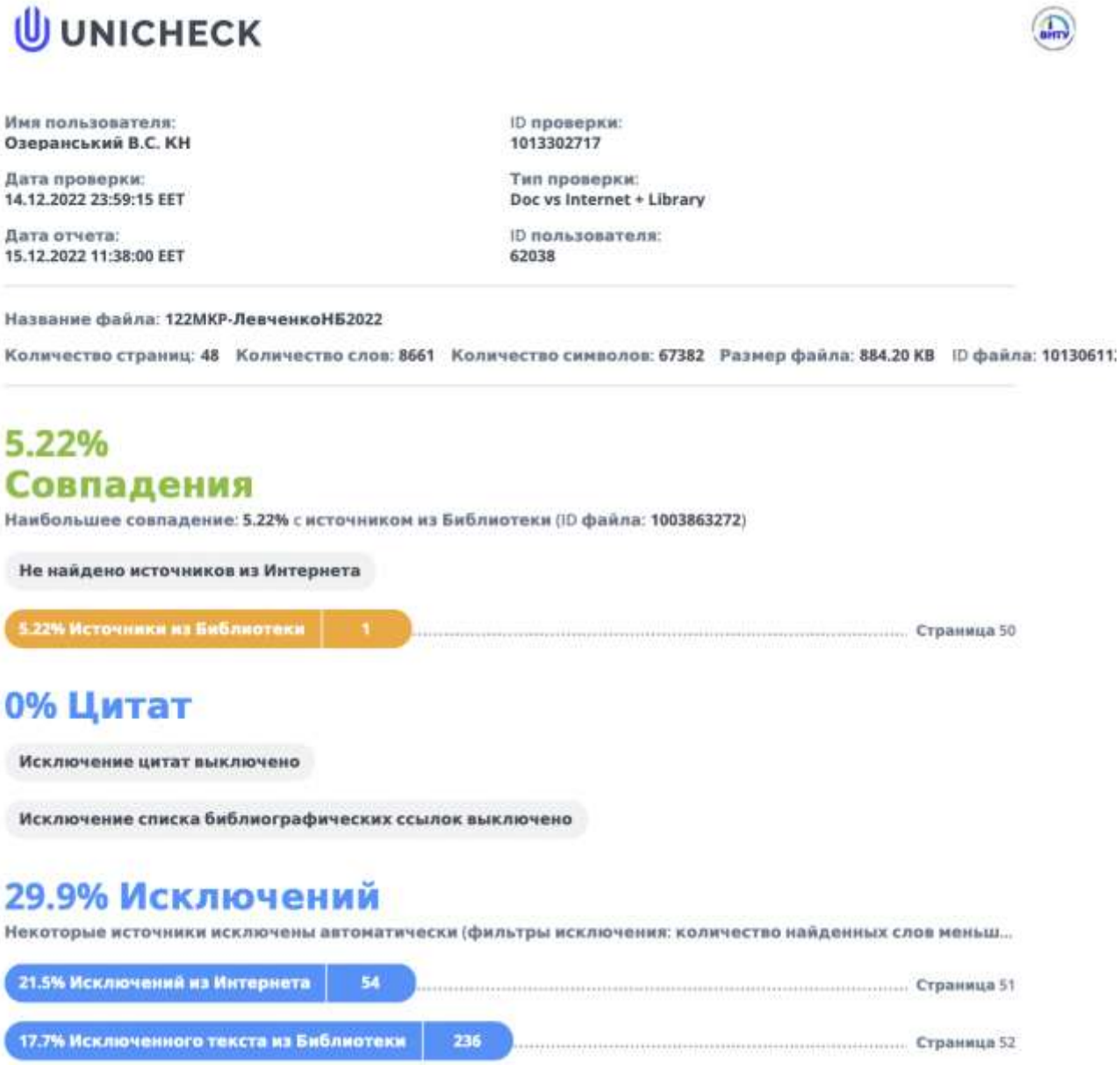
26. Visual Studio [Електронний ресурс]. Доступний з https://visualstudio.microsoft.com/ru/

27. Node.js [Електронний ресурс]. Доступний з https://habr.com/ru/post/420123/

28. Postman [Електронний ресурс]. Доступний з https://learning.postman.com/docs/writing-scripts/test-scripts/

ДОДАТКИ

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)
ПРОТОКОЛ ПЕРЕВІРКИ МКР НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ



UNICHECK

Имя пользователя:
Озеранський В.С. КН

ID проверки:
1013302717

Дата проверки:
14.12.2022 23:59:15 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
15.12.2022 11:38:00 EET

ID пользователя:
62038

Название файла: 122МКР-ЛевченкоНБ2022

Количество страниц: 48 Количество слов: 8661 Количество символов: 67382 Размер файла: 884.20 KB ID файла: 10130611

5.22%
Совпадения

Наибольшее совпадение: 5.22% с источником из Библиотеки (ID файла: 1003863272)

Не найдено источников из Интернета

5.22% Источники из Библиотеки 1 ..... Страница 50

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

29.9% Исключений

Некоторые источники исключены автоматически (фильтры исключения: количество найденных слов меньш...

21.5% Исключений из Интернета 54 ..... Страница 51

17.7% Исключенного текста из Библиотеки 236 ..... Страница 52

ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ)
ЛІСТИНГ ПРОГРАМИ

```
const http = require( 'http' );
const express = require( 'express' );
const session = require( 'express-session' );
const cors = require( 'cors' );
const fs = require( 'fs' );
const helmet = require( 'helmet' );
const asyncHandler = require( 'express-async-handler' );
const { errorHandler } = require( './middlewares/errorHandler' );
const mongoose = require('./mongoose');
const cookieParser = require('cookie-parser');
const config = require('./config/config');
const { NODE_ENV, SESSION_SECRET } = require('./dotenv');

require( 'colors' );

const app = express();

const MongoStore = require('connect-mongo');

app.use( cors() );
app.use( helmet() );
app.use( cookieParser() );

app.use( session( {
    secret: SESSION_SECRET,
    resave: false,
    saveUninitialized: false,
```

```
        key: 'es',
        store: MongoStore.create( {
            mongoUrl: config[ NODE_ENV ].mongoDB,
            mongoOptions: {
                useUnifiedTopology: true
            }
        } )
    } ) )

    require( './sequelize' );

    app.use( ( req, res, next ) => {
        console.log( `${req.method},AppVersion => ${req.get( 'AppVersion' )} |
${req.url}`.red );
        next();
    } );

    app.use( express.urlencoded( { extended: true } ) );
    app.use( express.json( { limit: '100mb' } ) );
    // Create HTTP server.
    const server = http.createServer( app );

    const authRoutes = require('./routes/auth')
    const docsRoutes = require('./routes/docs')

    app.use( '/api/auth', authRoutes );
    app.use( '/api/docs', docsRoutes );
    app.use( errorHandler );

    module.exports = { server, app };
```

```
const router = require( 'express' ).Router();
const asyncHandler = require( 'express-async-handler' );
const AuthService = require( '../../services/AuthService' );
const checkValidationResult = require( '../../middlewares/checkValidationResult' );
const { check } = require( 'express-validator' );

router.post( '/login', [
    check( 'email', 'email should be a string.' )
        .exists()
        .isString(),
    check( 'password', 'password should be a string.' )
        .exists()
        .isString(),
], checkValidationResult, asyncHandler( async ( req, res ) => {
    const { email, password } = req.body;

    const result = await AuthService.login( {
        email, password
    } );
    const { user, tokens } = result;

    return res.status( 200 ).send( {
        data: {
            user,
            tokens
        }
    } );
} ) );
```

```
router.post( '/signup', [
    check( 'name', 'name should be a string.' )
        .optional()
        .isString(),
    check( 'email', 'email should be a string.' )
        .exists()
        .isString(),
    check( 'password', 'password should be a string.' )
        .exists()
        .isString(),
], checkValidationResult, asyncHandler( async ( req, res ) => {
    const { name, email, password } = req.body;

    if ( !( email && password ) ) {
        return res.status(400).send({ error: "Data not formatted properly" });
    }

    const user = await AuthService.signup( {
        name, email, password
    } );

    return res.status(201).send( {
        data: {
            user
        }
    } )
} ) );

router.post( '/refresh', asyncHandler( async ( req, res ) => {
    const { refreshToken } = req.body;
```

```
        const newTokens = await AuthService.refreshTokens( refreshToken );

        return res.status( 200 ).send( {
            data: newTokens
        } );
    } ) );

module.exports = router;
    const router = require( 'express' ).Router();
    const asyncHandler = require( 'express-async-handler' );
    const DocumentService = require( '../../services/DocumentService' );
    const checkValidationResult = require( '../../middlewares/checkValidationResult' );
    const checkAuth = require( '../../middlewares/checkAuth' );
    const { check } = require( 'express-validator' );
    const multer = require( 'multer' );
    const storage = multer.memoryStorage();
    const upload = multer( { storage: storage, limits: { fileSize: 100000000 } } );

    router.patch( '/save', [
        check( 'email', 'email should be a string.' )
            .exists()
            .isString(),
         check( 'password', 'password should be a string.' )
            .exists()
            .isString(),
    ], checkValidationResult, checkAuth, upload.single( 'file' ), asyncHandler( async (
req, res ) => {
        const { file, tokenPayload } = req;
        const { fileId, title } = req.body;
```

```
    const { userId } = tokenPayload;

    if( !file ) {
        throw new Error('FILE_SHOULD_BE_EXIST')
    }

    const result = await DocumentService.saveFile( {
        file: file.buffer, userId, title, fileId
    } )

    return res.status( 200 ).send( {
        success: true,
        errors: null,
        data: {
            result
        }
    } );
} ) );

router.get( '/', checkAuth, asyncHandler( async ( req, res ) => {
    const { tokenPayload } = req;
    const { userId } = tokenPayload;

    const result = await DocumentService.getListByUser( {
        userId
    } )

    return res.status( 200 ).send( {
        success: true,
        errors: null,
```

```
            data: result
        } );
    } ) );

    router.delete( '/:fileId', checkAuth, asyncHandler( async ( req, res ) => {
        const { tokenPayload } = req;
        const { fileId } = req.params;
        const { userId } = tokenPayload;

        const result = await DocumentService.deleteFile( {
            userId, fileId
        } )

        return res.status( 200 ).send( {
            success: true,
            errors: null,
            data: result
        } );
    } ) );

module.exports = router;
```

Додаток В (ОБОВ'ЯЗКОВИЙ)

Додаток В (ОБОВ'ЯЗКОВИЙ)
84

ІЛЮСТРАТИВНА ЧАСТИНА

WEB - ОРІЄНТОВАНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ. ЧАСТИНА 1

Виконав: студент 2-го курсу,
групи 1КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Левченко Н.Б.
(прізвище та ініціали)

Керівник: д.т.н., професор каф. КН
Яровий А.А.
(прізвище та ініціали)
«15» 12 2022 р.

ВНТУ - 2022 рік

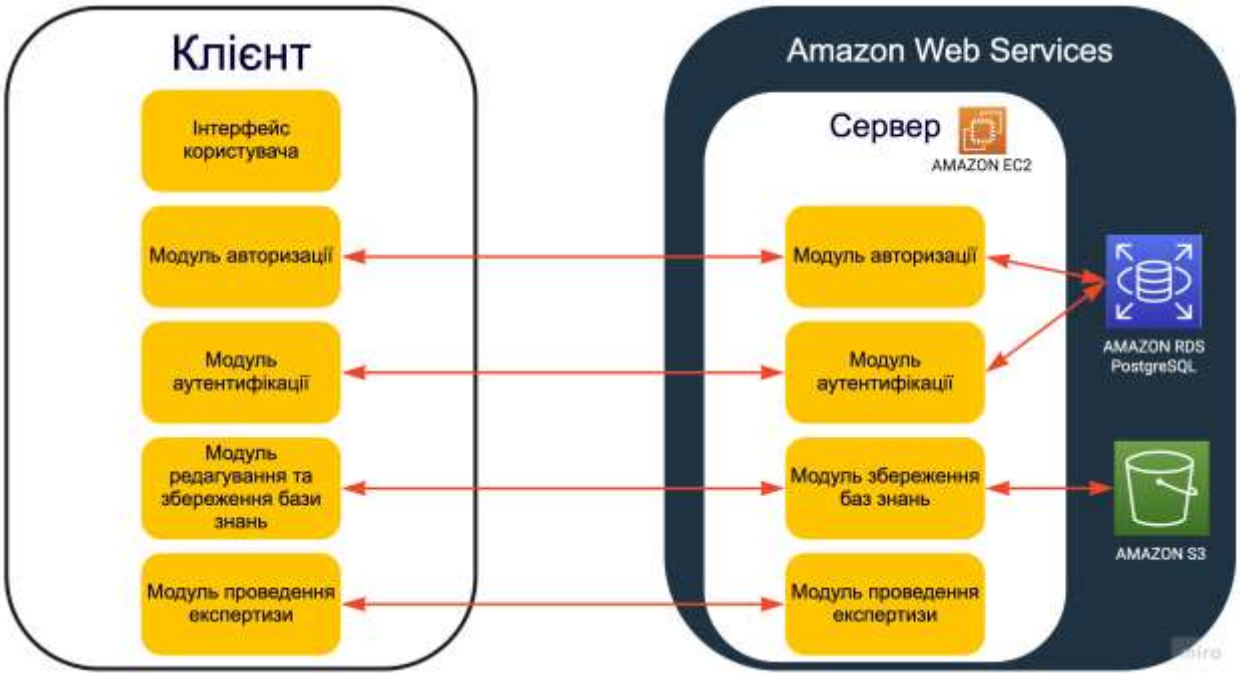


Рисунок В.1 – Узагальнена структурна схема WEB - орієнтованої інформаційної технології для розробки експертних систем.

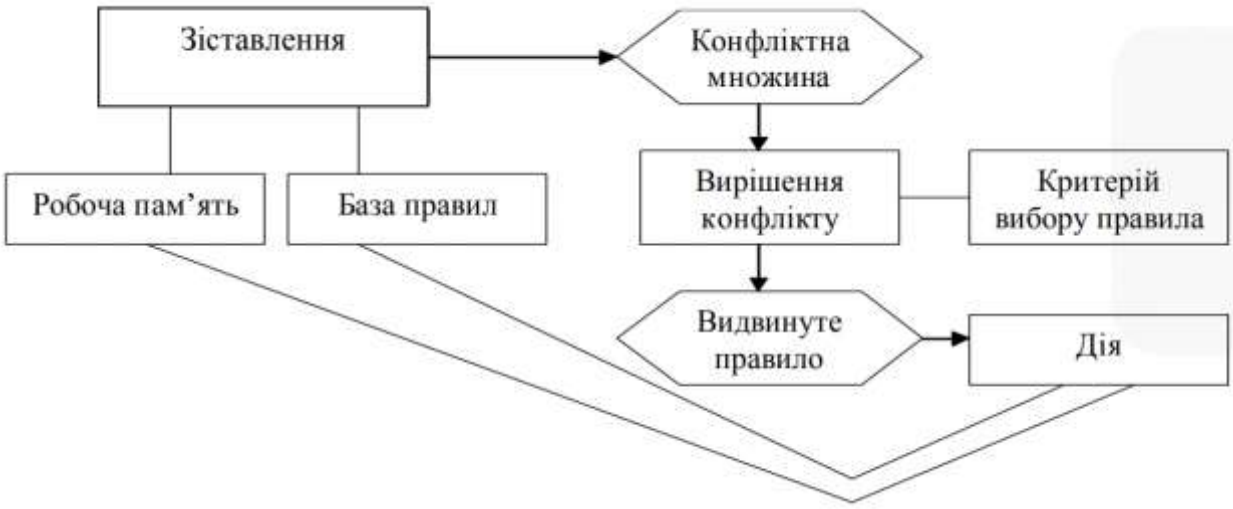


Рисунок В.2 – Схема циклу роботи інтерпретатора.

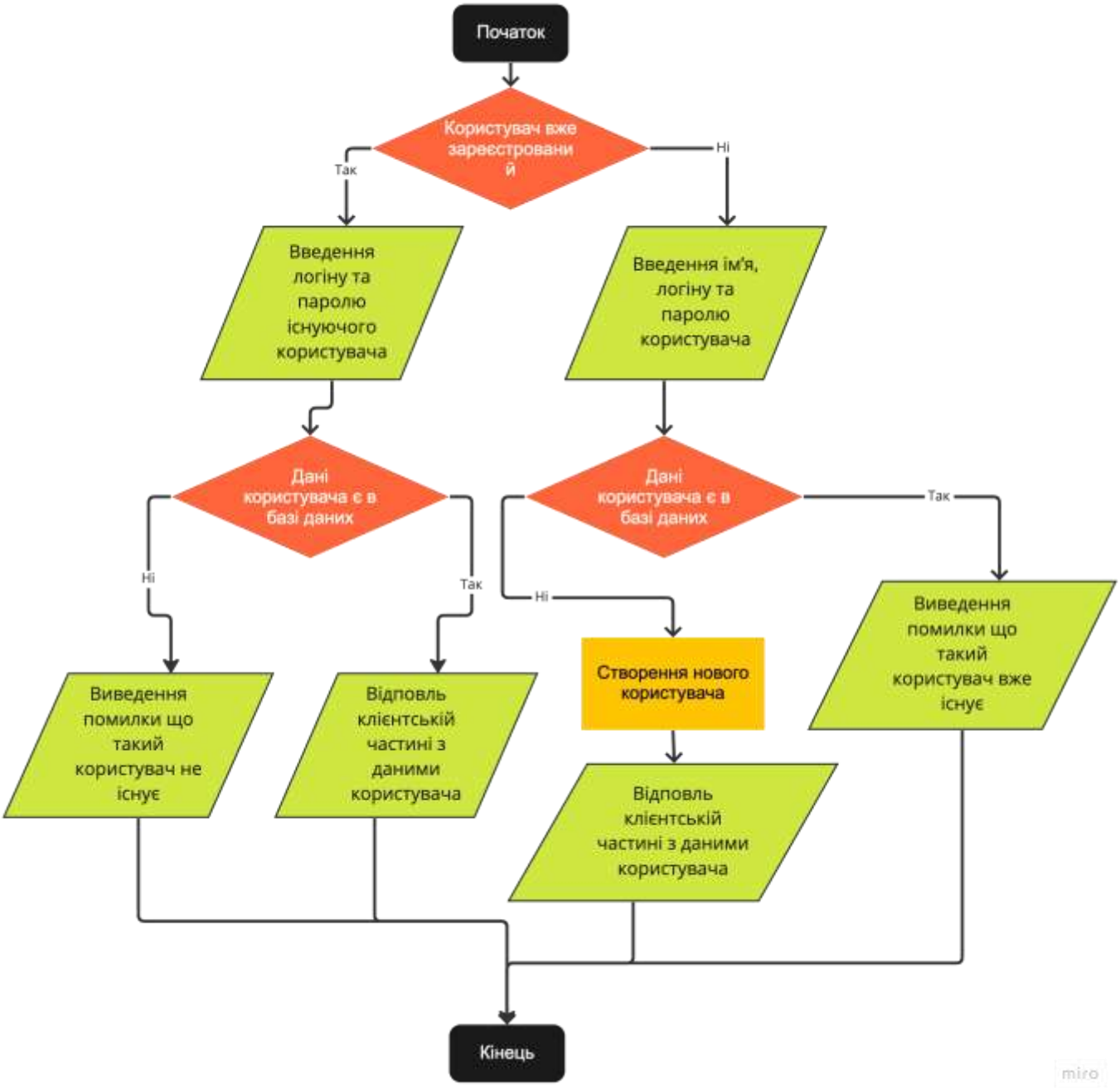


Рисунок В.3 – Схема алгоритму процесу реєстрації / авторизації користувача.

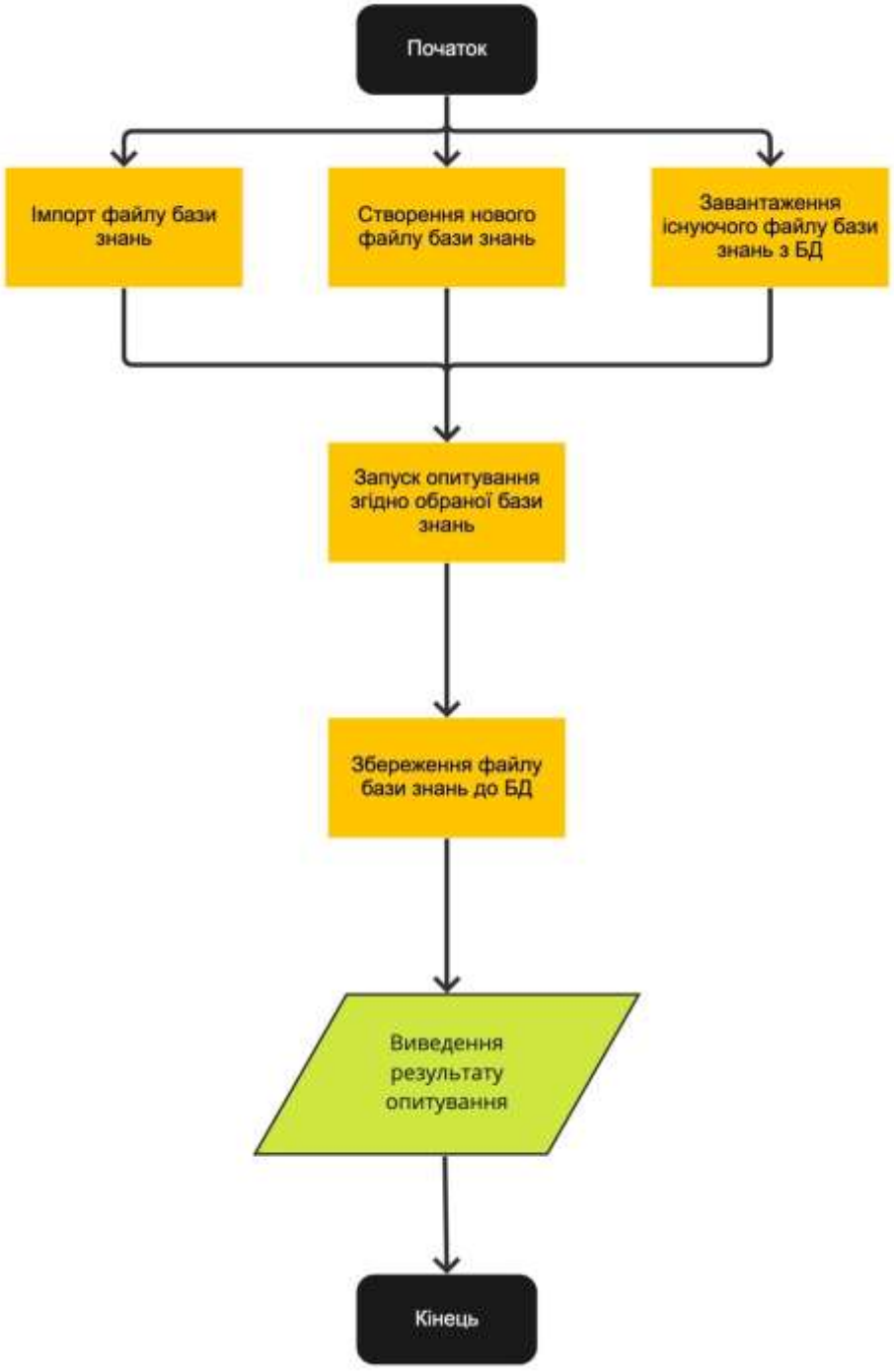


Рисунок В.4 – Схема алгоритму процесу створення / завантаження бази знань.

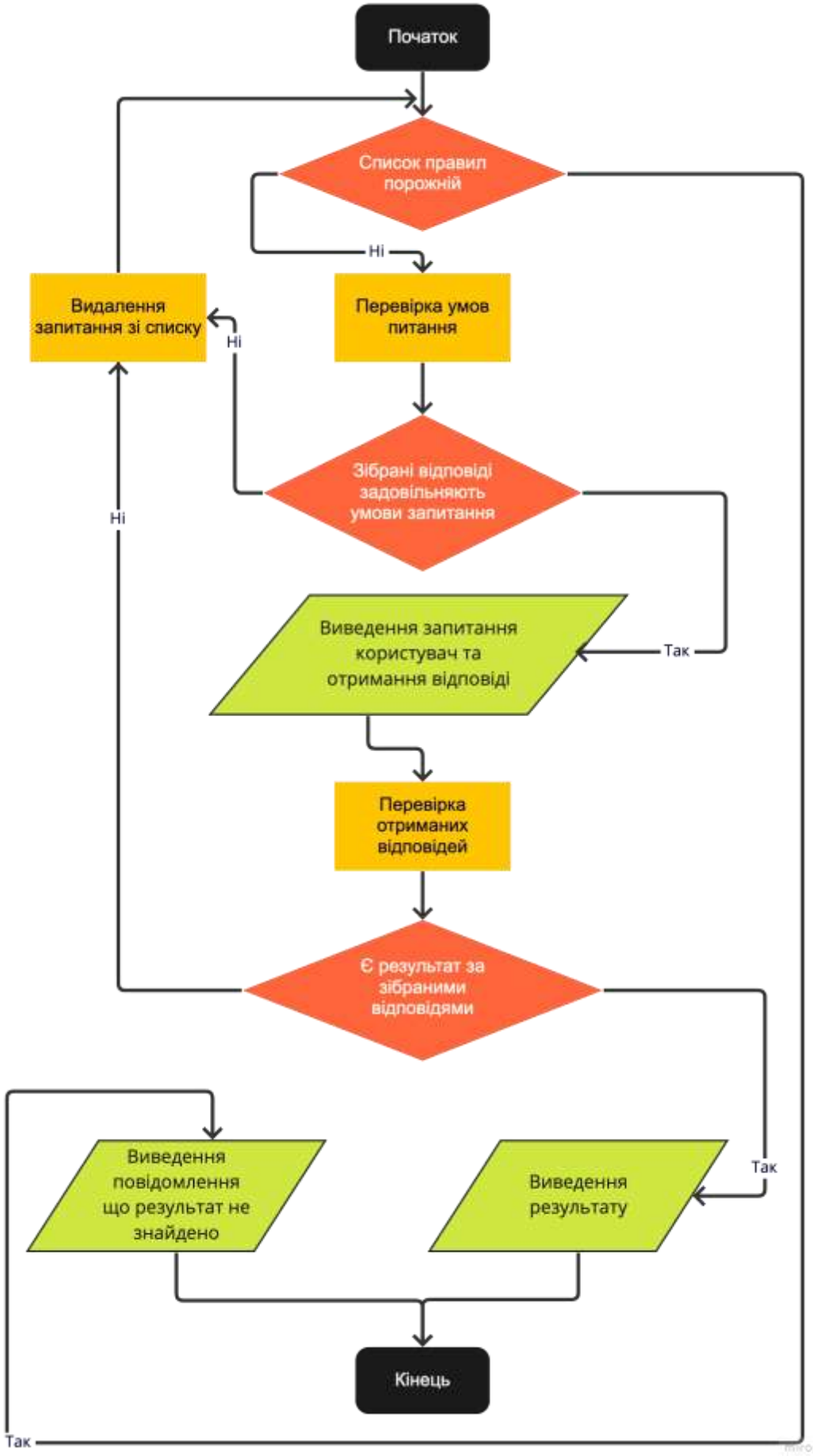


Рисунок В.5 – Схема алгоритму процесу опитування користувача.

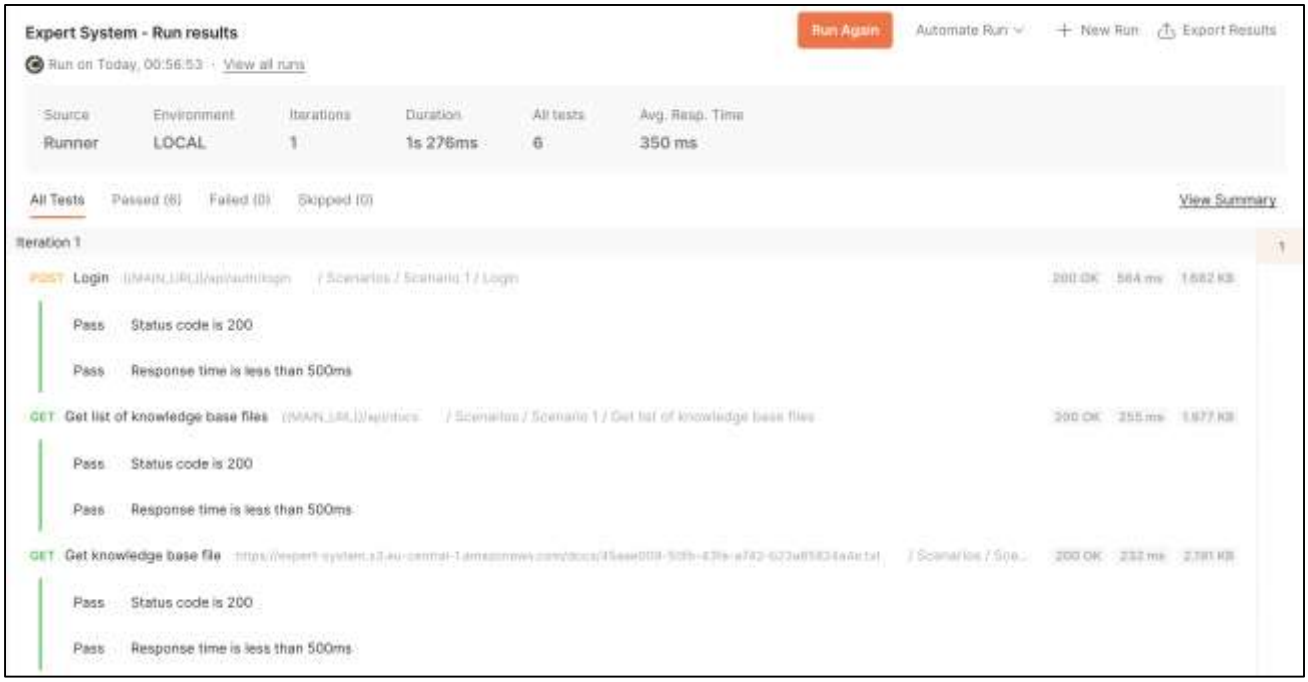


Рисунок В.6 – Приклад звіту автоматизованого тестування.

Додаток Г (довідниковий)

Акт впровадження

Товариство з обмеженою відповідальністю
Код ЄДРПОУ 41051004
21050, м. Вінниця, вул. Соборна, 24

Довідка дана студенту групи 1КН-21м Левченку Нікіті Борисовичу в тому, що програмний продукт «WEB - орієнтована інформаційна технологія для розробки експертних систем» пройшов експериментальне дослідження на ТОВ «АСТА.МОБІ».

За результатами дослідження програмний продукт планується до впровадження.

Директор
ТОВ «АСТА.МОБІ»

А.О. Стахов