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 2»

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

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

Опонент: д.т.н., професор
Бісікало О.В.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 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

Шептяков Ігор Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи WEB-орієнтована інформаційна технологія для розробки експертних систем. Частина 2

керівник роботи д.т.н., професор кафедри КН Яровий А.А.
затверджені наказом вищого навчального закладу від "14" вересня 2022 року №203
2. Строк подання студентом роботи 18 листопада 202_ року
3. Вихідні дані до роботи: наявність опції авторизації та автентифікації користувачів, підтримка браузерів(Chrome, FireFox та Opera), мова програмування – об`єктно-орієнтована, максимальний час завантаження сторінки до 2000 мілісекунд, максимальний обсяг бази знань до 50 мб, формат файлів бази знань .TXT з кодуванням UTF-8.
4. Зміст текстової частини: вступ, аналіз сучасного стану розвитку засобів розробки експертних систем, моделі та структурна організація web-середовища інформаційної технології для розробки експертних систем, програмна реалізація web-середовища інформаційної технології для розробки експертних систем, економічна частина, висновки, перелік використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): загальна схема роботи WEB-середовища для розробки експертних систем, схема алгоритму процесу реєстрації / авторизації користувача, схема алгоритму процесу створення / завантаження бази знань, схема алгоритму процесу опитування користувача, головна сторінка, результати тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Яровий А.А., д.т.н., проф. каф. КН	14.09.2022	14.11.2022
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

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану розвитку засобів розробки експертних систем. Постановка задач дослідження	14.09.2022 - 01.10.2022	
2	Моделі та структурна організація WEB-середовища інформаційної технології для розробки експертних систем	02.10.2022 - 16.10.2022	
3	Програмна реалізація WEB-середовища інформаційної технології 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 _____ (підпис) Шептяков І.О.

Керівник роботи _____ (підпис) Яровий А.А.

АНОТАЦІЯ

УДК 004.89

Шептяков І. О. WEB-орієнтована інформаційна технологія для розробки експертних систем. Частина 2. Магістерська кваліфікаційна робота зі спеціальності 122 – Комп’ютерні науки, освітня програма – Системи штучного інтелекту. Вінниця: ВНТУ, 2022. 105 с.

На укр. мові. Бібліогр.: 21 назв; рис.: 18; табл. 7.

В роботі пропонується WEB-орієнтована інформаційна технологія для розробки експертних систем. Проаналізовано предметну область у сфері розробки експертних систем та визначено принцип її роботи. Проведено поверхневий порівняльний аналіз програмних засобів для створення WEB-інтерфейсів та показано переваги та недоліки кожного з них. Визначена постановка задачі, необхідні критерії та вимоги.

Наведено класифікацію експертних систем. Визначено критерії розробки. Проаналізовано моделі подання знань та здійснено порівняльний аналіз.

Здійснено програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем та проаналізовано її роботу на предмет помилок.

Ключові слова: експертні системи, інформаційна технологія, продукційна система, WEB-середовище, програмний продукт

ABSTRACT

Sheptiakov I. O. WEB-oriented information technology for the development of expert systems. Part 2. Master's qualification work in specialty 122 - Computer Science, educational program - Artificial Intelligence Systems. Vinnytsia: VNTU, 2022. 105 c.

In Ukrainian language. Bibliographer: 21 titles; fig.: 18; table 7.

The paper proposes a WEB-oriented information technology for the development of expert systems. The subject area in the field of expert systems development is analysed and the principle of its work is determined. A superficial comparative analysis of software tools for creating WEB-interfaces is carried out and the advantages and disadvantages of each of them are shown. The problem statement, necessary criteria and requirements are defined.

The classification of expert systems is given. The development criteria are defined. The models of knowledge representation are analysed and a comparative analysis is carried out.

The software implementation of WEB-oriented information technology for the development of expert systems is carried out and its work is analysed for errors.

Keywords: expert systems, information technology, product system, WEB-environment, software product.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Найважливішою прикладною сферою застосування ШІ є сфера експертних систем. Експертна система - це система, заснована на знаннях, яка використовує знання про свою прикладну область і застосовує процедуру виведення для вирішення проблем, які в іншому випадку вимагали б людської компетенції або досвіду. Потужність експертних систем зумовлена, насамперед, специфічними знаннями про вузьку предметну область, що зберігаються в базі знань експертної системи.

Експертні системи є помічниками осіб, які приймають рішення, а не замінюють їх. Експертні системи не володіють можливостями людини. Вони використовують базу знань з певної предметної області і застосовують ці знання до фактів конкретної ситуації, що розглядається.

Для реалізації платформи для розробки та супроводу експертних систем за основу були обрані сучасні WEB-технології. Однією з переваг WEB-орієнтованої платформи є той факт, що користувачі не залежать від конкретної операційної системи, тому WEB-застосунки є міжплатформовими сервісами, що є великою перевагою над подібними оболонками для розробки експертних систем. Актуальність теми дослідження полягає в поєднанні сучасних WEB-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 клієнтської частини.

Для досягнення поставленої мети необхідно розв’язати такі наступні завдання:

– провести аналіз сучасного стану розвитку засобів розробки експертних систем;

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

– розробити алгоритм роботи програмного продукту;

– розробити структурно організаційну схему WEB-середовища для розробки експертних систем;

– програмна реалізація WEB-середовища для розробки експертних систем;

– провести розрахунок витрат на розробку WEB-орієнтованої інформаційної технології для розробки експертних систем.

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 експертних систем, методи інженерії знань, методи та підходи до розробки систем штучного інтелекту, методи та підходи до розробки WEB-орієнтованих програмних додатків, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному:

Розроблено WEB-орієнтовану інформаційну технологію для розробки експертних систем, що відрізняється від існуючих застосуванням WEB-орієнтованого підходу з покращеними функціональними можливостями, що

забезпечило покращення автоматизації процесу розробки експертних систем та підвищення швидкодії їх роботи.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритми процесу реєстрації / авторизації користувача, процесу створення / завантаження бази знань та процесу опитування користувача. Що покращило автоматизацію процесу розробки експертних систем та підвищило швидкодію клієнтської частини.

2. Здійснено програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем із 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-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ(2022) [1, 2].

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

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ ЗАСОБІВ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ

1.1 Аналіз предметної області розробки експертних систем

З середини 1960-х років почали розроблятися експертні системи для дослідницьких цілей. Вони були комерційно вигідними рішеннями реальних проблем з початку 1980-х років[3]. Деякі з основних видів діяльності та питань у різноманітних проектах розробки експертних систем[4]:

- Визначте та переконайте експертів у цій галузі до співпраці;
- Здобути знання;
- Презентація знань;
- Прототипування та програмування;
- Розробка всієї експертної системи та визначення її архітектури;
- Підтвердження та верифікація.

Принципи проектування ЄС дуже відрізняються від принципів традиційних проектів. Така ситуація зумовлена неформальним характером завдання та відсутністю остаточної теорії та методології проектування ЕК. Результатом цього є те, що принципи та методи побудови ЄС необхідно модифікувати безпосередньо в процесі проектування, оскільки знання розробників про предметну область (програмне забезпечення) зростають. У зв’язку з цим найбільш популярним на даному етапі є підхід ЕК «швидкого прототипування», згідно з яким процес розробки ЕК передбачає послідовну розробку кількох прототипів[4].

Перший прототип – це обмежена версія ЄС, покликана продемонструвати доцільність обраного підходу та забезпечити правильне кодування фактів,

зв’язків і стратегій для підтвердження експертних міркувань. Це дозволяє інженерам із знань активно залучати експертів до розвитку ЄС. Область застосування прототипу зазвичай складається з десятків правил, рамок або прикладів. Час розробки коливається від 4 до 12 тижнів, залежно від вибраного інструменту та досвіду розробника. Якщо перший прототип успішний, експерт починає розширювати знання про програмне забезпечення прототипу через інженера з знань. Це накопичення може привести до створення прототипу, який почне успішно вирішувати всі необхідні проблеми з програмним забезпеченням. Однак для перетворення його в кінцевий продукт як правило виконують перепрограмування ЕС на мові більш низького рівня, що дозволяє підвищити продуктивність і зменшити обсяги необхідної пам'яті[4].

База знань – сукупність знань, що відносяться до деякої предметної області, формально представлених так, щоб на їх основі можна було здійснювати міркування. Бази знань найчастіше використовуються в контексті експертних систем, де з їх допомогою подаються навики і досвід експертів, зайнятих практичною діяльністю у відповідної області (наприклад, в медицині або в математиці). Зазвичай база знань є сукупністю правил виводу [5].

Незважаючи на всі свої недоліки, найбільшого поширення набула продукційна модель подання знань. При використанні продукційної моделі база знань складається з набору правил. Програма, яка керує пошуком правил, називається машиною виведення. Машина виведення (інтерпретатор правил) виконує дві функції: по-перше, вона переглядає існуючі факти в робочій пам'яті (базі даних) і правила в базі знань і додає (якщо можливо) нові факти в робочу пам'ять, а по-друге , визначає правила порядку розгляду та застосування. Цей механізм керує процесом консультації, зберігаючи інформацію про результати для користувача, і пропонує йому надати інформацію, щоб запустити наступне правило, коли в робочій пам’яті виявлено недостатньо даних [5].

У переважній більшості систем, заснованих на знаннях, механізм виведення є невеликою за об'ємом програмою і включає два компоненти — один

реалізує власне виведення, інший управляє цим процесом. Дія компонента виведення заснована на вживанні правила, яке називається modus ponens [5].

Правило modus ponens. Якщо відомо, що достеменне твердження А, то існує правило вигляду – «Якщо А, то В», тоді твердження В теж істинно. Правила спрацьовують, коли знаходяться факти, що задовольняють їх лівій частині: якщо достеменна посилання, то має бути достеменний і висновок. Компонент виведення повинен функціонувати навіть при недоліку інформації. Отримане рішення може і не бути точним, проте, система не повинна зупинятися через те, що відсутня частина вхідної інформації [5].

Компонент, що управляє, визначає порядок вживання правив і виконує чотири функції[5]:

Зіставлення — зразок правила зіставляється з наявними фактами.

Вибір — якщо в конкретній ситуації може бути застосовано відразу декілька правив, то з них вибирається одне, найбільш відповідне по заданому критерію (вирішення конфлікту).

Спрацьовування — якщо зразок правила при зіставленні збігся з фактами робочої пам'яті, то правило спрацьовує.

Дія — робоча пам'ять піддається зміні шляхом додавання в неї висновку що спрацював, правила. Якщо в правій частині правила міститься вказівка на дію, то вона виконується (як, наприклад, в системах забезпечення безпеки інформації).

Інтерпретатор продукцій працює в циклі. У кожному циклі він перевіряє всі правила, знаходить з робочої пам’яті те, чия цитата відповідає відомим на даний момент фактам, вибирає правильне правило, заносить його висновок у робочу пам’ять і спочатку повторює цикл. У циклі може працювати лише одне правило. Якщо кілька правил успішно порівнюються з фактом, інтерпретатор вибирає єдиний критерій для одного правила, яке діє в цьому циклі [5].

WEB-додаток — це розподілений додаток, де клієнт браузер - це сервер, а сервер - WEB-сервер. Браузер може бути реалізацією так званого тонкого клієнта

— логіка програми централізована на сервері, і основною функцією браузера є відображення завантаженої інформації з сервера і передачі даних користувача. Одна з переваг такого підходу полягає в тому, що клієнт не прив'язаний до конкретної операційної системи користувача, тому WEB-додаток є кросплатформним сервісом. Через це універсальність і відносна простота розробки WEB-додатків досі популярна з кінця 1990-х і початку 2000-х років.

Значні переваги створення WEB-додатків - це функції, які повинні виконуватися незалежно від клієнтської операційної системи. Замість того, щоб писати іншу версію для Microsoft Windows, Mac OS X, Linux та інші операційні системи, програма створюється один раз [6, 7, 8].

1.2 Аналіз програмних засобів створення WEB-інтерфейсів

Для дослідження обрано чотири мови програмування: Python, Ruby, Javascript, PHP. Ці мови є самими популярними мовами для розробки в WEB-середовищі за даними сайту Github.com.

Python – інтерпретуєма об'єктно-орієнтована мова програмування високого рівня з динамічною типізацією, автоматичним управлінням пам'яттю і зручними високорівневими структурами даних, такими як словники (хештаблиці), списки, кортежі. Підтримує класи, модулі (які можуть бути об'єднані в пакети), обробку винятків, а також багатопотокові обчислення. Пітон має простий і виразним синтаксисом. Мова підтримує кілька парадигм програмування: структурний, об'єктно-орієнтоване, функціональне і аспектноорієнтоване [9].

PHP – скриптова мова загального призначення, інтенсивно застосовується для розробки WEB-додатків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних WEB-сайтів [10].

В області WEB-програмування, зокрема серверної частини, PHP - один з популярних сценарних мов Популярність в області побудови WEB-сайтів визначається наявністю великого набору вбудованих засобів для розробки WEB-додатків.

Основні з них:

– автоматичне вилучення POST і GET-параметрів, а також змінних оточення WEB-сервера;

– взаємодія з великою кількістю різних систем управління базами даних (MySQL, MySQLi, SQLite, PostgreSQL);

– автоматизована відправка HTTP-заголовків;

– робота з HTTP-авторизацією;

– робота з cookies і сесіями;

– робота з локальними і віддаленими файлами, сокетами;

– обробка файлів, що завантажуються на сервер;

– робота з XForms.

В даний час PHP використовується сотнями тисяч розробників. Згідно з рейтингом корпорації TIOBE, що базується на даних пошукових систем, в травні 2016 року PHP знаходився на 6 місці серед мов програмування. До найбільших сайтів, які використовують PHP, відносяться Facebook, Wikipedia та ін. Входить в LAMP - поширений набір програмного забезпечення для створення та хостингу WEB-сайтів (Linux, Apache, MySQL, PHP).

Ruby – динамічна, рефлективна, що інтерпретується високорівнева мова програмування. Мова має незалежної від операційної системи реалізацією багатопоточності, суворої динамічною типізацією, складальником сміття і багатьма іншими можливостями. За особливостями синтаксису він близький до

мов Perl і Eiffel, по об'єктно-орієнтованого підходу - до Smalltalk. Також деякі риси мови взяті з Python, Lisp, Dylan і Клу [11].

Javascript - скриптова мова, призначена для створення інтерактивних WEB-сторінок, проте на цьому сфера її застосування не закінчується. Javascript активно використовують при розробці серверної частини WEB-додатку чи навіть ігор [12].

У javascript є свій стандарт: ECMAScript, специфікація якого знаходиться на сайті в розділі «стандарт мови». За допомогою javascript можна виконувати наступні дії:

– змінювати сторінку, писати на ній текст, додавати і видаляти теги, міняти стилі елементів;

– реагувати на події: скрипт може чекати, коли щось станеться (клік, закінчення завантаження сторінки) і реагувати на це виконанням функції;

– виконувати запити до сервера і завантажувати дані без перезавантаження сторінки;

– встановлювати і зчитувати cookie, валідувати дані, виводити повідомлення і багато іншого.

До сильної сторони даної мови належить повна підтримка браузерами. Наприклад, такі технології як ActiveX, VBScript, XUL - підтримуються не в кожному браузері. Такі технології як Flash, Silverlight, Java - не повністю інтегровані з браузером, працюють в своєму оточенні. Також, дана мова програмування не позбавлена недоліків, таких, як, наприклад, нестрога типізація, що є причиною неявного переведення типів. Однак усі недоліки перекриваються поширеністю та кросбраузерністю мови. Для більш продуктивного і простого процесу розробки було вибрано фреймворк React.js.

React (React.js або ReactJS) - інтерфейсна бібліотека JavaScript із відкритим кодом для побудови користувальницьких інтерфейсів або компонентів інтерфейсу[13].

React - найпопулярніша інтерфейсна бібліотека JavaScript у галузі WEB-розробки. Її використовують великі, засновані компанії та нові стартапи (Netflix, Airbnb, Instagram та New York Times, щоб назвати декілька). React приносить в розробку багато переваг, що робить його кращим вибором, ніж інші фреймворки, такі як Angular.js. React зазвичай використовують через його особливість архітектури додатку – усі елементи зовнішнього виду сайту зберігаються в компонентах та окремих файлах, що спрощує розробку та масштабування WEB-додатку[13].

Для написання коду React, використовується спеціальний тип файлів –JSX. Даний тип файлів – це деяка суміш JavaScript та HTML коду, але насправді це все JS код[13]. Відомі проекти на React:

– Facebook. У ньому React використовується частково, але і в версії для ПК, і в мобільному додатку;

– Instagram. В такому популярному додатку даній бібліотеці відводиться величезна роль. Починаючи з можливості визначення геопозіції і закінчуючи точністю функціоналу пошуку - подібні речі часто роблять на React;

– Netflix. Найактивніше задіюється на платформі Gibbon. Основною функцією стає можливість налаштувати параметри для телевізорів з низькою продуктивністю. Бібліотека допомагає прискорити завантаження і підвищити продуктивність - ось де програмісти точно знали для чого потрібен React js;

– WhatsApp - фахівці цього сервісу вирішили використовувати React для створення користувацьких інтерфейсів;

– Dropbox. На хвилі популярності бібліотеки, її почали застосовувати і для цього сайту. В розробці WEB-додатку React.js використовується як

основний фреймворк для створення користувацького інтерфейсу зручним та легким способом, який легко масштабується, саме тому ми обрали його в якості розробки WEB-середовища.

1.3 Аналіз об’єкту проектування

Основною метою розробки платформи для створення експертних систем є надання користувачам можливості легкого створення бази знань підвищенної складності та подальшого використання її для надання користувачам вирішення їхніх задач. Загальні вимоги до WEB-орієнтованого середовища:

- Реєстрація користувачів та створення облікових записів;
- Наявність персонального кабінету;
- Можливість користувачу створювати власті експертні системи;
- Доступність ЕС для проходження консультації;
- Імпорт бази знань;
- Експорт бази знань.

Це завдання є актуальним з огляду на розвиток інтелектуальної комп’ютерної техніки, що забезпечує вирішення конкретних завдань (консультації, навчання, діагностика, тестування, проектування тощо). Різноманітні дані можуть бути використані як вхідні дані для розробки експертної системи. Вимоги до клієнтської частини технології:

- швидкість завантаження сторінки не більше 1.8 секунди;
- мова програмування JavaScript;
- фреймворк React.js.

1.4 Висновок до розділу 1

В даному розділі було проаналізовано предметну область в сфері баз знань, визначено принцип її роботи. Проведено аналіз технологій створення WEB-інтерфейсів на прикладі Python, Ruby, Javascript, PHP та здійснено поверхневий порівняльний аналіз Показано переваги та недоліки мов програмування. Визначена постановка задачі, необхідні критерії та вимоги.

Виходячи з попередніх аналізів предметної області у сфері баз знань, програмних засобів для створення баз знань, дало можливість визначити основні проблеми та недоліки.

2 МОДЕЛІ ТА СТРУКТУРНА ОРГАНІЗАЦІЯ WEB-СЕРЕДОВИЩА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ

2.1 Класифікація експертних систем

Експертна система – інтелектуальна комп’ютерна програма, яка використовує знання та процедури логічного виведення для розв’язання проблем у певній предметній області, настільки складних, що для їх розв’язання потрібно запрошувати експерта, а також виробляє рекомендації для розв’язання цих проблем [4, 14].

Особливість систем, основаних на знаннях, полягає в тому, що вони моделюють діяльність людини, яка часто здійснюється у неформальному вигляді. Якщо методи вирішення обчислювальних задач основуються на чітких алгоритмах, обґрунтованість яких базується на понятті збіжності, то моделі подання знань мають справу з інформацією, що одержується від експертів, яка часто має якісний, і до того ж, суперечливий характер. Проте, у силу специфіки функціонування комп’ютерних систем, подібна інформація повинна бути приведена до однозначного нормалізованого вигляду. Реалізація цього відбувається на основі ідей багатозначної логіки, теорії нечітких множин і аналогічних математичних моделей [4, 15].

У той же час моделі подання знань є предметом досліджень і розробок у середовищі множини вузькоспеціалізованих фахівців, у той час як потреба в таких моделях відчувається практично у всіх предметних областях. Це пояснюється необхідністю побудови експертних систем, структура яких нерозривно пов'язана з формами подання знань, обумовлених у свою чергу особливостями предметної області [4, 14, 15, 16].

Класифікація експертних систем за деякими глобальними критеріями [4, 15]:

За типом задачі:

– Інтерпретація даних. Під інтерпретацією розуміється визначення значення даних, результати якого повинні бути узгодженими і коректними. Звичайно передбачається багатоваріантний аналіз даних. Це одна з традиційних задач для ЕС;

– Діагностика. Під діагностикою розуміється виявлення несправності в деякій системі. Несправність – це відхилення від норми. Таке трактування дозволяє з єдиних теоретичних позицій розглядати і несправність устаткування в технічних системах, і захворювання живих організмів і всілякі природні аномалії. Важливою специфікою є необхідність розуміння функціональної структури системи діагностування;

– Моніторинг. Основна задача моніторингу – безперервна інтерпретація даних в реальному масштабі часу і сигналізація про вихід тих чи інших параметрів за допустимі межі. Головні проблеми – пропуск тривожної ситуації і інверсна задача помилкового спрацьовування. Складність цих проблем – в розмитості симптомів „тривожних” сигналів;

– Проектування. Підготовка специфікацій на створення об'єктів з наперед визначеними властивостями. Під специфікацією розуміється весь набір необхідних документів: креслення, записка пояснення і т.д. Основні проблеми тут – отримання чіткого структурного опису знань про об'єкт. Для організації ефективного проектування необхідно формувати не тільки самі проектні рішення, але і мотиви їх ухвалення. Таким чином, в задачах проектування тісно пов'язуються два основні процеси: процес виведення рішення і процес пояснення;

– Прогнозування. Системи прогнозування логічно виводять вірогідні наслідки із заданих ситуацій. У системі прогнозування звичайно

використовується параметрична динамічна модель, в якій значення параметрів підганяються під задану ситуацію. Наслідки, що виводяться з цієї моделі, складають основу для прогнозів з оцінками вірогідності;

– Планування. Під плануванням розуміється знаходження планів дій, що відносяться до об'єктів, здатних виконувати деякі функції. У таких ЕС використовуються моделі поведінки реальних об'єктів з тим, щоб логічно вивести наслідки планованої діяльності;

– Навчання. Системи навчання діагностують помилки при вивченні будь-якої дисципліни за допомогою ПК і підказують правильні рішення. Вони акумулюють знання про гіпотетичного учня і його характерні помилки, потім в роботі здатні діагностувати слабкості в знаннях тих, кого навчають, і знаходити відповідні способи для їх ліквідації. Такі системи планують процес спілкування з учнем залежно від успіхів учня, з метою передачі знань.

Класифікація за зв'язком з реальним часом:

– Статичні ЕС. Розробляються в предметних областях, в яких БЗ і дані, що інтерпретуються, не змінюються в часі, вони стабільні;

– Квазідинамічні ЕС. Інтерпретують ситуацію, яка змінюється з деяким фіксованим інтервалом часу;

– Динамічні ЕС. Працюють в поєднанні з датчиками об'єктів в режимі реального часу з безперервною інтерпретацією даних, що надходять.

Класифікація за ступенем інтеграції з іншими програмами:

– Автономні ЕС. Працюють безпосередньо в режимі консультацій з користувачем для специфічних експертних задач при рішенні яких не вимагається залучати традиційні методи обробки даних;

– Гібридні ЕС. Програмний комплекс, що агрегує стандартні прикладні програми (наприклад, математичну статистику, лінійне програмування, СУБД) і засоби маніпулювання знаннями. Це може бути інтелектуальна надбудова над прикладними програмами або інтегроване середовище для вирішення складної задачі з елементами експертних знань. Не

дивлячись на зовнішню привабливість гібридного підходу, розробка таких систем є надзвичайно складною задачею. Компонування не просто різних програм, а різних методологій породжує цілий комплекс і теоретичних, і практичних труднощів.

У загальному випадку, всі системи, основані на знаннях, можна розділити на [4, 14]:

– системи, які вирішують задачі аналізу;

– системи, які вирішують задачі синтезу.

Основна відмінність задач аналізу від задач синтезу – в задачах аналізу безліч рішень може бути перераховано і включено в систему, а в задачах синтезу безліч рішень потенційно будується з рішень компонентів або під-проблем. Задачі аналізу – інтерпретація даних, діагностика, задачі синтезу – проектування, планування. Комбіновані задачі – навчання, моніторинг, прогнозування [4].

2.2 Критерії розробки експертних систем

Приступаючи до створення ЕС необхідно спочатку вирішити, чи є створення і застосування ЕС для задачі, що стоїть перед кінцевим користувачем, доцільним (доречним), виправданим і можливим [4, 14].

Критерії доцільності розробки ЕС [4, 14]:

– вирішення задачі опирається на використання операцій з символами, тобто задача не стільки пов'язана з розрахунками за формулами, скільки з логічним аналізом і перебором варіантів;

– задача не має чіткого алгоритмічного уявлення і опирається на використання евристик, які ведуть до результату, економлять набір переборів, але не дають гарантію успіху;

– задача не тривіальна;

– задача має кінцеву, прийнятну розмірність, тобто не є дуже громіздкою для вирішення на ПК;

– задача викликає великий інтерес для практики і дозволяє знаходити рішення, які за допомогою класичних методів не можуть бути одержані.

Критерії виправданості розробки ЕС (хоча б одна позиція виконується) [4]:

– вирішення задачі має високу значущість або обіцяє принести високий дохід (не обов'язково в грошовому виразі);

– досвід вирішення задач в даній предметній області поступово втрачається;

– рівень підготовки фахівців-експертів в даній предметній області надзвичайно низький;

– накопичений досвід потрібен в багатьох областях (тиражування досвіду і знань)

– експертизи проводяться в середовищі, небезпечному для людини.

Критерії можливості розробки ЕС (обов'язкові всі вимоги) [4, 14]:

– задача потребує лише інтелектуальних навиків;

– експерти можуть чітко висловити і описати використані ними методи роботи, які застосовуються у вирішенні даної задачі;

– є фахівці, чиї знання і досвід можна закласти в ЕС;

– експерти одностайні в рішенні задачі;

– задача не дуже важка;

– задача достатньо зрозуміла, якщо для вирішення задачі необхідно розробляти спеціальні методи, то ЕС не застосовується;

– задача вимагає знань і здорового глузду, які є результатом накопичення практичного досвіду, і цей досвід з якихось причин не вдається виділити і формалізувати.

Відповідно тепер можна сформулювати основні властивості експертної системи [4, 14]:

– наявність доступної бази знань;

– накопичення й зберігання високоякісного досвіду (інституціональна пам’ять);

– прогностичні можливості за рахунок включення відповідних моделей і алгоритмів обробки даних і знань;

– можливість навчання й тренування;

– здатність пояснення процесу прийняття рішень;

– наявність засобів редагування, що допомагають експертам модифікувати знання;

– наявність інтерфейсу з можливостями графічного виведення інформації.

2.3 Моделі подання знань

В галузі експертних систем подання знань означає не що інше, як систематизовану методику опису на машинному рівні того, що знає людина-експерт, яка спеціалізується в конкретній предметній області.

Існує десятки моделей (або мов) подання знань для різних предметних областей. Більшість з них може бути зведене до наступних класів [4, 15, 16]:

– Продукційні моделі (моделі, засновані на правилах);

– Семантичні мережі;

– Фрейми;

– Формальні логічні моделі.

Продукційна модель. Ця модель заснована на правилах, дозволяє представити знання у вигляді пропозицій типу “Якщо (умова), то (дія)”. Під

“умовою” розуміється деяка пропозиція-зразок, по якому здійснюється пошук у базі знань, а під “дією” - дії, виконувані при успішному результаті пошуку[4, 15].

Найчастіше висновок на такій базі знань буває прямій (від даних до пошуку мети) або зворотний (від мети для її підтвердження — до даних). Дані — це вихідні факти, що зберігаються в базі фактів, на підставі яких запускається машина висновку або інтерпретатор правил, що перебирає правила із продукційної бази знань. Продукційна, модель так часто застосовується в промислових експертних системах, оскільки залучає розробників своєю наочністю, високої модульності, легкістю внесення доповнень і змін і простотою механізму логічного висновку[4, 15].

Є велика кількість програмних засобів, що реалізують продукційний підхід (наприклад, мови високого рівня CLIPS й OPS 5; "оболонки" або "порожні" ЕС - EXSYS Professional і Kappa, інструментальні системи KEE, ARTS, PIES), а також промислових ЕС на його основі (наприклад, ЕС, створених засобами G2[4, 15].

Семантична мережа. Це орієнтований граф, вершини якого — поняття, а дуги — відносини між ними[4, 15].

Як поняття звичайно виступають абстрактні або конкретні об'єкти, а відносини це зв'язку типу: "це" ("AKO — A-KindOf, "is" або "елемент класу"), "має частиною" ("has part"), "належить", "любить"[4, 15].

Класифікації семантичних мереж, пов'язаних з типами відносин між поняттями[4, 15]:

По кількості типів відносин:

– однорідні (з єдиним типом відносин);

– неоднорідні (з різними типами відносин).

По типах відносин:

– бінарні (у які відносини зв'язують два об'єкти);

– N-арні (у які є спеціальні відносини, що зв'язують більше двох понять).

Найбільше часто в семантичних мережах використаються наступні відносини:

– елемент класу (троянда це квітка);

– атрибутивні зв'язки /мати властивість (пам'ять має властивість — об'єм);

– значення властивості (кольори має значення — жовтий);

– приклад елемента класу (троянда, наприклад — чайна);

– зв'язку типу "частина-ціле" (велосипед включає кермо);

– функціональні зв'язки (обумовлені звичайно дієсловами "робить", “впливає"...);

– кількісні (більше, менше, дорівнює…);

– просторові (далеко від, близько від, за, під, над…);

– часові (раніше, пізніше, протягом…);

– логічні зв'язки (і, або, не) і ін.

Мінімальний склад відносин у семантичній мережі такий:

– елемент класу або АКО;

– атрибутивні зв'язки /мати властивість;

– значення властивості.

Недоліком цієї моделі є складність організації процедури організації висновку на семантичній мережі[4, 15].

Ця проблема зводиться до нетривіальної задачі пошуку фрагмента мережі, що відповідає деякої підмережі, що відбиває поставлений запит до бази[4, 15].

Для реалізації семантичних мереж існують спеціальні мережні мови, наприклад, NET, мова реалізації систем SIMER + MIR й ін. Широко відомі експертні системи, що використають семантичні мережі як мову подання знань - PROSPECTOR, CASNET, TORUS[12, 14].

Фрейми. Це абстрактний образ для подання стереотипу об'єкта, поняття або ситуації. Інтуїтивно зрозуміло, що під абстрактним образом розуміється деяка узагальнена й спрощена модель або структура. Наприклад, проголошення вголос слова "кімната" породжує в почутому образ кімнати: "житлове приміщення із чотирма стінами, підлогою, стелею, вікнами й дверима, площею 6—20 м2 ". Із цього опису нічого не можна забрати (наприклад, забравши вікна, ми одержимо вже прикомірок, а не кімнату), але в ньому є "дірки" або "слоти"— це незаповнені значення деяких атрибутів — наприклад, кількість вікон, кольори стін, висота стелі, покриття підлоги й інші[4, 15].

У теорії фреймів такий образ кімнати називається фреймом кімнати. Фреймом також називається й формалізована модель для відображення образу.

Розрізняють фрейми-зразки або прототипи, що зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що надходять. Модель фрейму є досить універсальної, оскільки дозволяє відобразити все різноманіття знань про світ через[4, 15]:

– фрейму-структури, що використаються для позначення об'єктів і понять (позика, застава, вексель);

– фреймів-ролі (менеджер, касир, клієнт);

– фрейми-сценарії (банкрутство, збори акціонерів, святкування);

– фреймів-ситуації (тривога, аварія, робочий режим пристрою) і ін.

Існує кілька способів одержання слотом значень у фреймі екземплярі:

– за замовчуванням від фрейму-зразка (Default-значення);

– через спадкування властивостей від фрейму, зазначеного в слоті AKO;

– по формулі, зазначеної в слоті;

– через приєднану процедуру;

– явно з діалогу з користувачем;

– з бази даних.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж - так називане спадкування властивостей. І у фреймах, і в семантичних мережах спадкування відбувається по Ако-зв’язкам(A-Kind-Of = це). Слот АКО вказує на фрейм більше високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів[4, 15].

Основною перевагою фреймів як моделі подання знань є те, що вона відбиває концептуальну основу організації пам'яті людини, а також її гнучкість і наочність[4, 15].

Спеціальні мови подання знань у мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова "оболонка" Карра й інші програмні засоби дозволяють ефективно будувати промислові ЕС. Широко відомі такі фрейм-орієнтовані експертні системи, як ANALYST, МОДИС, TRISTAN, ALTERID[4, 15].

Формально логічні моделі. Традиційно в поданні знань виділяють формальні логічні моделі, засновані на класичному вирахуванні предикатів 1-го порядку, коли предметна область або задача описується у вигляді набору аксіом. Реальне вирахування предикатів 1-го порядку в промислових експертних системах практично не використається. Ця логічна модель застосовна в основному в дослідницьких "іграшкових" системах, тому що пред'являє дуже високі вимоги й обмеження до предметної області[4, 15, 26].

Формально-логічні системи задаються двома способами: матричним і аксіоматичним. Матричний спосіб побудови формально-логічної системи означає[4, 15, 26]:

– побудову матриці (таблиці) істинності для логічних постійних або пропозиційних зв'язок кон'юнкції, диз'юнкції, імплікації, еквівалентності, заперечення, на підставі якої задаються функції істинності для формул, побудованих за певними правилами;

– логічні закони визначаються як елементи певної системи.

Аксіоматичний спосіб побудови формально-логічної системи означає побудову числення певного типу логічної теорії, що надається більша строгість виведення одного висловлювання (формули) з іншого відповідно до правил, встановлених на підставі логічних законів[4, 15, 26].

Розрізняють два етапи в розвитку символічної логіки, в межах якої будували різні формально-логічні системи, які отримали назву "класична символічна логіка" та "некласична символічна логіка"[4, 15, 26].

У межах класичної символічної логіки (формально-логічних систем, побудованих методом формалізації) розрізняють такі рівні формування логічного знання[4, 15, 26]:

– логіка висловлювань (логічне числення першого порядку);

– розширення логіки висловлювань через створення нової формально-логічної системи, яка отримала назву "логіка предикатів" ("логічне числення другого порядку").

На кожному рівні формування логічного знання розробляють особливу формалізовану мову, за допомогою якої створюють формально-логічну систему. Формалізована мова першого і другого порядку - мова символічної логіки, що складається з алфавіту цієї мови. До формалізованої мови першого порядку належить мова логіки висловлювань, а до формалізованої мови другого порядку - мова логіки предикатів[4, 15, 26].

2.4 Обґрунтування вибору моделі подання знань

Найбільше поширення одержала продукційна модель подання знань. При її використанні база знань складається з набору правил, а програма, що управляє перебором правил, називається машиною висновку(інтерпретатор) [4, 15].

У загальному випадку продукційна система включає наступні компоненти[4, 15]:

– базу продукційних правил;
– базу даних (робочу пам'ять);
– інтерпретатор.

Множину продукційних правил утворить базу правил, кожне з який представляє відособлений фрагмент знань про розв'язувану проблему. Психологи називають такі фрагменти чанками. Вважається, що чан – це об'єктивно існуюча одиниця знань, що виділяється людиною в процесі пізнання навколишнього світу. [4, 15]

Передумова правила часто розглядається як зразок. Це деяка інформаційна структура, що визначає узагальнену ситуацію (умова, стан і т.п.) навколишньої дійсності, при якій активізується правило. Робоча пам'ять (глобальна база даних) відбиває конкретні ситуації (стану, умови), що виникають у зовнішньому середовищі. Інформаційна структура, що представляє конкретну ситуацію зовнішнього середовища в робочій пам'яті, називається образом[4, 14].

Інтерпретатор реалізує логічний вивід. Процес виводу є циклічним і називається пошуком за зразком. Розглянемо його в спрощеній формі. Поточний стан предметної області, що моделюється відбивається в робочій пам'яті у вигляді сукупності образів, кожний з яких представляється за допомогою фактів. Робоча пам'ять ініціалізується фактами, що описують задачу. Потім вибираються ті правила, для яких зразки, що представляються передумовами правил, порівнянні з образами в робочій пам'яті. Данні правила утворюють конфліктну множину. Усі правила, що входять у конфліктну множину можуть бути активізовані. Відповідно до обраного механізму дозволу конфлікту активізується одне з правил. Виконання дії, що міститься у виводу правила, приводить до зміни стану робочої пам'яті. Надалі цикл керування виведенням повторюється. Зазначений процес завершується, коли не виявиться правил, передумови яких порівнянні з образами робочої пам'яті (рис. 2.1) [4, 14].

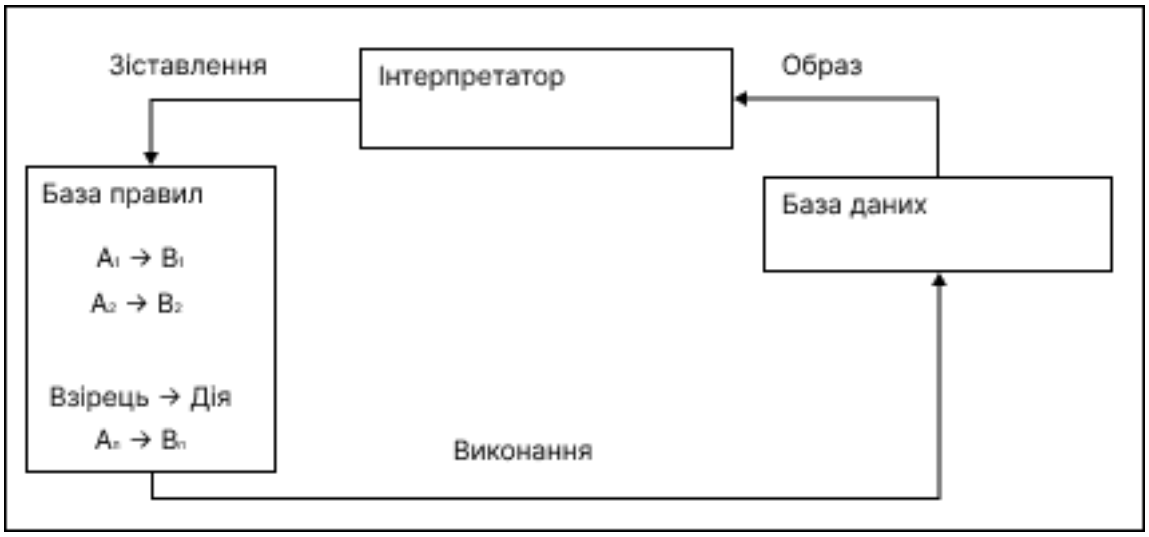


Рисунок 2.1 – Схема продукційної системи

Широке застосування продукційних моделей визначається наступними основними перевагами[4, 14]:

– універсальністю, практично кожна область знань може бути представлена в продукційній формі;

– модульністью, кожна продукція являє собою елемент знань про предметну область, видалення одних і додавання інших продукцій виконується незалежно;

– декларативністю, продукції визначають ситуації, предметної області, а не механізму керування;

– природністю процесу виводу, що багато в чому аналогічний процесу міркувань експерта;

– асинхронністю і природним паралелізмом, що; робить їх дуже перспективними для реалізації на рівнобіжних ПК.

Однак продукційні моделі мають недоліків[4, 14]:

– процес виводу має низьку ефективність, тому що при великому числі продукцій значна частина часу затрачається на невиробничу перевірку умов застосування правил;

– перевірка несуперечності системи продукцій стає дуже складною через недетермінованості вибору виконуваної продукції з конфліктної множини.

Отже, недоліки для нашої задачі не є суттєвими, а переваги важливими і саме тому ми обираємо продукційну модель подання знань.

2.5 Структурна організація WEB-середовища інформаційної технології для розробки експертних систем

Розглянемо загальну схему роботи WEB-середовища для розробки експертних систем(рис. 2.2).

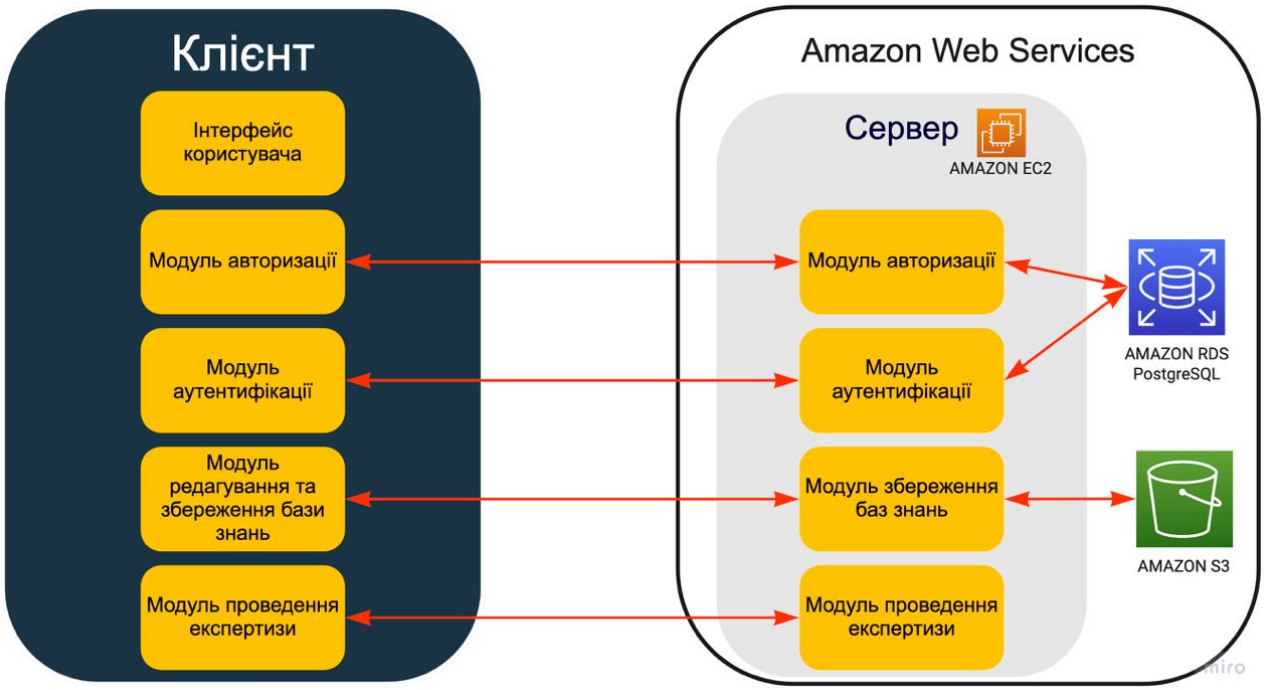


Рисунок 2.2 – Загальна схема роботи WEB-середовища для розробки експертних систем

Детальніше розглянемо клієнтську частину, а саме «інтерфейс користувача», який складається з декількох модулів:

– модуль авторизації;

- модуль автентифікації;
- модулі створення, редагування та збереження бази знань;
- модуль проведення експертизи.

Модулі авторизації та автентифікації пов’язані з усіма іншими модулями, через те, що поки користувач не буде авторизований у системі, він не матиме змоги взаємодіяти з системою.

Модуль створення. Дозволяє створювати бази знань одразу ж у WEB-середовищі. Модуль редагування, працює так як модуль створення, єдина відмінність в тому, що при редагуванні користувач обирає певний файл для редагування та модифікує його базу знань чи назву файлу.

Модуль збереження. Працює в поєднані з модулями створення та редагування, зберігає будь-які зміни, які стосуються файлу.

Модуль експертизи. Працює в поєднані з модулями створення та редагування, проводить консультацію на основі правил з обраного файлу.

2.6 Висновок до розділу 2

В другому розділі було наведено класифікацію експертних систем: за типом задачі, за зв’язком з реальним часом та за ступенем інтеграції з іншими програмами. Розглянуто системи основані на знаннях та розділено їх на системи, які вирішують задачі аналізу та синтезу. Наведено відмінність задач аналізу від задач синтезу. Визначено критерії розробки, такі як доцільність, визначеність та можливість. Наведено основні властивості експертної системи. Проаналізовано моделі подання знань та наведено їх переваги та недоліки. Обґрунтовано доцільність використання продукційної моделі. Наведено схему подукційної системи. Наведено структурно-організаційну схему та описано взаємодію головних модулів WEB-середовища.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСНОЇ ЧАСТИНИ WEB-СЕРЕДОВИЩА ДЛЯ СТВОРЕННЯ ЕКСПЕРТНИХ СИСТЕМ

3.1 Обґрунтування вибору середовища для створення експертної системи

Кожного дня з’являються оновлення існуючих чи принципово нові середовища розроблення, але найбільшим попитом користуються вже перевірені часом рішення, як, наприклад, Microsoft Visual Studio, WebStorm та Atom [17-19].

Visual Studio – одна з найпопулярніших серій середовищ розроблення, створена компанією Microsoft, що дозволяє розроблювати WEB-сайти та WEB-застосунки, консольні програми та програми із графічним інтерфейсом, для усіх платформ, що підтримуються Microsoft. Основою Visual Studio є редактор вихідного коду, що підтримує технологію IntelliSense та найпростіші засоби рефакторингу коду. IntelliSense – це набір функцій, що надає відомості про код безпосередньо в редакторі, а в деяких випадках автоматично генерує невеликі уривки коду. Фактично, технологія представляє собою базову документацію, що вбудована в редактор. Функції IntelliSense залежать від мови, що використовується. Серед інструментів цього середовища розроблення є [17]:

- вбудований відлагоджувач;
- редактор форм графічних інтерфейсів;
- редактор WEB-сторінок;
- дизайнер класів.

Одним із важливих інструментів для WEB-програмування є вбудований WEB-сервер, який дозволяє запускати WEB-сайт безпосередньо з середовища програмування, а також підвищує безпеку, виключаючи ймовірність отримання доступу до тестового серверу із зовнішніх пристроїв, оскільки тестовий сервер може мати з'єднання лише з локальним комп'ютером. Окрім наявної великої кількості вбудованих засобів для роботи з кодом, Visual Studio підтримує

підключення сторонніх плагінів що дозволяє значно розширити функціональність різноманітних засобів, починаючи від технології IntelliSense та закінчуючи доданням системи контролю версій. Основним недоліком Visual Studio є її «важкість» - через велику кількість функціональних можливостей відкриття, завантаження та запуск проектів може потребувати значних ресурсів комп'ютера [17].

WebStorm - 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 перед-установленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безкоштовно [18].

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart. 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 [18].

Основні можливості [18]:

– 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;

– 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»);

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

– Можливості Zen Coding і Emmet.

Підтримка:

– Web(Angular, React.js,Vue.js);

– Server(Node.js, Meteor);

– Mobile(Ionic, Cordova, React Native);

– Desktop(Electron).

LiveEdit — можливість WebStorm, котра з'явилася з версії 5 і дозволяє одночасно редагувати код html, css або javascript і бачити, як результат відображається в браузері . Для цього потрібна підтримка такої можливості з боку браузера, тому WebStorm при установці ставить плагін для Google Chrome [18].

WebStorm підтримує завантаження застосунків у node.js. Також підтримується повний набір функцій редагування застосунків на javascript — як для виконання на сервері, так і в браузері: автодоповнення, навігація по коду, рефакторінг і перевірка на помилки [18].

Для node.js підтримується також виведення повідомлень node.js на окрему вкладку в IDE [18].

Мови стилів LESS, Sass і SCSS, які розширюють можливості описів стилів у CSS, повністю підтримуються в WebStorm, зокрема, підтримується рефакторинг коду для них, коли треба змінити вираз (наприклад, #a9a9a9) на змінну (наприклад @grey), для того, щоб зробити код читанішим і простіше перевизначати параметри (наприклад, шляхом присвоєння їм значення @grey: #a9a9a9) [18].

У версіях від WebStorm 5 для CoffeeScript є навігація за кодом, автодоповнення, рефакторинг, підсвічування синтаксису і перевірка на помилки [18].

Atom розроблений компанією «GitHub» вільний текстовий редактор і редактор коду, надає засоби кросплатформового редагування коду, включає вбудований пакетний менеджер і інтерфейс навігації файловою системою, надає засоби для одночасної спільної роботи з кодом, має інтелектуальну систему автодоповнення вводу, надає режими сумісності з Vim і Emacs, підтримує API для розробки розширень. Кілька файлів можуть бути відкриті в різних вкладках і одночасно показані з використанням вертикального або горизонтального розбиття панелей. Інтерфейс може налаштовуватися через теми оформлення, підтримуються вкладки, закладки, розумний контекстний пошук коду, схлопування блоків коду, одночасне використання декількох курсорів і областей виділення, наочна позначка змін, автодоповнення та перевірка коду для різних мов (Ruby, Python, PHP, Objective-C, C/C++, JavaScript, Java, Go тощо). Для формування стате та документації може бути використана розмітка Markdown [19].

Функціональність редактора формується внаслідок надання набору пакетів-доповнень, для установки яких пропонується вбудований пакетний менеджер apm, схожий на npm від проєкту Node.js. Формат пакунків аналогічний npm і відрізняється наданням деяких додаткових блоків для визначення меню, стилів, клавіатурних комбінацій, завдання логіки активації. Розробка доповнень мало чим відрізняється від створення програми для Node.js, у тому числі доступні всі модулі Node.js, а також популярні JavaScript-бібліотеки, такі як jQuery, Underscore і SpacePen. Через доповнення реалізовані всі функції, що виходять за рамки базового редагування коду, в тому числі панелі, підсвічування синтаксису, оформлення інтерфейсу, форми роботи з файлами тощо. Крім базових доповнень надається каталог сторонніх пакетів, в

якому вже присутні понад дві тисяч доповнень і майже сімсот тем оформлення [19].

Основу Atom становить компонент Electron (раніше Atom Shell), що становить собою засноване на Chromium і Node.js ядро, поверх якого реалізований редактор. Electron поставляється у формі самодостатнього фреймворку, який можна використовувати для створення довільних користувацьких застосунків, логіка роботи якого визначається на JavaScript, HTML і CSS, а функціональність може бути розширена через систему доповнень. Розробникам доступні модулі Node.js, а також розширений API для формування нативних діалогів, інтеграції застосунків, створення контекстних меню, маніпуляції вікнами, взаємодії з підсистемами Chrome [19].

Необхідність використання власного браузерного ядра на основі Chromium, замість оформлення редактора у формі WEB-застосунку, що працює у звичайному браузері, обумовлена необхідністю реалізації додаткових можливостей, недоступних через звичайний Web API. Наприклад, Atom надає вбудований файловий менеджер і гнучкі засоби пошуку файлів, які неможливо реалізувати при використанні звичайних WEB-застосунків. У редакторі також безпосередньо використовуються деякі внутрішні підсистеми Chromium, такі як рушій обробки регулярних виразів і нативні елементи формування діалогів. Крім того, оскільки компоненти Atom завжди виконуються локально, спрощується розробка доповнень, звернення до ресурсів і розмежування доступу [19].

3.2 Обґрунтування вибору мови програмування

Вибір потрібної мови програмування може забезпечити рішення, які є стислими, легко налагоджувати, легко розширити, легко документувати та легко виправляти. Фактори, які необхідно враховувати при виборі мови програмування:

– Цільова платформа;

– Еластичність мови;

– Час виробництва;

– Підтримка та спільнота.

Найважливішим фактором, який необхідно враховувати, є платформа, на якій буде працювати програма. Подумайте з точки зору мови Java і C. Якщо програма написана на мові C і має працювати на машинах з операційними системами Windows і Linux, то це вимагатиме компіляторів платформ і двох різних виконуваних файлів. За допомогою мови Java, згенерований код байта достатній для запуску програми на різноманітних машинах з інстальованою віртуальною машиною Java (JVM) [6, 8, 21, 22].

Подібний аргумент застосовується до WEB-сайтів. Всі вони повинні виглядати і працювати однаково у всіх браузерах. Використовуючи теги CSS, HTML, не перевіряючи сумісність із WEB-переглядачем, змусить один і той самий сайт виглядати і поводитися по-різному в браузерах [23, 24].

Перед тим як почати розробку WEB-додатку потрібно вибрати найкомфортнішу технологію. Для порівняння було обрано найпопулярніші мови програмування, на даному відрізку часу, для створення WEB-орієнтованих додатків. Це мови Python і JavaScript [9, 12].

Python - це мова з простим синтаксисом і потужним набором бібліотек. Це інтерпретована мова, з багатим середовищем програмування, включаючи надійний відладчик і профайлер. Python легка у вивченні мова програмування, проте широко використовується в багатьох наукових напрямках для дослідження даних [9].

Простий синтаксис Python дозволяє легко вчитися і легко писати, оскільки розробники часто вибирають писати на Python, оскільки вони відчувають себе більш продуктивними. Оскільки основна реалізація розробки для Python буде працювати в різноманітних середовищі, програмістам легко почати

користуватися нею, незважаючи на свою операційну систему. Багато розробників звертаються до Python навіть, якщо до цього були незнайомі з нею і рідко знаходять, що це було поганим рішенням [9].

Хоча Python може бути швидким через його базову реалізацію C, вона може бути повільнішою, оскільки вона є вбудованою мовою. Накладні витрати на виконання можуть бути проблемою у великих додатках. Розгортання WEB-додатків Python не настільки просте, як на інших мовах, як PHP, і WEB-фреймворки Python ще не досягли корпоративного рівня, такого як Java або .NET, коли йдеться про такі функції, як кластеризація, перехід на другий план і керування [9, 10, 21, 22, 25].

Javascript - це мова сценаріїв, яка в основному використовується в Інтернеті. Вона використовується для поліпшення HTML-сторінок і зазвичай знаходиться в HTML-коді. JavaScript є інтерпретованою мовою. Таким чином, її не потрібно компілювати. JavaScript надає WEB-сторінці інтерактивності та динамічності. Це дозволяє сторінкам реагувати на події, демонструвати спеціальні ефекти, приймати змінний текст, перевіряти дані, створювати файли cookie, виявляти браузер користувача тощо [12, 23].

Python можна використовувати для створення WEB-додатків на стороні сервера. Хоча для створення WEB-додатків не потрібна WEB-структура, рідко розробники не використовують існуючі бібліотеки з відкритим кодом для прискорення прогресу в роботі своїх програм [9].

Python не використовується в WEB-переглядачі. Мова, яка виконується в браузерах, таких як Chrome, Firefox і Internet Explorer, є JavaScript. Такі проекти, як pyjs, можуть бути зібрані з Python на JavaScript. Однак більшість розробників Python записують свої WEB-програми за допомогою комбінації Python і JavaScript. Python виконується на стороні сервера, а JavaScript завантажується на клієнт і запускається WEB-браузером [9, 12].

Також проблемою Python є однопоточність, що буде значно гальмувати роботу сервера. Звичайно, можна створити декілька потоків виконання, але в JavaScript є кращий аналог – Event Loop [9, 12].

Уже з цього короткого порівняльного аналізу зрозуміло, що кращим рішенням буде обрати JavaScript як на клієнті так і на сервері. Завжди легше й зрозуміліше користуватися однією мовою програмування, особливо якщо написання й підтримка WEB-орієнтованого додатку буде лежати на одній людині.

3.3 Розробка алгоритму роботи програмного продукту

Програмний продукт містить 3 основних частини:

- процес реєстрації / авторизації користувача;
- процес створення / завантаження бази знань;
- процес опитування користувача.

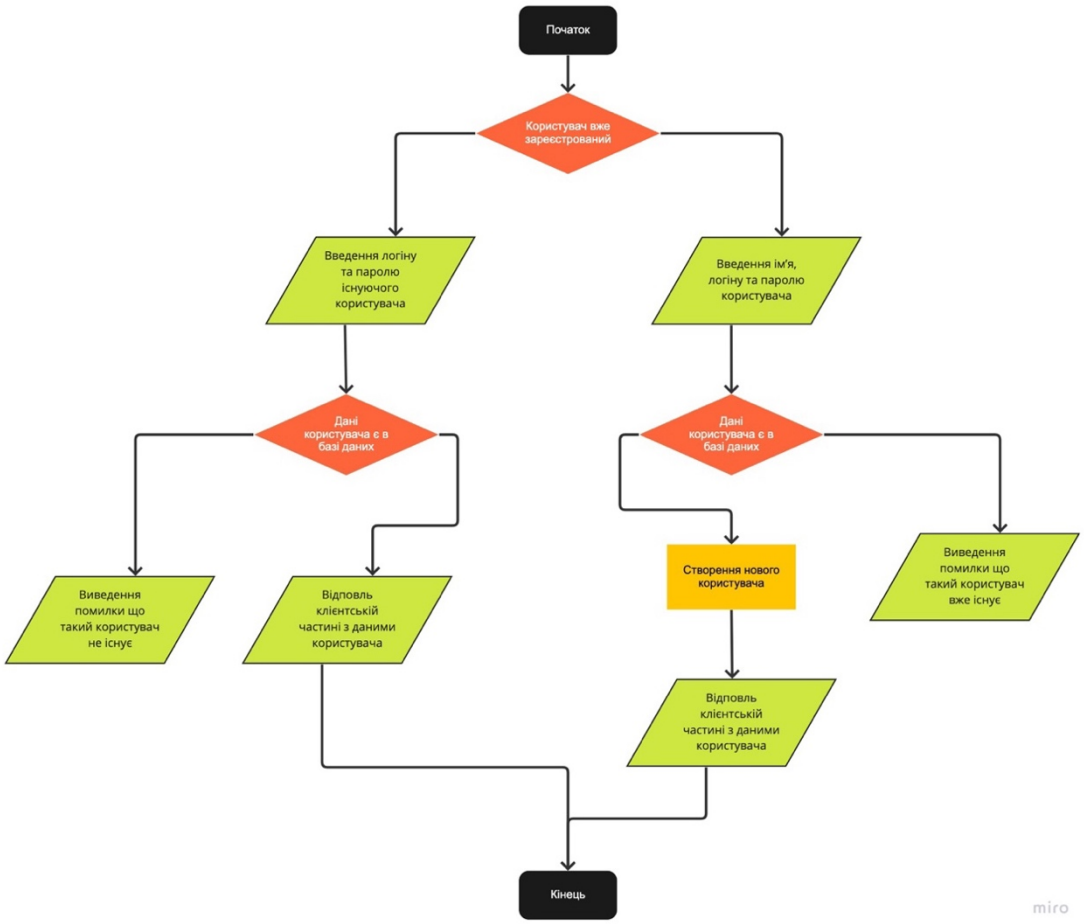


Рисунок 3.1 – Схема алгоритму процесу реєстрації / авторизації користувача

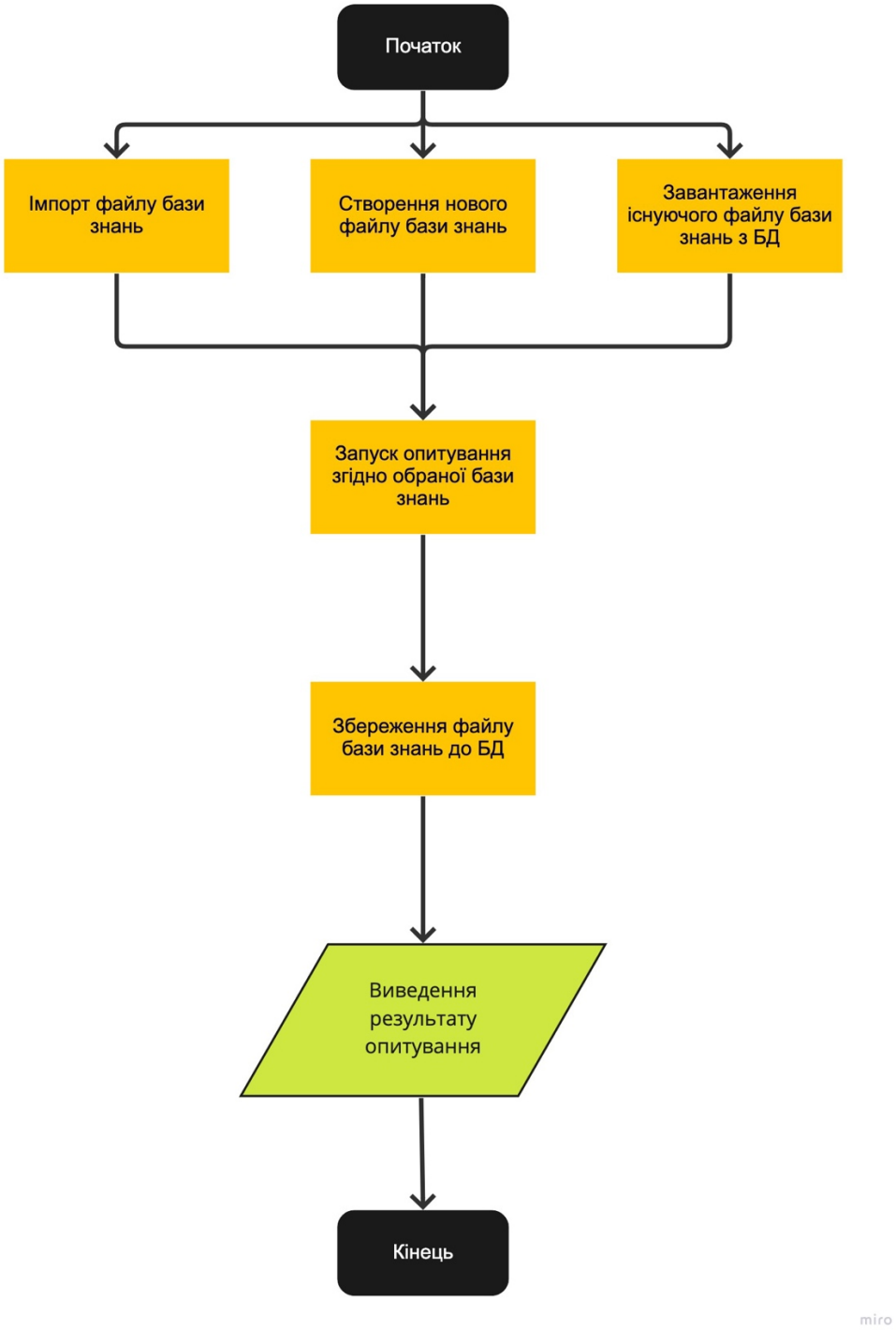


Рисунок 3.2 – Схема алгоритму процесу створення / завантаження бази знань

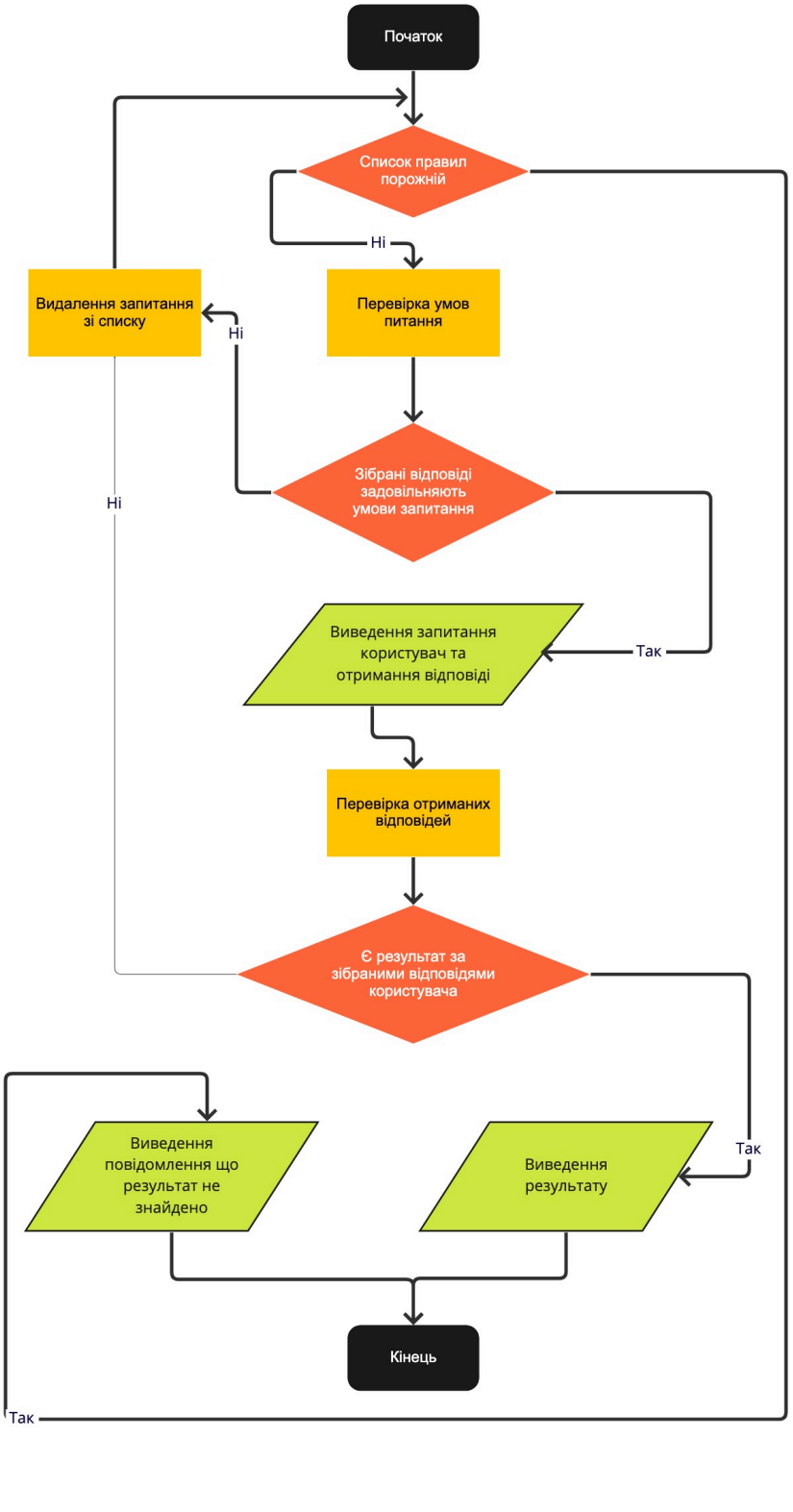
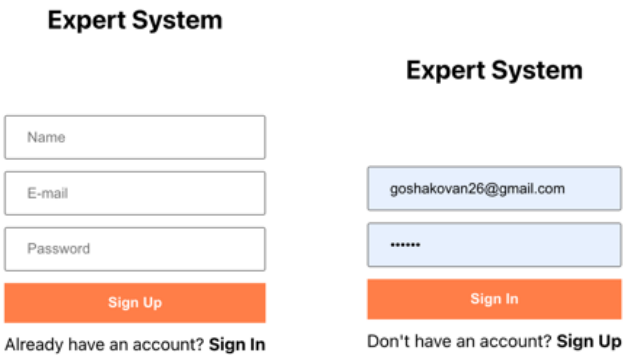


Рисунок 3.3 – Схема алгоритму процесу опитування користувача

3.4 Програмна реалізація експертної системи

Розроблене WEB-середовище дозволяє користувачу створювати та редагувати експертні системи маючи в наявності лише браузер та підключення до інтернету.

Для роботи з WEB-середовищем користувачу необхідно зареєструватись в системі(рис. 3.4а) або пройти автентифікацію за наявності створеного облікового запису(рис. 3.4б).



а) б)

Рисунок 3.4 – Вигляд вікон:
а – реєстрації користувача; б – автентифікації

Логіку переключення вікон реєстрації та автентифікації можна побачити на рисунку 3.5.

```
function Auth() {
    const [authData, setAuthData] = useContext(AuthContext)
    const {signInStatus} = authData

    const signInStatusHandler = () => {
        setAuthData({
            ...authData,
            signInStatus: !signInStatus
        })
    }

    return (
        <AuthWrapper>
            <AuthWrapperTitle>
                Expert System
            </AuthWrapperTitle>
            <AuthWrapperWindow>
                {
                    signInStatus
                        ? <SignIn/>
                        : <SignUp/>
                }
                <AuthWrapperWindowSwitch>
                    {
                        signInStatus
                            ? <>Don't have an account? <b onClick={signInStatusHandler}>Sign Up</b></>
                            : <>Already have an account? <b onClick={signInStatusHandler}>Sign In</b>
</>                 }
                </AuthWrapperWindowSwitch>
            </AuthWrapperWindow>
        </AuthWrapper>
    )
}

export default Auth
```

Рисунок 3.5 – Логіка переключення вікон реєстрації та автентифікації

Після успішного входу до WEB-середовища, користувач потрапляє на головну сторінку(рис. 3.6). На головній сторінці для користувача доступні такі опції як завантаження до WEB-середовища власної бази знань, завантаження з WEB-середовища поточної бази знань, початок консультації та вихід з облікового запису.

Кодове представлення головної сторінки можна побачити на рисунку 3.7

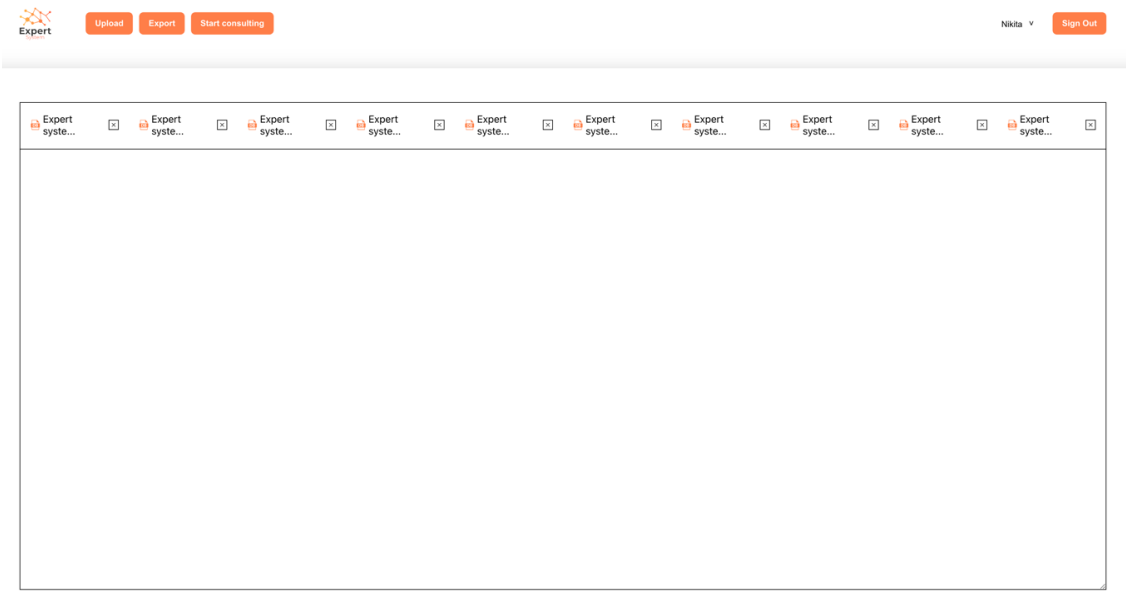


Рисунок 3.6 – Головна сторінка

```
function Main(props) {
    const {
        code,
        setCode,
    } = props
    const [splitter, setSplitter] = useContext(SplitterContext)

    return (
        <AppWrapperSplitter
            column={splitter?.column}
        >
            {
                splitter?.spaces?.map((space, spaceIndex) => (
                    <AppWrapperMain>
                        <Tabs list={space} listIndex={spaceIndex}/>
                        <AppWrapperMainFirstArea
                            column={splitter?.column}
                        >
                            <Editor
                                value={code}
                                onValueChange={(code) => setCode(code)}
                                highlight={(code) => highlight(code, languages.js)}
                                padding={10}
                                style={{
                                    fontFamily: '"Fira code", "Fira Mono",
monospace',                                 fontSize: 12,
                                    width: '104%',
                                    height: '100%',
                                    overflowY: 'auto',
                                    boxSizing: 'content-box'
                                }}
                            />
                        </AppWrapperMainFirstArea>
                    </AppWrapperMain>
                ))
            }
        </AppWrapperSplitter>
    )
}

export default Main
```

Рисунок 3.7 – Кодове представлення головної сторінки

При відкриті, створені або завантажені бази знань, файл бази знань буде відображатись у верхній частині робочої зони. Якщо буде відкрито до 10 файлів одночасно, то для переключення між ними користувачу буде потрібно натиснути на потрібний йому файл. Також у кожного файла є меню взаємодії з такими опціями як перенесення файлу до окремої робочої зони(робоча зона буде ділитися навпіл в залежності від обраного перенесення файлу) та перейменування файлу(рис. 3.8).

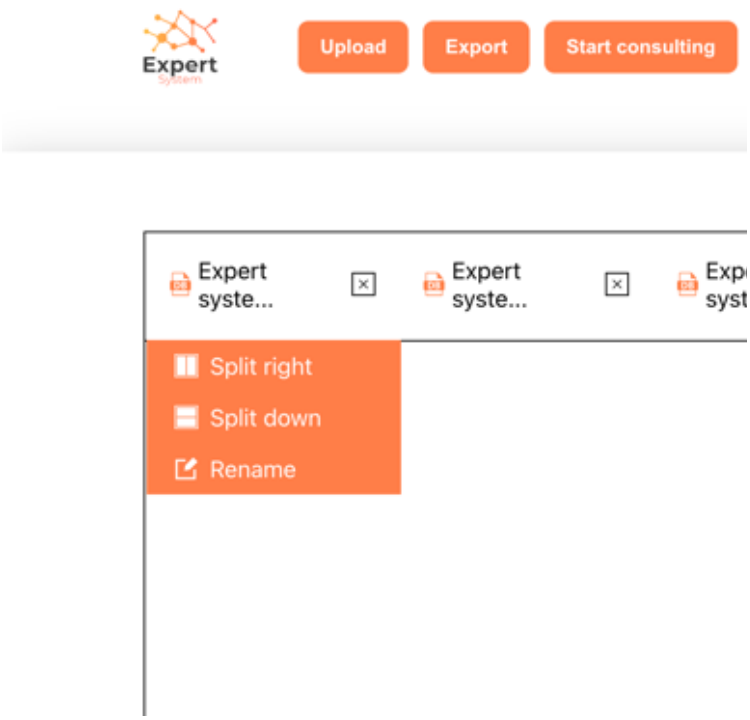


Рисунок 3.8 – Меню взаємодії з файлом

Кодове представлення файлу, меню взаємодії з файлом та логіку взаємодії можна побачити на рисунку 3.9.

```
function Tab({list, listIndex, es}) {
    const tabRef = useRef(null)
    const [splitter, setSplitter] = useContext(SplitterContext)
    const [menuStatus, setMenuStatus] = useState(false)

    const tabHandler = (id) => {
        const filteredList = list
            ?.map((es) => ({
                ...es,
                active: id === es?.id
            }))
        setSplitter({
            ...splitter,
            spaces: splitter.spaces.map((space, spaceIndex) =>
                spaceIndex === listIndex
                    ? filteredList
                    : space
            )
        })
    }

    const tabMenuHandler = (e) => {
        e?.preventDefault()
        setMenuStatus(!menuStatus)
    }

    const closeTabHandler = (e, id) => {
        e.stopPropagation()
        const filteredList = list
            ?.filter((es) => id !== es?.id)

        setMenuStatus(false)
        setSplitter({
            ...splitter,
            spaces: splitter.spaces.map((space, spaceIndex) =>
                spaceIndex === listIndex
                    ? filteredList
                    : space
            )
        })
    }

    return (
        <TabsWrapperTab
            key={es?.id}
            ref={tabRef}
            menuStatus={menuStatus}
            activeTabStatus={es?.active}
            onClick={() => tabHandler(es?.id)}
            onContextMenu={(e) => tabMenuHandler(e, es)}
        >
            {
                menuStatus &&
                <TabMenu
                    top={tabRef?.current?.offsetHeight}
                    left=
{tabRef?.current?.getBoundingClientRect()?.left}
                    es={es}
                    listIndex={listIndex}
                    tabMenuHandler={tabMenuHandler}
                />
            }
            <TabsWrapperTabIcon
                src={DBColorIcon}
                alt={'Database icon'}
            />
            <TabsWrapperTabTitle
                title={es?.title}
            >
                {es?.title?.slice(0, 12)}...
            </TabsWrapperTabTitle>
            <TabsWrapperTabClose
                activeTabStatus={es?.active}
                src={CloseIcon}
                alt={'Close'}
                onClick={(e) => closeTabHandler(e, es?.id)}
            />
        </TabsWrapperTab>
    )
}

export default Tab
```

Рисунок 3.9 – Кодове представлення файлу, меню взаємодії з файлом та логіку взаємодії

У кожного користувача є можливість зберігати у своєму обліковому записі до 10 баз знань загальним обсягом до 50мб. На головній сторінці WEB-середовища користувач має змогу побачити збережені бази знань та при необхідності відкрити їх для редагування, видалення або початку консультації(рис. 3.10).

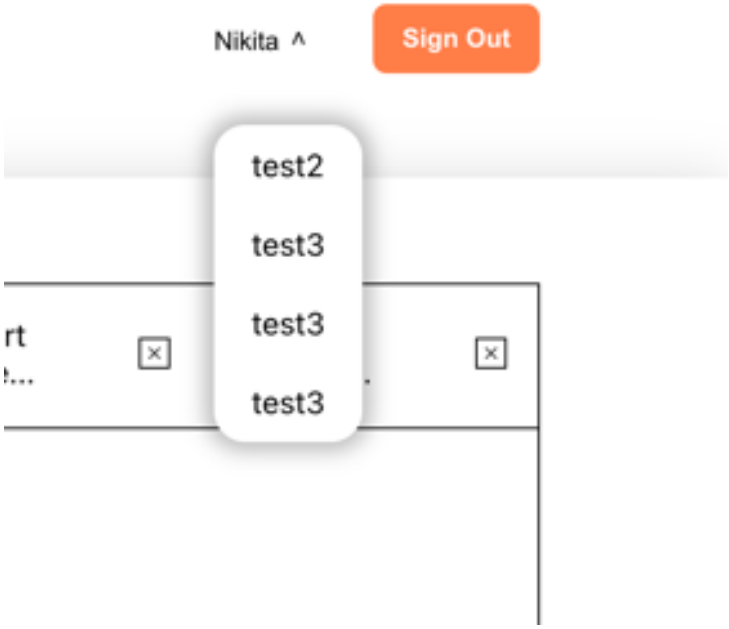


Рисунок 3.10 – Список завантажених файлів до профілю користувача

Кодове представлення списку завантажених файлів, а також логіку виходу з системи можна побачити на рисунку 3.11.

```
function Header(props) {
    const {
        uploadButtonRef,
        uploadFileHandler,
        startConsultationHandler,
        controlUploadButtonHandler,
        exportDataHandler,
    } = props
    const [authData, setAuthData] = useContext(AuthContext)
    const {user: {name}} = authData
    const [expertSystemsList, setExpertSystemsList] =
useContext(ExpertSystemsContext)
    const [listStatus, setListStatus] = useState(false)

    const signOutHandler = () => {
        setAuthData({
            ...authData,
            authorised: false,
            signInStatus: true
        })
        localStorage.clear()
    }

    const listHandler = () => setListStatus(!listStatus)

    return (
        <HeaderWrapper>
            <HeaderWrapperLogo
                src={Logo}
                alt={'Logo'}
            />
            <HeaderWrapperButtons>
                <HeaderWrapperUpload
                    type={'file'}
                    ref={uploadButtonRef}
                    onChange={uploadFileHandler}
                />
                <HeaderWrapperUploadButton
                    onClick={controlUploadButtonHandler}
                >
                    {'Upload'}
                </HeaderWrapperUploadButton>
                <HeaderWrapperStartConsultation
                    onClick={exportDataHandler}
                >
                    {'Export'}
                </HeaderWrapperStartConsultation>
                <HeaderWrapperStartConsultation
                    onClick={startConsultationHandler}
                >
                    {'Start consulting'}
                </HeaderWrapperStartConsultation>
            </HeaderWrapperButtons>
            <HeaderWrapperAuthButtons>
                <HeaderWrapperAuthButtonsWrapper>
                    <HeaderWrapperAuthButtonsUsername
                        isEmptyList={!expertSystemsList.length}
                        status={expertSystemsList.length && listStatus}
                        onClick={listHandler}
                    >
                        {name}
                    </HeaderWrapperAuthButtonsUsername>
                    <HeaderWrapperAuthButtonsWrapperList
                        status={expertSystemsList.length && listStatus}
                    >
                        {
                            expertSystemsList?.map(({title, id}, i) =>
                                <HeaderWrapperAuthButtonsWrapperListItem
                                    key={id}
                                >
                                    {title || `Expert system ${i + 1}`}
                                </HeaderWrapperAuthButtonsWrapperListItem>
                            )
                        }
                    </HeaderWrapperAuthButtonsWrapperList>
                </HeaderWrapperAuthButtonsWrapper>
                <HeaderWrapperAuthButtonsSignOut
                    onClick={signOutHandler}
                >
                    {'Sign Out'}
                </HeaderWrapperAuthButtonsSignOut>
            </HeaderWrapperAuthButtons>
        </HeaderWrapper>
    )
}

export default Header
```

Рисунок 3.11 – Кодове представлення списку завантажених файлів та логіки виходу з системи

При натисканні на кнопку початку консультації, відкривається модальне вікно для введення ключового слова по якому буде здійснюватися консультація(рис. 3.12).

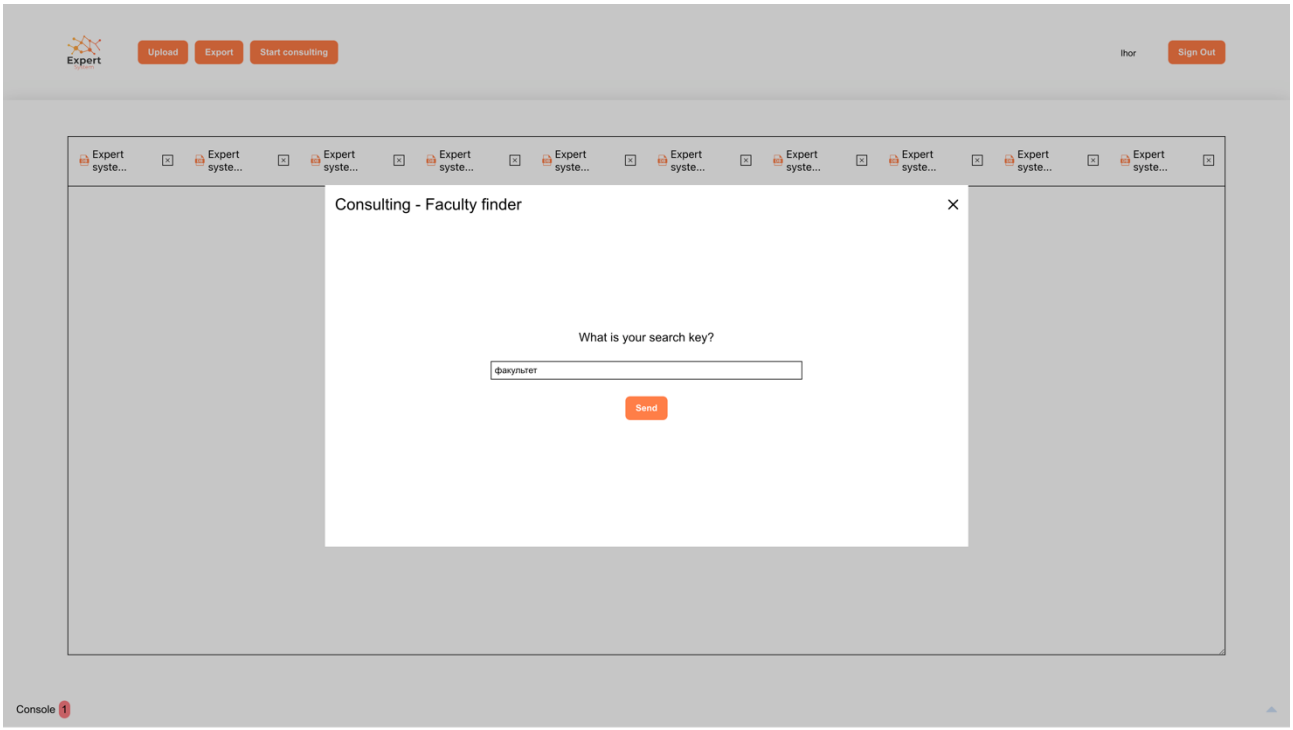


Рисунок 3.12 – Вікно для введення ключового слова для початку консультації

Кодове представлення вікна для введення ключового слова для початку консультації можна побачити на рисунку 3.13.

```
const ConsultingModule = ({parsedCode, setConsultingStatus}) => {
    //Search key
    const [searchKey, setSearchKey] = useState('')

    const searchKeyHandler = e => {
        const value = e.target.value
        //Getting search key value and setting it to state
        setSearchKey(value)
    }

    const setSearchKeyHandler = () => {
        //Setting search key value to global state, where we store answers and search key
        setData({
            searchKey: searchKey,
            localKeys: {}
        })
    }

    return data.searchKey === ''
        ? <>
            <ModalWrapperWindowBodyQuestion>
                {'What is your search key?'}
            </ModalWrapperWindowBodyQuestion>
            <ModalWrapperWindowBodyAnswerInput
                value={searchKey}
                onChange={searchKeyHandler}
            />
            <ModalWrapperWindowBodyAnswerButton
                onClick={setSearchKeyHandler}
            >
                {'Send'}
            </ModalWrapperWindowBodyAnswerButton>
        </>
        : (remoteData.length !== 0 && !answerStatus)
            ? setQuestion(remoteData[0][1]?.question, remoteData[0][1]?.allows, remoteData[0][0])
            : showResult(JSON.stringify(parsedCode.rules.filter(e => JSON.stringify(e.conditions) ===
JSON.stringify(data.localKeys))[0]?.action[data.searchKey]))
}

export default ConsultingModule
```

Рисунок 3.13 – Кодове представлення вікна для введення ключового слова для початку консультації

При проходженні консультації, будуть підбиратися питання на основі відповідей користувача(рис. 3.14). Також, можна побачити логіку запису відповіді на питання(рис. 3.15) та компонент підбору питання на основі відповіді(рис. 3.16).

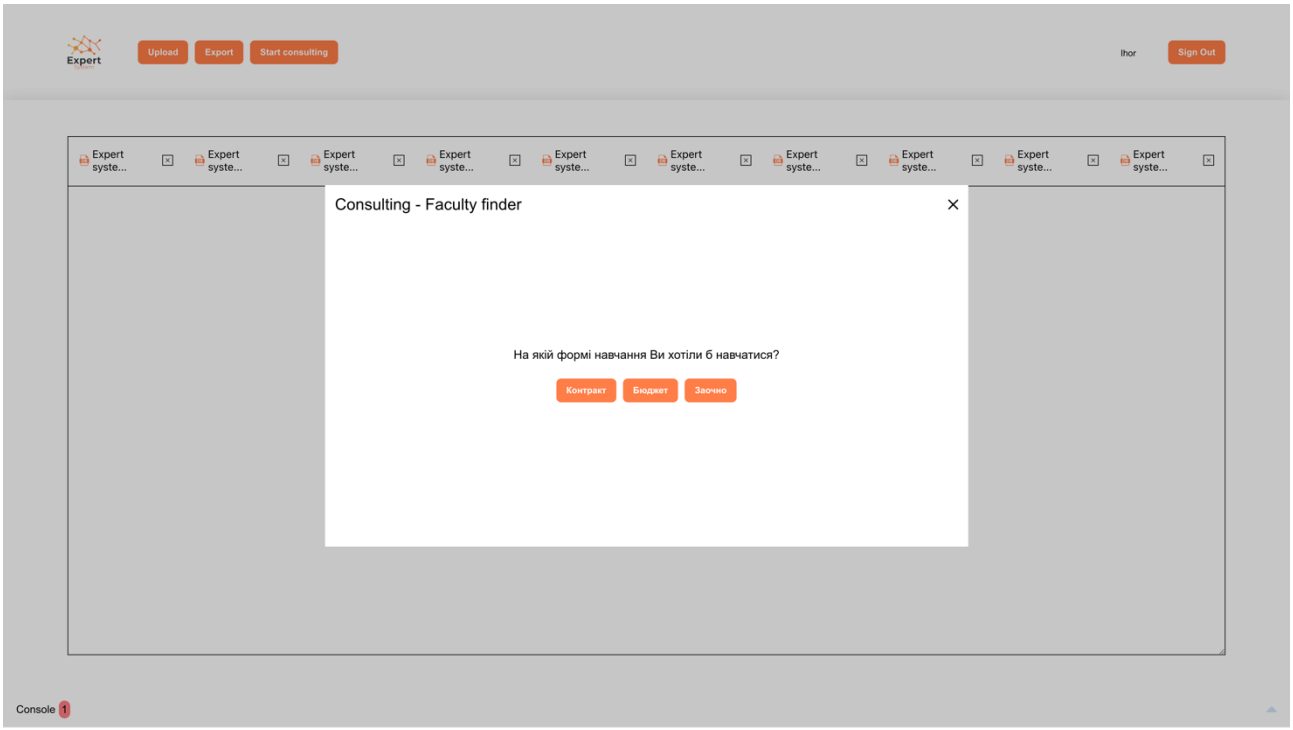


Рисунок 3.14 – Приклад підібраного питання під час консультації

```
const answerIndexHandler = (d) => {
    //Here we are writing our answers to global state and filtering
    //remote data by excluding question on which we get an answer
    setRemoteData(remoteData.filter(e => e[0] !== d.question))
    setData({
        searchKey: searchKey,
        localKeys: {
            ...data.localKeys,
            [d.question]: d.answer
        }
    })
}
```

Рисунок 3.15 – Логіка запису відповіді на питання

```
const setQuestion = (question, buttons, q) => {
    //Here we are setting new questions
    return (
        <>
            <ModalWrapperWindowBodyQuestion>
                {question}
            </ModalWrapperWindowBodyQuestion>
            <ModalWrapperWindowBodyAnswerButtons>
                {
                    buttons?.map(button => {
                        return (
                            <ModalWrapperWindowBodyAnswerButtonsItem
                                onClick={() => answerIndexHandler({
                                    question: q,
                                    answer: button
                                })}
                            >
                                {button}

</ModalWrapperWindowBodyAnswerButtonsItem>
                        })
                }
            </ModalWrapperWindowBodyAnswerButtons>
        </>
    )
}
```

Рисунок 3.16 – Компонент підбору питання на основі відповіді

При успішному проходженні консультації WEB-середовище покаже результат у вигляді «ключове слово – “результат”» (рис. 3.17). Також є можливість одразу пройти консультацію ще раз або ж вийти до головної сторінки закривши модальне вікно.

Кодове відображення компоненту для показу результату консультації та логіки початку консультації з початку можна побачити на рисунку 3.18.

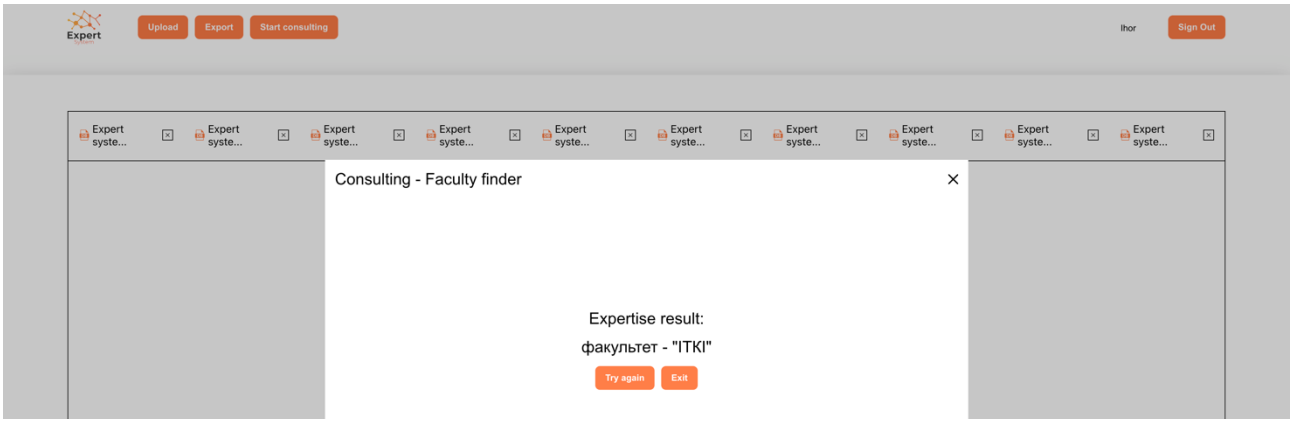


Рисунок 3.17 – Кінець консультації з відображеним результатом та можливістю почати консультацію заново або вийти

```
    //Global state
    const [data, setData] = useState({
        searchKey: '',
        localKeys: {}
    })

    const result = JSON.stringify(
      parsedCode.rules
      .filter(e => JSON.stringify(e.conditions) === JSON.stringify(data.localKeys))
[0]   ?.action[data.searchKey]
    )

    const tryAgainHandler = () => {
        //In this part, when pass the survey until the end we could press "Tru again"
        //and it will set our states to default values
        setSearchKey('')
        setData({
            searchKey: '',
            localKeys: {}
        })
        setRemoteData(Object.entries(rData?.data))
        setAnswerStatus(false)
    }

    const showResult = result => {
        //Here we are showing result of our survey
        return (
            <>
                <ModalWrapperWindowResultTitle>
                    {'Expertise result:'}
                </ModalWrapperWindowResultTitle>
                <ModalWrapperWindowResultTitle>
                    {result ? `${data.searchKey} - ${result}` : 'No data'}
                </ModalWrapperWindowResultTitle>
                <ModalWrapperWindowBodyAnswerButtons>
                    <ModalWrapperWindowBodyAnswerButtonsItem
                        onClick={tryAgainHandler}
                    >
                        {'Try again'}
                    </ModalWrapperWindowBodyAnswerButtonsItem>
                    <ModalWrapperWindowBodyAnswerButtonsItem
                        onClick={closeModalHandler}
                    >
                        {'Exit'}
                    </ModalWrapperWindowBodyAnswerButtonsItem>
                </ModalWrapperWindowBodyAnswerButtons>
            </>
        );
    };
```

Рисунок 3.18 – Кодове відображення компоненту для показу результату консультації та логіки початку консультації з початку

3.5 Тестування програмного продукту і аналіз результатів

Тестування продуктивності часто проводиться двома способами: на клієнтській та на серверній стороні. Потрібно перевірити, що сервер може впоратися з навантаженням, коли від користувачів одночасно надсилається кілька запитів. Якщо сервер не витримує, користувачі отримають помилку 503. Проте можна передбачити, як буде реагувати сервер, використовуючи інструменти тестування навантаження, такі як JMeter, K6 або Gatling.

За допомогою тестування продуктивності сервер може відправити відповідь протягом очікуваного часу, але це не обов'язково означає, що користувачі побачать відповідь миттєво. Те, як різні браузери обробляють дані корисного навантаження з серверів, також впливає на продуктивність. JavaScript робить WEB-сайти інтерактивними і повністю функціональними, але він також може додавати затримки, особливо якщо він не оптимізований і блокує повне відображення контенту. Те ж саме стосується і каскадних таблиць стилів.

Тестування клієнтської частини також має безліч інструментів, які допоможуть вам перевірити продуктивність вашого веб-сайту. Одним з найпопулярніших є Google Lighthouse [27].

Google Lighthouse з відкритим вихідним кодом дозволяє проводити аудит інших сфер, окрім продуктивності, включаючи доступність та пошукову оптимізацію. Lighthouse простий у використанні: для роботи з ним не потрібні додаткові технічні навички, щоб почати аудит ефективності WEB-сайту. Lighthouse можна запустити кількома способами, але найпростіший з них - як частину Інструментів для розробників Chrome, оскільки він вбудований в Chrome. Просто відкривши Інструменти розробника, переходимо на вкладку Lighthouse і починаємо аудит. На наведених нижче зображеннях(рис. 3.19 – 3.20) показано Google Lighthouse в дії.

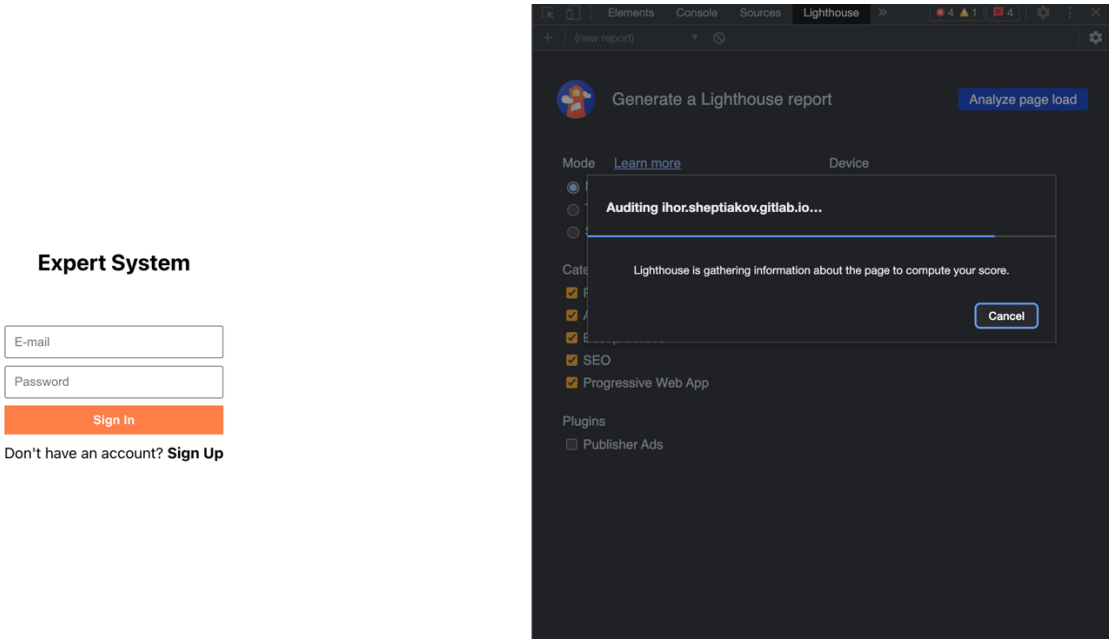


Рисунок 3.19 – Процес аудиту за допомогою Google Lighthouse

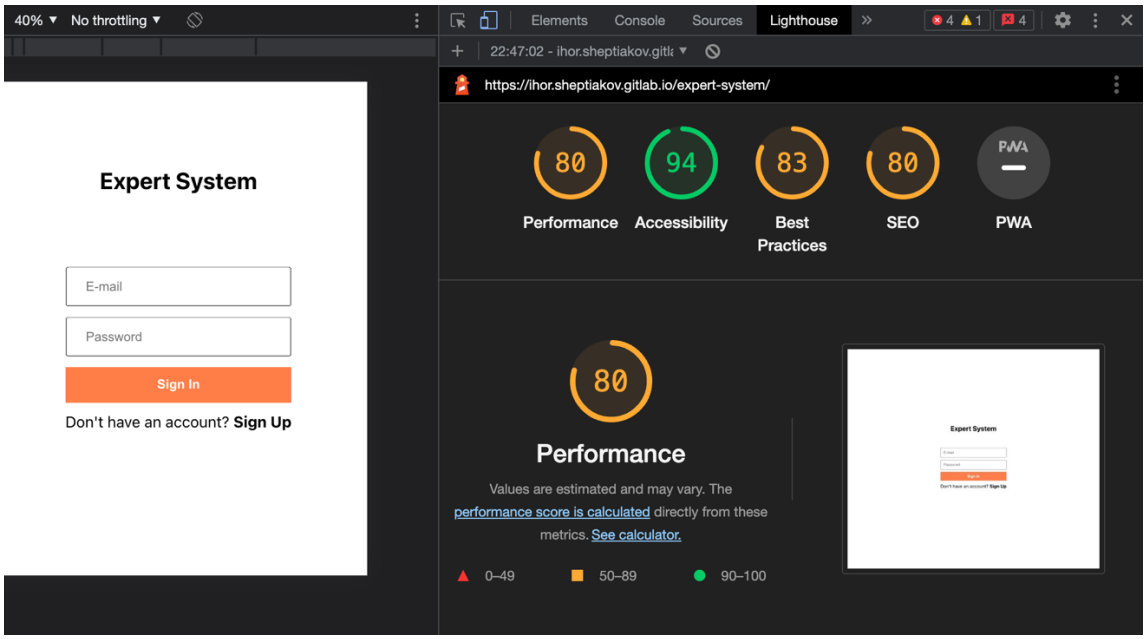


Рисунок 3.20 – Google Lighthouse оцінює ефективність, доступність, найкращі практики та SEO

Коли Lighthouse завершує аудит, він підраховує набір балів на основі таких показників, як "first contentful paint", “speed index”, "largest contentful paint", “time to interactive” та "total blocking time"[28 - 32].

Результати тестування за домогою Google Lighthouse можна звести до наступної таблиці(табл. 3.1):

1

Таблиця 3.1 – Результати тестування

	Результат	Очікування
First Contentful Paint	0.8 секунд	0 – 1.8 секунд
Speed Index	1.5 секунд	0 – 3.4 секунд
Largest Contentful Paint	2.5 секунд	0 – 2.5 секунд
Time to Interactive	3.1 секунд	0 – 3.8 секунд
Total Blocking Time	60 мілісекунд	0 – 200 мілісекунд

В якості тестування бази знань проведемо консультацію у віртуального експерта.

На даному прикладі представлено проходження консультації у експерта по рекомендаціям факультетів в університеті при вступі(рис. 3.21).

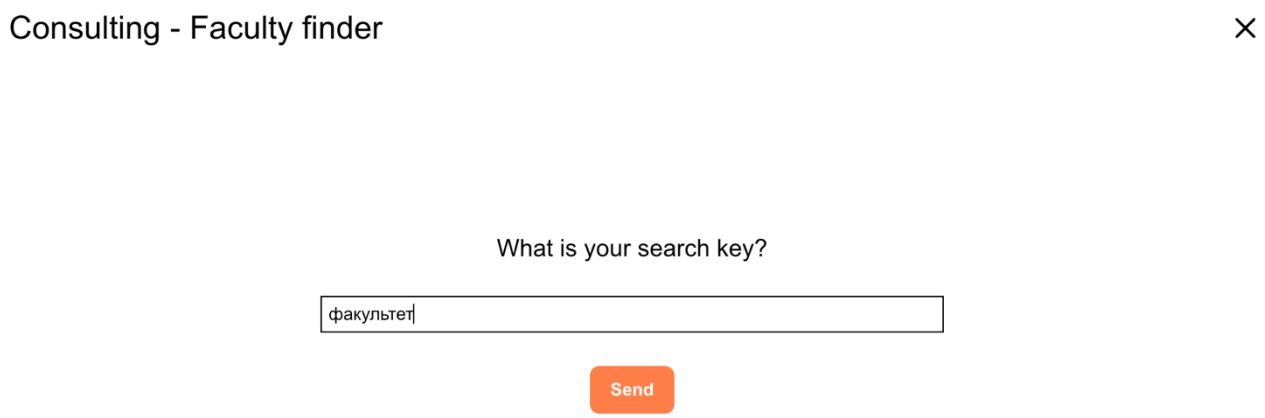


Рисунок 3.21 – Початок проходження консультації у віртуального експерта

Після введення ключового слова по якому буде проводитися консультація, користувачу в наступному вікні буде запропоновано розпочати консультацію по введеному ним ключовому слову(рис. 3.22).

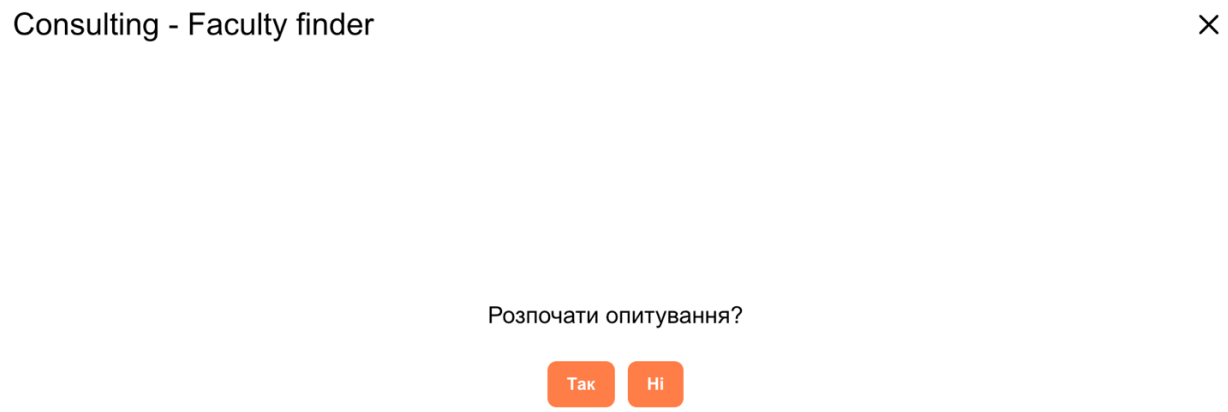


Рисунок 3.22 – Запит на проходження консультації по введеному ключовому слову

Після згоди на запит по проходженню консультації користувачу будуть даватися уточнюючі питання(рис. 3.23 – 3.27).

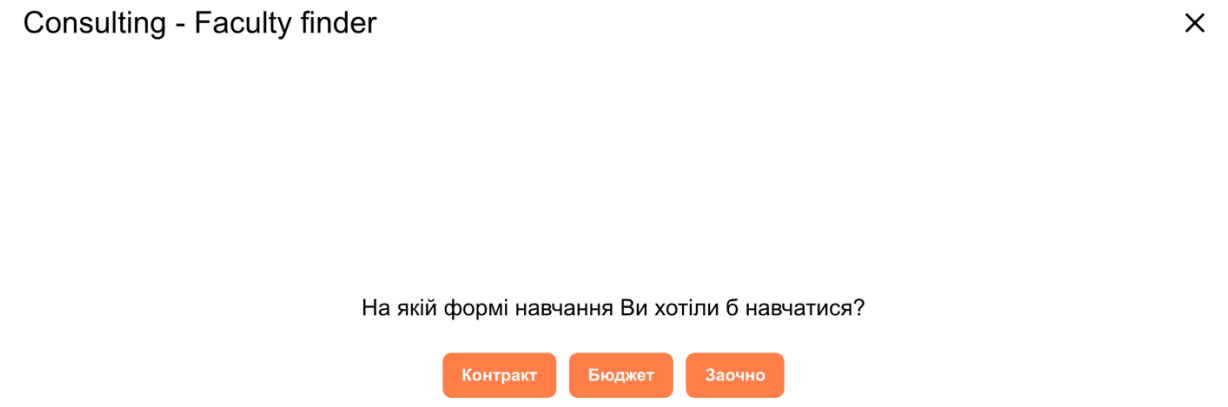


Рисунок 3.23 – Уточнююче питання

Користувач обрав «бюджет», і отримує наступне уточнююче питання.

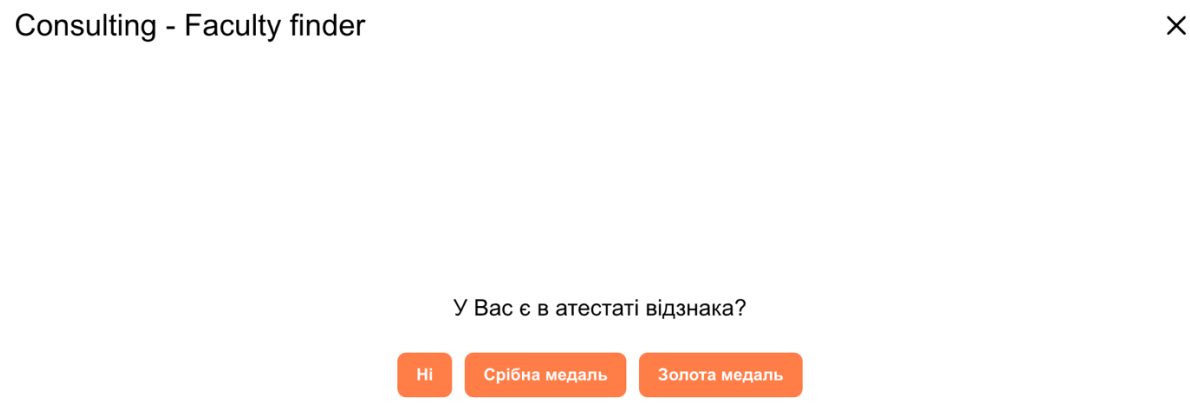


Рисунок 3.24 – Наступне уточнююче питання

Користувач обрав «золота медаль», і отримує наступне уточнююче питання.

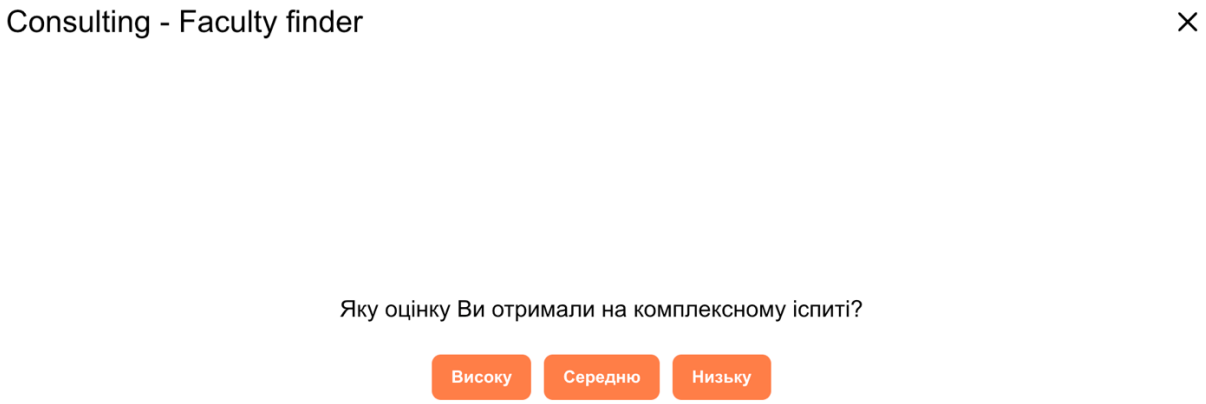


Рисунок 3.25 – Наступне уточнююче питання

Користувач обрав «висока», і отримує наступне уточнююче питання.

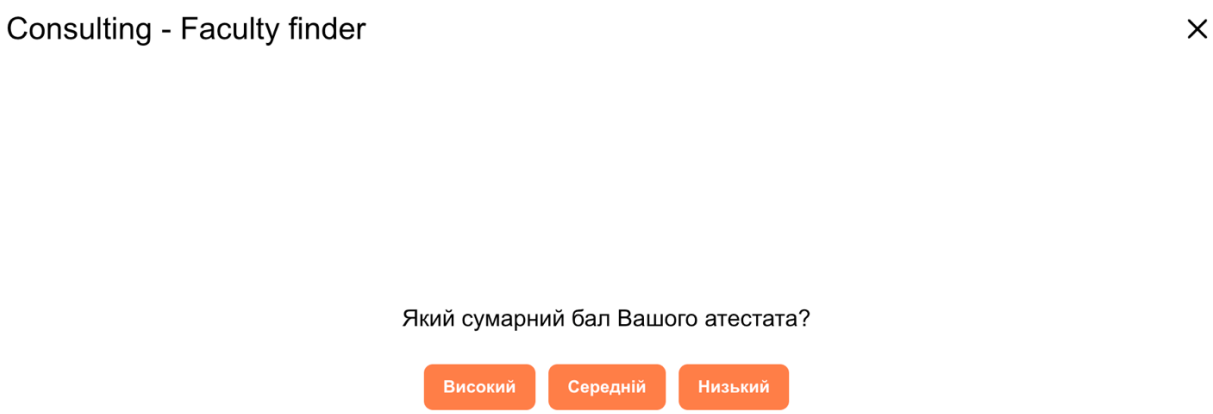


Рисунок 3.26 – Наступне уточнююче питання

Користувач обрав «високий», і отримує наступне уточнююче питання.

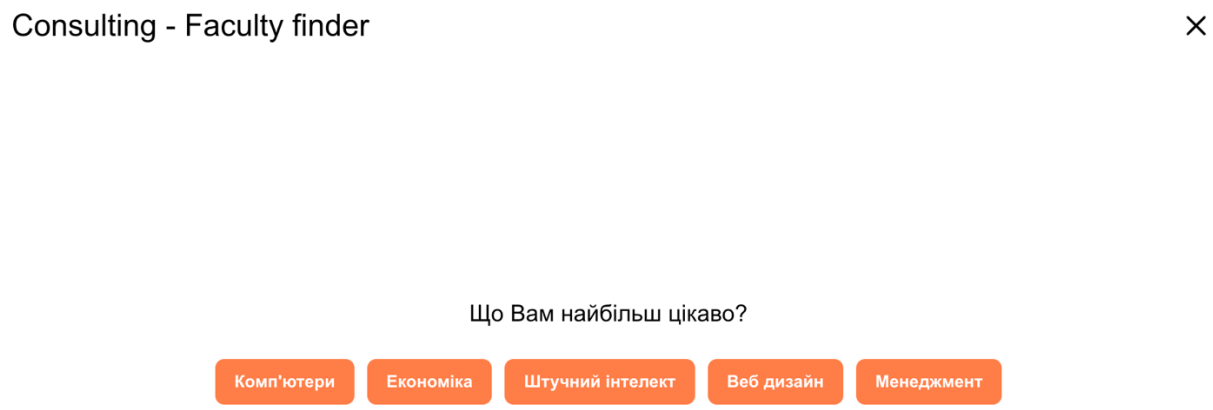


Рисунок 3.27 – Наступне уточнююче питання

Користувач обрав «штучний інтелект», і отримує від середовища відповідь зважаючи на попередні відповіді(рис. 3.28).

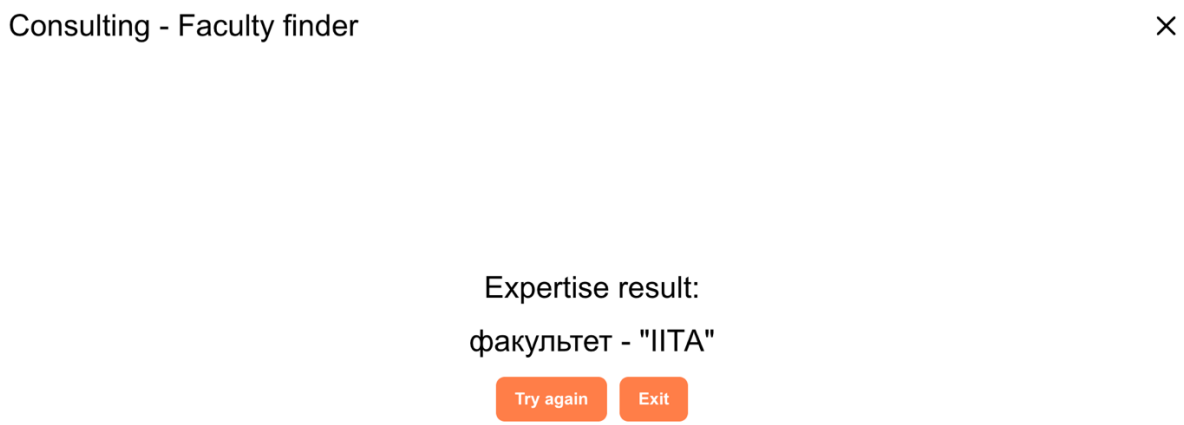


Рисунок 3.28 – Відповідь експерта на пройдену консультацію

На екрані отримання відповіді, користувач має змогу пройти консультацію ще раз, або ж вийти з вікна консультації.

В ході експериментальних досліджень реалізовано сто тестових запусків та зібрано статистичні дані, які показують, що у 97% випадків система показує точні результати роботи.

3.6 Висновок до розділу 3

В даному розділі розроблено програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем. Для цього було обґрунтовано вибір програмного середовища, розроблено структурно-логічну модель програмного засобу, розроблено UML діаграму. Тестування показало надійну роботу розробленої програми.

Виконавши програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем, було проаналізовано роботу програми на предмет неточностей, помилок та некорректної роботи. В ході аналізу роботи програми було проведено сто тестових запусків, які показали, що у 97% випадків система показує точні результати роботи. Також, WEB-

середовище було протестовано на швидкодію, де було отримано наступні результати: швидкість завантаження першого елементу сторінки 0.8 секунд, швидкість завантаження 1.5 секунд, швидкість завантаження найбільшого файлу сторінки 2.5 секунд, час до взаємодії за сторінкою 3.1сек та загальний час блокування 60 мілісекунд.

Вибір програмного середовища обумовив логічну структуру та напрямок розробки програмної реалізації. Структурно-логічні моделі обумовили головну логіку роботи програми, що являється основним в даній розробці.

Аналіз результатів роботи довів, що розроблена програма має перспективні можливості в плані навантаження та подальшого розширення чи модернізації. Таким чином, поставлена у роботі мета досягнута.

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 12-ма критеріями, наведеними в табл. 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

1	*2*	*3*	*4*	*5*	*6*
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 анало-гів	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 гіль ки повідомлен- ня 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 4.2.

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 $СБ_c$	$СБ_c = \frac{\sum_{i=1}^{3} СБ_i}{3} = 35.6$		

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 табл. 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

Встановивши досягнутий науково-технічний рівень розробки та її комерційний потенціал, одним-двома короткими реченнями обов’язково потрібно пояснити, за рахунок чого такий рівень було досягнуто:

– покращення та/або розширення функціональних можливостей нової науково-технічної розробки порівняно з аналогічними розробками, існуючими в цей час на ринку;

– значно вища якість, конкурентоспроможність нової науковотехнічної розробки за рахунок … (вказати за рахунок чого);

– суттєве зростання продуктивності, ефективності нової науковотехнічної розробки за встановленими показниками (вказати ці показники) та пояснити, за рахунок чого саме це було досягнуто;

– значне зменшення витрат ресурсів і часу на виконання поставлених завдань (вказати яких саме) та за рахунок чого саме це відбувається;

– суттєве покращення соціальних показників розвитку суспільства (збереження здоров’я населення, покращення якості життя тощо) та екологічних показників (зменшення забруднення навколишнього середовища, зменшення витрат енергії і сировини тощо).

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:

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

де k – кількість посад дослідників, залучених до процесу досліджень; $М_{ni}$ – 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 4.4.

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	1590,9	40	63 636
Науковий співробітник	25 000	1136,3	40	45 452
Всього				109 088

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.

$$З_д = (109\ 088 * 12\ \% / 100\ \%) = 13\ 090{,}56 \text{ (грн.)}$$

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.

$$Н_з = (109\ 088 + 13\ 090{,}56) \cdot 22\ \% / 100\ \% = 37\ 089{,}92 \text{ (грн.)}$$

4.2.3 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \times С_{прг.і} \times К_i, \quad (4.4)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.; $К_i$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($К_i$ = 1,10…1,12); k – кількість найменувань програмних засобів.

Отримані результати необхідно звести до таблиці(рис. 4.5).

Таблиця 4.5 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Visual Studio Code	1	Безкоштовно	Безкоштовно
Google Lighthouse	1	Безкоштовно	Безкоштовно
Всього			Безкоштовно

4.2.4 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:

$$А_{обл} = \frac{Ц_б}{Т_в} \times \frac{Т_{вик}}{12}, \quad (4.5)$$

$$А_{обл} = \frac{90\,000}{2} \times \frac{2}{12} = 7\,200,$$

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 4.6.

Таблиця 4.6 – 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	2	7 200
Приміщення	1 342 000	20	2	10 736
Офісне обладнання	45 000	4	2	1 800
Всього				19 736

4.2.5 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:

$$B_e = \sum \frac{{}_n W_{yi} \times t_i \times Ц_е \times К_{впі}}{\eta_{i=1}}, \quad (4.6)$$

$$B_e = \sum \frac{0{,}096 \times 320 \times 6{,}2 \times 0{,}9}{\eta_{i=1}} = 171{,}4,$$

$$B_e = \sum \frac{0{,}036 \times 320 \times 6{,}2 \times 0{,}9}{\eta_{i=1}} = 64{,}2,$$

$$B_e = \sum \frac{0{,}024 \times 80 \times 6{,}2 \times 0{,}9}{\eta_{i=1}} = 10{,}7,$$

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 4.7.

$$Ц_е = \left(Ц_{опт} + Ц_{розп} + Ц_{пост}\right)\left(1 + \frac{ПДВ}{100\%}\right), \quad (4.7)$$

$$Ц_е = (4{,}76606 + 0{,}19947 + 0{,}19947)\left(1 + \frac{20\%}{100\%}\right) = 6{,}2$$

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

вартість постачання електроенергії від енергорозподільної компанії до конкретного споживача (без ПДВ), грн за 1 кВт·год.

ПДВ – величина податку на додану вартість, %, у 2022 році ПДВ=20%.

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп`ютер (MacBook Pro 14-inch, 2021)	0.096	320	171,4
Монітор (DELL P2219H)	0.036	320	64,2
Освітлення	0.024	80	10,7
Всього			246,3

4.2.6 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:

$$\mathrm{I_в} = \left(\mathrm{З_о} + \mathrm{З_р}\right) \times \frac{\mathrm{Н_{ів}}}{100\%}, \quad (4.8)$$

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

$$I_в = 109\,088 * 55\% / 100\% = 59\,998{,}4 \text{ (грн.)}$$

4.2.7 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:

$$В_{нзв} = (З_о + З_р) \times \frac{Н_{нзв}}{100\%}, \quad (4.9)$$

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

$Н_{нзв}$ = 109 088 * 111% / 100% = 121 087,68 (грн.)

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

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.10)$$

$В_{заг}$ = 109 088 + 13 090,56 + 37 089,92 + 19 736 + 246,3 + 59 998,4 + 121 087,68 = 360 336,86 грн.

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

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

де η – коефіцієнт, який характеризує етап (стадію) виконання науководослідної 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 = 360 336,86 / 0,5 = 720 673,72 грн.

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

В ринкових умовах узагальнювальним 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 науковотехнічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науковотехнічної 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:

$$\Delta П_i = (\pm \Delta Ц_o \times N + Ц_o \times \Delta N)_i \times \lambda \times \rho \times \left(1 - \frac{\vartheta}{100}\right), \qquad (4.12)$$

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:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.13)$$

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:

$$PV = k_{інв} \times ЗВ, \quad (4.14)$$

де $k_{інв}$ – 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 $k_{інв}$ = 2...5, але може бути і більшим;

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

$$PV = 2 * 720\,673{,}72 = 1\,441\,347{,}44 \text{ грн.}$$

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

$$Е_{абс} = ПП – PV, \quad (4.15)$$

$$Е_{абс} = 7\,604\,256{,}29 – 1\,441\,347{,}44 = 6\,162\,908{,}85 \text{ грн.}$$

Оскільки $Е_{абс} > 0$ 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 $E_в$. Для цього використаємо формулу:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (4.16)$$

$Т_ж$ життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 6\,162\,908{,}85 \div 1\,441\,347{,}44)} - 1 = 0{,}74$$

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

$$\tau = d + f, \qquad (4.17)$$

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).

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

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:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.18)$$

$$T_{ок} = 1 / 0{,}74 = 1{,}35 \text{ р.}$$

Оскільки < 3-х років, а саме термін окупності рівний 1,35 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 720 673,72 гривень. Було спрогнозовано 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

Під час виконання магістерської кваліфікаційної роботи розроблено WEB-орієнтована інформаційна технологія для розробки експертних систем: клієнтська частина.

В першому розділі проаналізовано предметну область у сфері розробки експертних систем, визначено принцип її роботи. Проведено поверхневий порівняльний аналіз програмних засобів створення WEB-інтерфейсів, таких як Python, Ruby, JavaScript та PHP. Показано переваги та недоліки мов програмування. Визначена постановка задачі, необхідні критерії та вимоги.

В другому розділі було наведено класифікацію експертних систем: за типом задачі, за зв’язком з реальним часом та за ступенем інтеграції з іншими програмами. Визначено критерії розробки, такі як доцільність, визначеність та можливість. Проаналізовано моделі подання знань, що дозволило визначити модель подання знань, здійснити порівняльний аналіз та визначити слабкі та сильні сторони кожної моделі. Як результат, доцільною моделлю була обрана продукційна, яка є універсальною щодо методів програмування, легкою щодо внесення змін та високо модульною. Розроблено структурну схему.

В третьому розділі було розроблено програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем. Для цього було обґрунтовано вибір програмного середовища, розроблено структурно-логічні моделі модулів програмного засобу. Тестування показало надійну роботу розробленої програми.

Виконавши програмну реалізацію WEB-орієнтованої інформаційної технології для розробки експертних систем, було проаналізовано роботу програми на предмет неточностей, помилок та некорректної роботи.

Вибір програмного середовища обумовив логічну структуру та напрямок розробки програмної реалізації. Структурно-логічні моделі обумовили головну логіку роботи програми, що являється основним в даній розробці.

Аналіз результатів роботи показав наступні результати тестування WEB-середовища: швидкість завантаження першого елементу сторінки 0.8 секунд, швидкість завантаження 1.5 секунд, швидкість завантаження найбільшого файлу сторінки 2.5 секунд, час до взаємодії за сторінкою 3.1сек та загальний час блокування 60 мілісекунд.

Економічна частина даної роботи містить розрахунок витрат на розробку WEB-орієнтованої інформаційної технології для розробки експертних систем, сума яких складає 720 673,72 гривень. Було спрогнозовано 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.

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. Експертна система [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%95%D0%BA%D1%81%D0%BF%D0%B5%D1%80%D1%82%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

4. Месюра В.І., Яровий А.А., Арсенюк І.Р. Експертні системи. Частина 1. Навчальний посібник. – Вінниця: ВНТУ, 2006. – 24 с.

5. Організація баз даних та знань. Загальна характеристика баз знань [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151030211833/44618/index.html

6. Microsoft [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/uk-ua/

7. Mac Os [Електронний ресурс]. – Режим доступу: https://www.apple.com/macos/ventura/

8. Linux [Електронний ресурс]. – Режим доступу: https://www.linux.org/

9. Python [Електронний ресурс]. – Режим доступу: https://www.python.org/

10. PHP [Електронний ресурс]. – Режим доступу: https://www.php.net/

11. Ruby [Електронний ресурс]. – Режим доступу: https://www.ruby-lang.org/en/

12. JavaScript [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

13. React.js [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/

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

15. Джарратано Дж., Райли Г. Экспертные системы: принципы разработки и программирование. – М.: ИД «Вильямс», 2007. – 1152 с.

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

17. Visual Studio Code[Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/

18. WebStorm [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/webstorm/

19. Atom [Електронний ресурс]. – Режим доступу: https://atom.io/

20. Інтегроване середовище розробки для JavaScript. WebStorm [Електронний ресурс]. – Режим доступу: https://www.wiki.uk-ua.nina.az/Webstorm.html

21. Java [Електронний ресурс]. – Режим доступу: https://www.java.com/

22. C(мова програмування) [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/C_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)

23. HTML [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML

24. CSS [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS

25. .NET | Free. Cross-platform. Open source [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com/en-us/

26. Принципи побудови формально-логічних систем [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/1056041240328/logika/printsipi_pobudovi_formalno-logichnih_sistem

27. Lighthouse [Електронний ресурс]. – Режим доступу: https://developer.chrome.com/docs/lighthouse/overview

28. First Contentful Paint [Електронний ресурс]. – Режим доступу: https://web.dev/first-contentful-paint/?utm_source=lighthouse&utm_medium=devtools

29. Speed Index [Електронний ресурс]. – Режим доступу: https://web.dev/speed-index/?utm_source=lighthouse&utm_medium=devtools

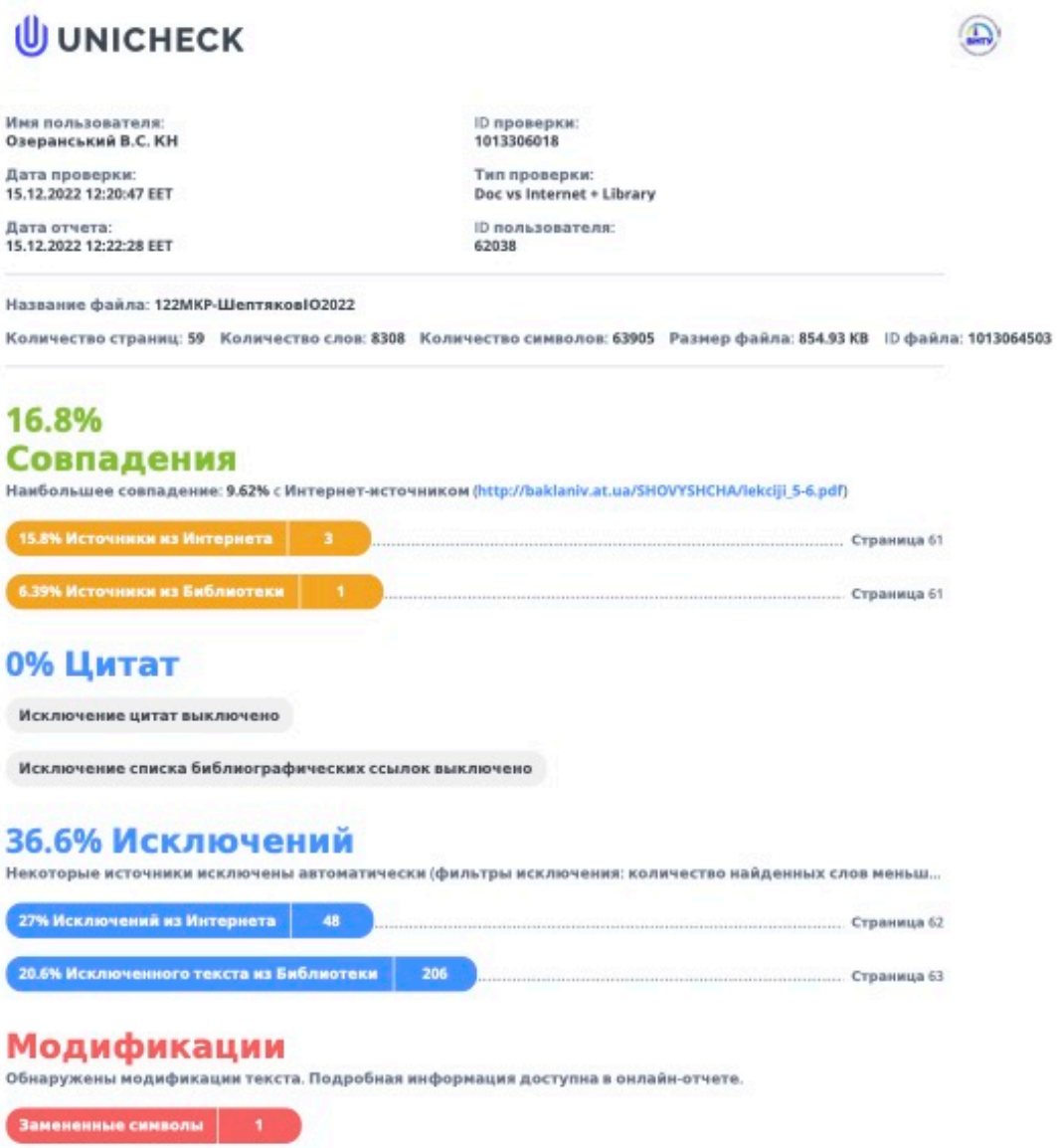
30. Largest Contentful Paint [Електронний ресурс]. – Режим доступу: https://web.dev/lighthouse-largest-contentful-paint/?utm_source=lighthouse&utm_medium=devtools

31. Time to Interactive [Електронний ресурс]. – Режим доступу: https://web.dev/interactive/?utm_source=lighthouse&utm_medium=devtools

32. Total Blocking Time [Електронний ресурс]. – Режим доступу: https://web.dev/lighthouse-total-blocking-time/?utm_source=lighthouse&utm_medium=devtools

Додаток А

Протокол перевірки МКР на наявність текстових запозичень



UNICHECK

Имя пользователя:
Озеранський В.С. КН

ID проверки:
1013306018

Дата проверки:
15.12.2022 12:20:47 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
15.12.2022 12:22:28 EET

ID пользователя:
62038

Название файла: 122МКР-ШептяковІО2022

Количество страниц: 59 Количество слов: 8308 Количество символов: 63905 Размер файла: 854.93 KB ID файла: 1013064503

16.8%
Совпадения

Наибольшее совпадение: 9.62% с Интернет-источником (http://baklaniv.at.ua/SHOVYSHCHA/lekciji_5-6.pdf)

15.8% Источники из Интернета 3 ..... Страница 61

6.39% Источники из Библиотеки 1 ..... Страница 61

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

36.6% Исключений

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

27% Исключений из Интернета 48 ..... Страница 62

20.6% Исключенного текста из Библиотеки 206 ..... Страница 63

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы 1

Додаток Б
Лістинг програми

```
function App() {
    const uploadButtonRef = useRef(null)
    const [authData, setAuthData] = useState(authDefaultData)
    const [renameFileData, setRenameFileData] = useState(null)
    const [expertSystemsList, setExpertSystemsList] = useState(defaultList)
    const [splitter, setSplitter] = useState(splitterDefaultData)
    const [code, setCode] = useState('')
    const [parsedCode, setParsedCode] = useState(remoteData)
    const [consultingStatus, setConsultingStatus] = useState(false)
    const [consoleStatus, setConsoleStatus] = useState(false)
    const [choosenFile, chooseFile] = useState('')
    const [selectedFile, selectFile] = useState('')
    const [mimeType, setMimeType] = useState('')
    const [fileName, setFileName] = useState('')
    const [file, setFile] = useState('')
    const [logCounter, setLogCounter] = useState(0)

    const consoleStatusHandler = () => {
        setLogCounter(0)
        setConsoleStatus(!consoleStatus)
        if (!consoleStatus) {
            setLogCounter(0)
        }
    }

    useEffect(() => {
        setLogCounter(prevState => prevState + 1)
    }, [])

    const eslParser = (str, opts) => {
        const options = opts || {};
        const {logs} = options;

        const kbObject = {
            data: {},
            rules: []
        };

        const dataObj = str
            .split(';')
            .map(row => row.replace(/(\r\n|\n|\r)/gm, '').trim())
            .filter(str => str)
            .map(row => {
                if (row.substr(0, 5) === 'allow') {
                    const rowData = row.split('=');
                    const parsedKey = rowData[0]
                        .replace('allow', '')
                        .replace('(', '')
```

```
                .replace(')', '')
                .replace(/\"/gm, '')
                .trim();
            const parsedValues = rowData[1].split(',').map(v => v.trim());
            kbObject.data[parsedKey] = {allows: parsedValues};
        } else if (row.substr(0, 8) === 'question') {
            const rowData = row.split('=');
            const parsedKey = rowData[0]
                .replace('question', '')
                .replace('(', '')
                .replace(')', '')
                .replace(/\"/gm, '')
                .trim();
            const parsedValues = rowData[1].replace(/\"/gm, '').trim();

            kbObject.data[parsedKey] = {
                questions: parsedValues,
                allows: kbObject.data[parsedKey].allows,
            };
        } else if (row.substr(0, 4) === 'rule') {
            const rowData = row.split(':')[1].split('then');
            const ruleObj = {
                conditions: {},
                action: {}
            };


            rowData[0].split('&').forEach(cond => {
                const condData = cond.split('=');
                ruleObj.conditions[condData[0].trim()] = condData[1].trim();
            });
            rowData[1].split('&').forEach(cond => {
                const condData = cond.split('=');
                ruleObj.action[condData[0].trim()] = condData[1].trim();
            });

            kbObject.rules.push(ruleObj);
        } else {
            throw new Error('Line must start on allow | question | rule');
        }
        return row;
    });

let i = 1;

if (logs) {
    dataObj.forEach(row => {
        console.log(`${i++} | ` + JSON.stringify(row));
    });
}

return kbObject;
```

```
};

function readTextFile(file) {
    let rawFile = new XMLHttpRequest();
    rawFile.open("GET", file, false);
    rawFile.onreadystatechange = function () {
        if (rawFile.readyState === 4) {
            if (rawFile.status === 200 || rawFile.status == 0) {
                let allText = rawFile.responseText;
                const cText = eslParser(allText)
                setParsedCode(cText)
                setCode(allText);
            }
        }
    }
    rawFile.send(null);
}

const uploadFileHandler = event => {
    let file = event.target.files[0];
    const reader = new FileReader();
    let url = file && reader.readAsDataURL(file);
    const id = Math.random(), addedFiles = []
    if (event.target.files.length !== 0) {
        reader.onprogress = function (event) {
            const loaded = event.loaded, total = event.total
            const data = {
                progress: Math.floor((loaded / total) * 100),
                id: id
            }
            if (total < 10000000) {
                if (addedFiles.filter(e => e.id === id).length === 0) {
                    addedFiles.push(
                        ...addedFiles,
                        data
                    )
                }
            }
        };
        reader.onloadend = function (e) {
            const total = e.total
            if (total < 10000000) {
                chooseFile([reader.result])
                selectFile(file)
                readTextFile([reader.result])
                setMimeType(file.type)
                setFileName(file.name)
                setFile(reader.result.slice(reader.result.indexOf(',') + 1))
            }
        }.bind(this);
    }
}
```

```
    const controlUploadButtonHandler = () => uploadButtonRef?.current?.click()
    const startConsultationHandler = () => setConsultingStatus(!consultingStatus)
    const exportDataHandler = () => {
        const element = document.createElement("a");
        const file = new Blob([code], {type: 'text/plain'});
        element.href = URL.createObjectURL(file);
        console.log(code, file)
        element.download = "Expert system.txt";
        document.body.appendChild(element); // Required for this to work in FireFox
        element.click();
    }
    const setLog = (type, key, value) => {
        const allowedTypes = {
            'warning': { backgroundColor: color.keyWord2, textColor: '#000' },
            'error': { backgroundColor: color.keyWord3, textColor: '#000' },
        }
        if (!Object.keys(allowedTypes).includes(type)) {
            throw new Error('Invalid log type')
        }
        return (
            <AppWrapperTerminalBody
                textColor={allowedTypes[type].textColor}
                backgroundColor={allowedTypes[type].backgroundColor}
            >
                {key}: {value}
            </AppWrapperTerminalBody>
        )
    }
    return (
        <AuthContext.Provider value={[authData, setAuthData]}>
            <RenameFileContext.Provider value={[renameFileData, setRenameFileData]}>
                <ExpertSystemsListContext.Provider value={[expertSystemsList,
setExpertSystemsList]}>
                    <SplitterContext.Provider value={[splitter, setSplitter]}>
                        <AppWrapper>
                            {
                                !authData?.authorised
                                    ? <Auth/>
                                    : <>
                                        {
                                            consultingStatus &&
                                            <Modal
                                                title={'Consulting - Faculty finder'}
                                                body={
                                                    <ConsultingModule
                                                        parsedCode={parsedCode}
                                                         setConsultingStatus={
                                                           setConsultingStatus
                                                         }
                                                    />
                                                }
```

```
                                        setModalStatus={() =>
                                            setConsultingStatus(false)
                                        }
                                    />
                                }
                                {
                                    renameFileData?.status &&
                                    <Modal
                                        title={'Rename file'}
                                        body={<RenameFileModule/>}
                                        setModalStatus={() =>
                                        setRenameFileData(prevState => [{
                                            ...prevState,
                                            status: false
                                        }])}
                                    />
                                }
                                <Header
                                    uploadButtonRef={uploadButtonRef}
                                    exportDataHandler={exportDataHandler}
                                    uploadFileHandler={uploadFileHandler}
                                      startConsultationHandler={
                                            startConsultationHandler
                                      }
                                      controlUploadButtonHandler={
                                            controlUploadButtonHandler
                                      }
                                />
                                <Main
                                    code={code}
                                    setCode={setCode}
                                    consoleStatus={consoleStatus}
                                />
                                <Terminal
                                    consoleStatus={consoleStatus}
                                    consoleStatusHandler={consoleStatusHandler}
                                    logCounter={logCounter}
                                    setLog={setLog}
                                />
                            </>
                        }
                    </AppWrapper>
                </SplitterContext.Provider>
            </ExpertSystemsListContext.Provider>
        </RenameFileContext.Provider>
    </AuthContext.Provider>
  );
}
```

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

WEB-ОРІЄНТОВАНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ. ЧАСТИНА 2

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

Керівник: д.т.н., професор каф. КН
Яровий А.А.
(прізвище та ініціали)
«_____»_____2022 р.

Вінниця ВНТУ - 2022 рік

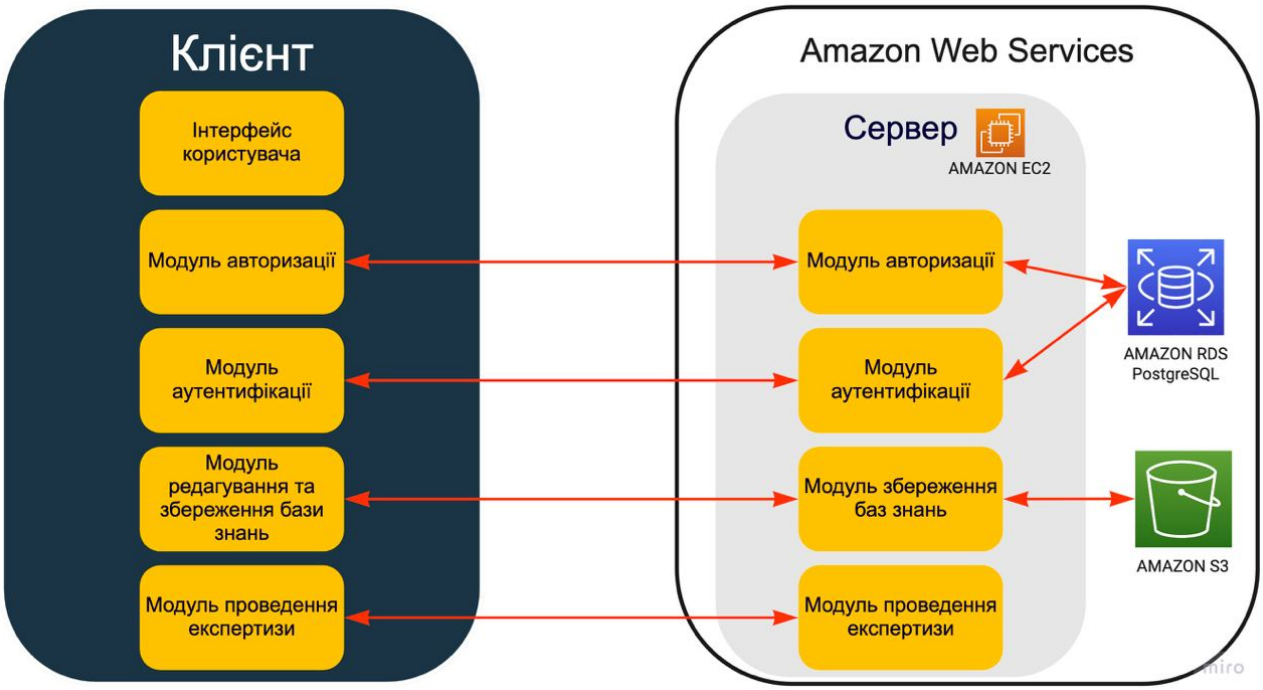


Рисунок В.1 – Загальна схема роботи WEB-середовища для розробки експертних систем

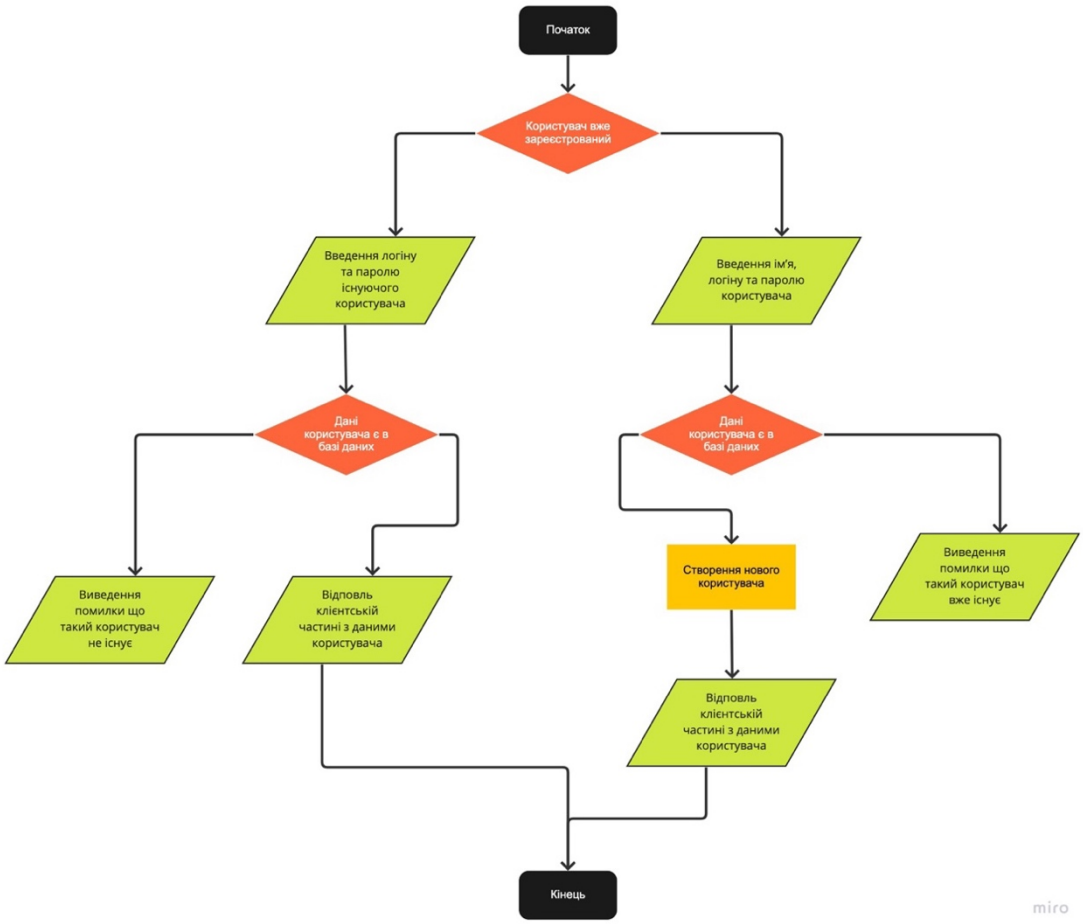


Рисунок В.2 – Схема алгоритму процесу реєстрації / авторизації користувача

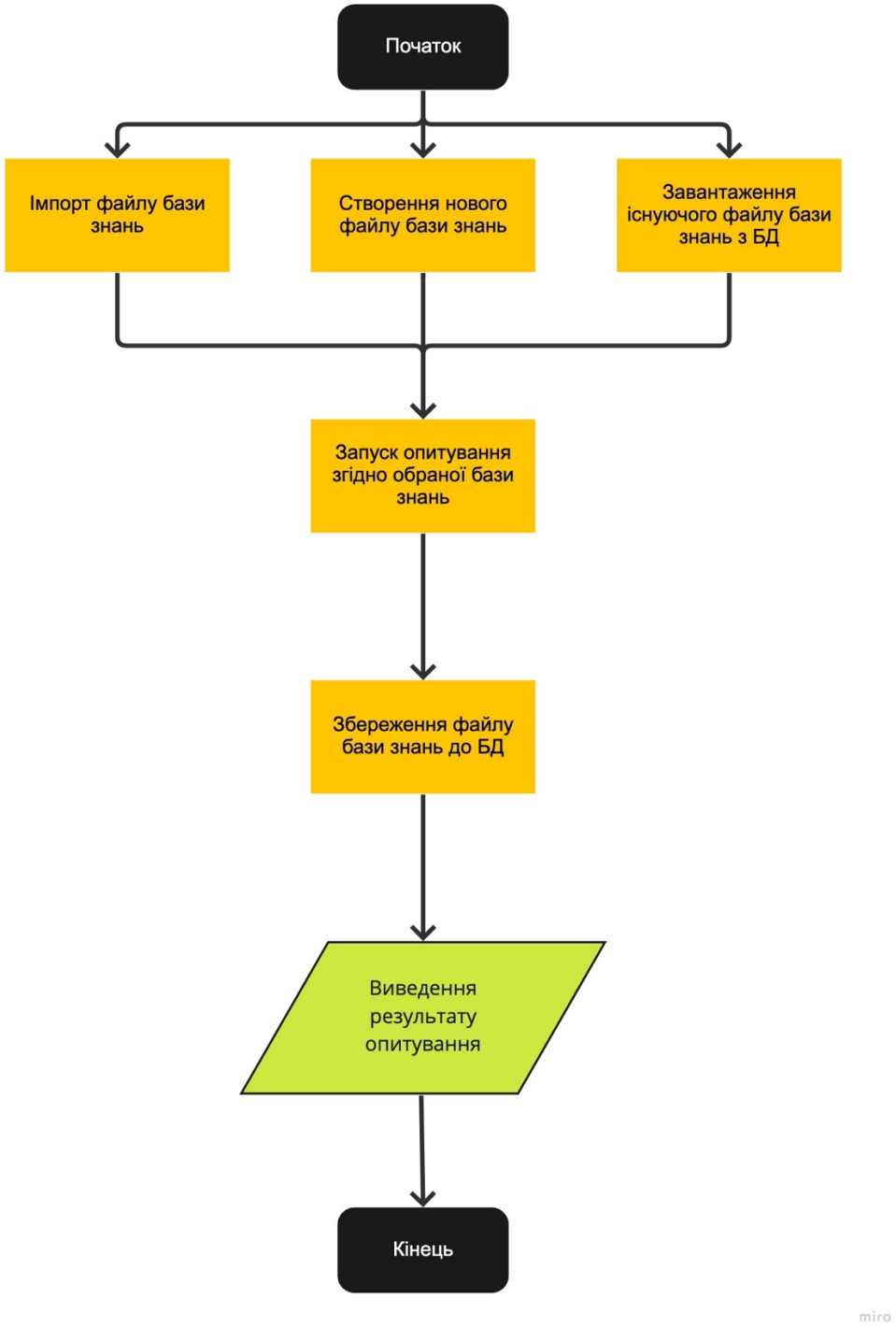


Рисунок В.3 – Схема алгоритму процесу створення / завантаження бази знань

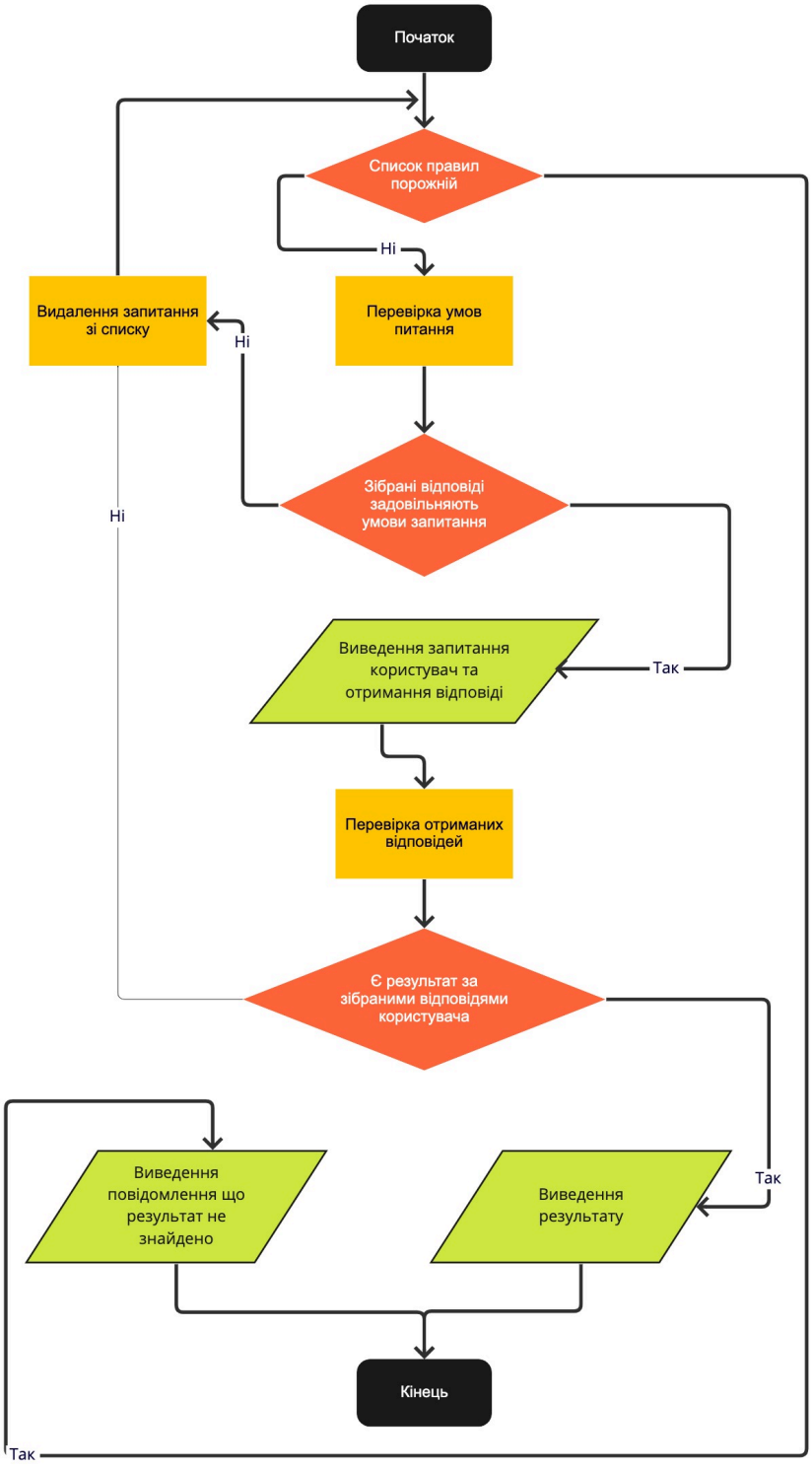


Рисунок В.4 – Схема алгоритму процесу опитування користувача

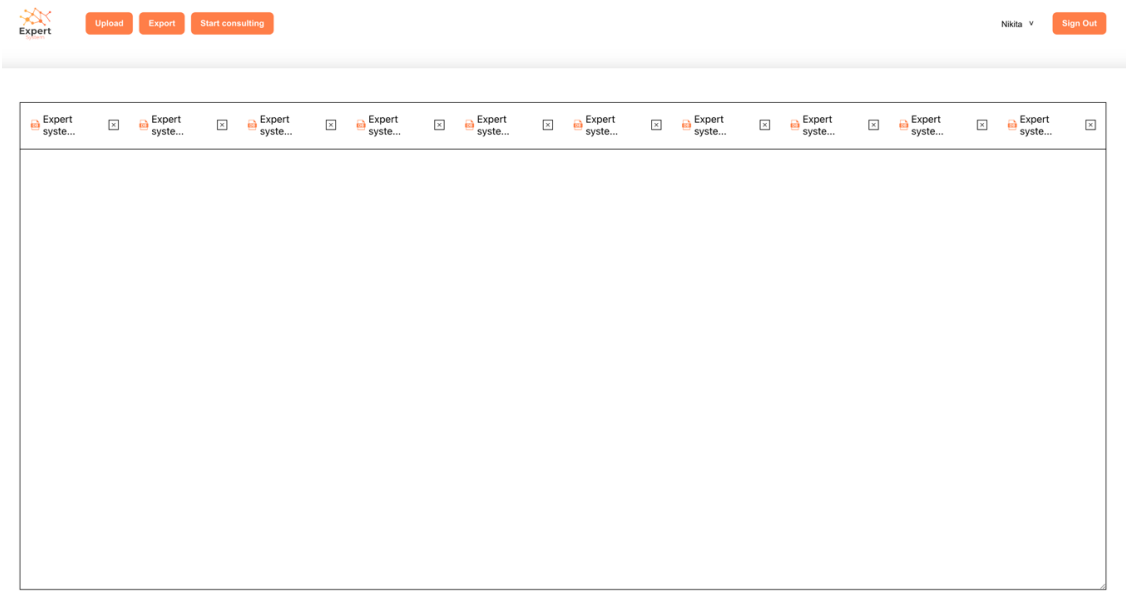


Рисунок В.5 – Головна сторінка

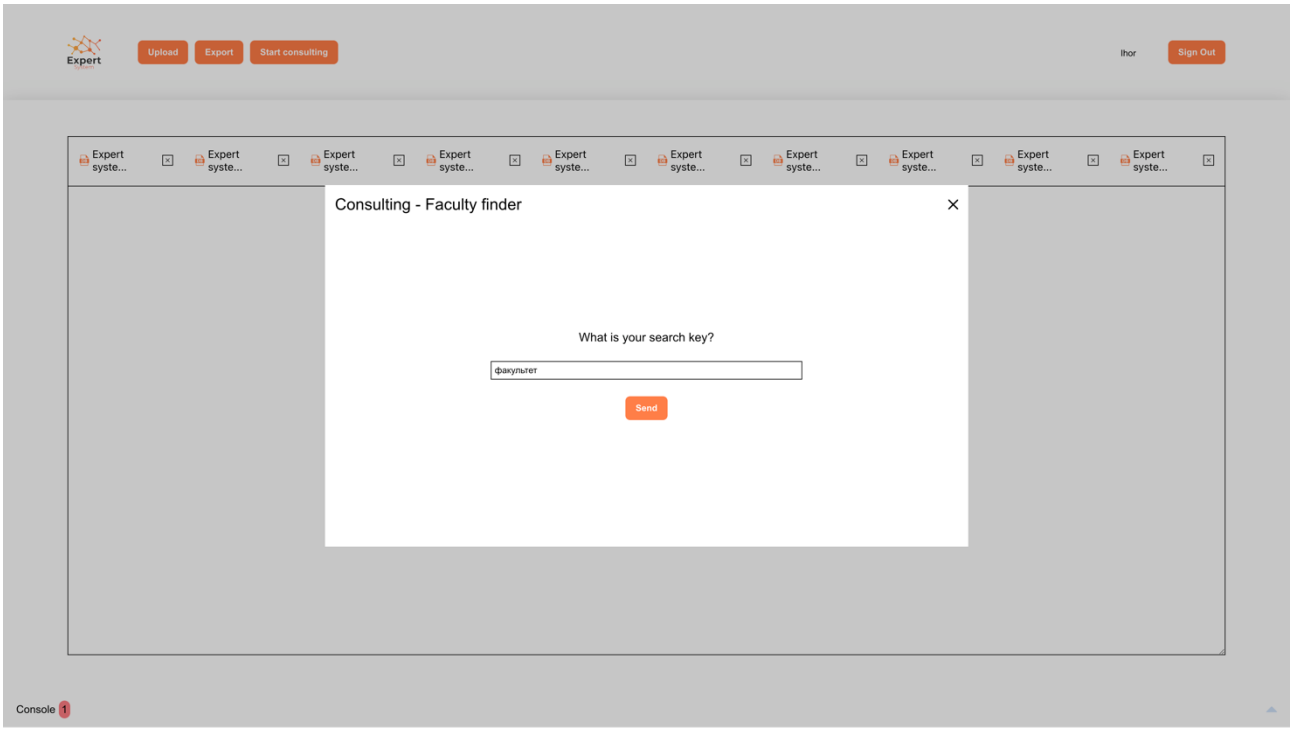


Рисунок В.6 – Початок проходження консультації у віртуального експерта

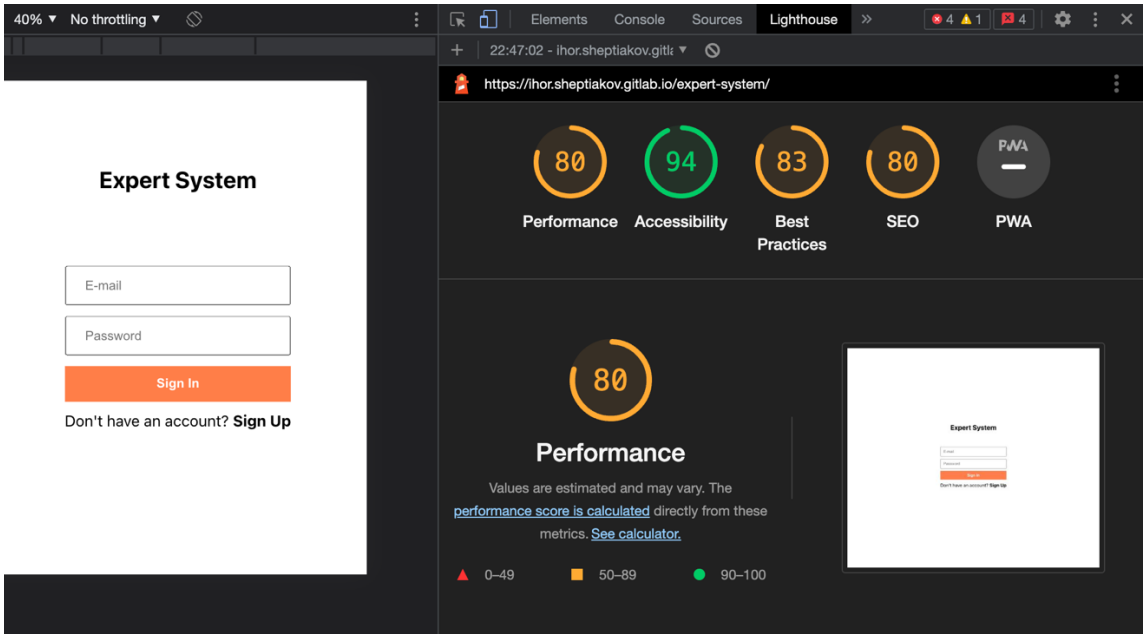


Рисунок В.7 – Результати тестування

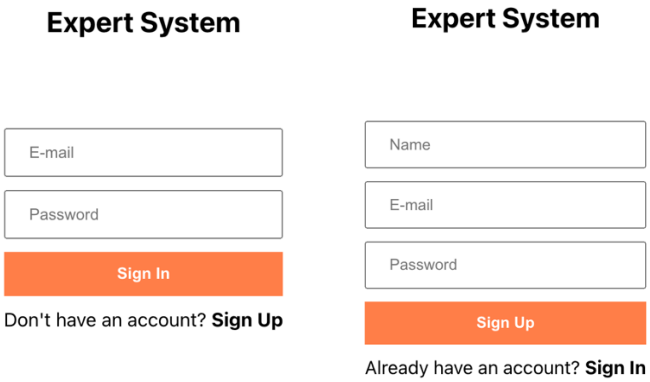
Таблиця В.1 – Результати тестування

	Результат	Очікування
First Contentful Paint	0.8 секунд	0 – 1.8 секунд
Speed Index	1.5 секунд	0 – 3.4 секунд
Largest Contentful Paint	2.5 секунд	0 – 2.5 секунд
Time to Interactive	3.1 секунд	0 – 3.8 секунд
Total Blocking Time	60 мілісекунд	0 – 200 мілісекунд

Додаток Г

Інструкція користувача

Для початку роботи з WEB-середовищем потрібно перейти за посиланням https://ihor.sheptiakov.gitlab.io/expert-system/. При переході за цим посиланням користувач потрапляє на екран автентифікації(рис. Г.1а) / авторизації(рис. Г.1б).



а) б)

Рисунок Г.1 - Екрани:

а – автентифікації; б – авторизації

Після проходження автентифікації, користувач потрапляє на головний екран(рис. Г.2). На головному екрані знаходяться всі функції для взаємодії з WEB-середовищем, такі як: завантаження бази знань до/з WEB-середовища, створення/редагування бази знань за допомогою вбудованого редактора, перегляд усіх завантажених до WEB-середовища баз знань та кнопка для виходу з системи.

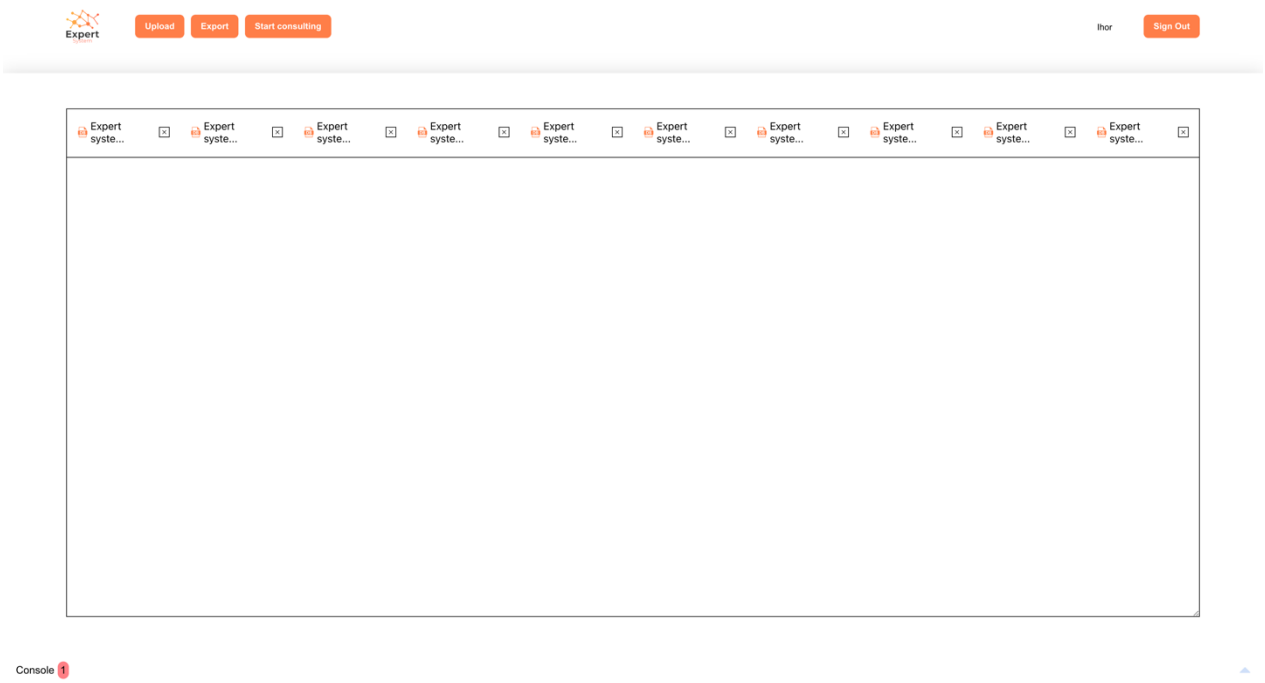


Рисунок Г.2 – Головний екран

Для створення бази знань у WEB-середовищі, потрібно знати певну структуру написання бази знань. Наприклад:

```
{
    data: {
        condition_key: {
            question: 'Start survey?',
            allows: [ 'Yes', 'No' ]
        },
    },
    rules: [
        {
            conditions: { condition_key: 'No' },
            action: { result_key: 'Result' }
        },
    }
}
```

У даній структурі в “data” прописуються всі умовні питання, де з кожного питання при відповіді користувача(рис. Г.3) береться відповідний елемент з масиву “allows”.

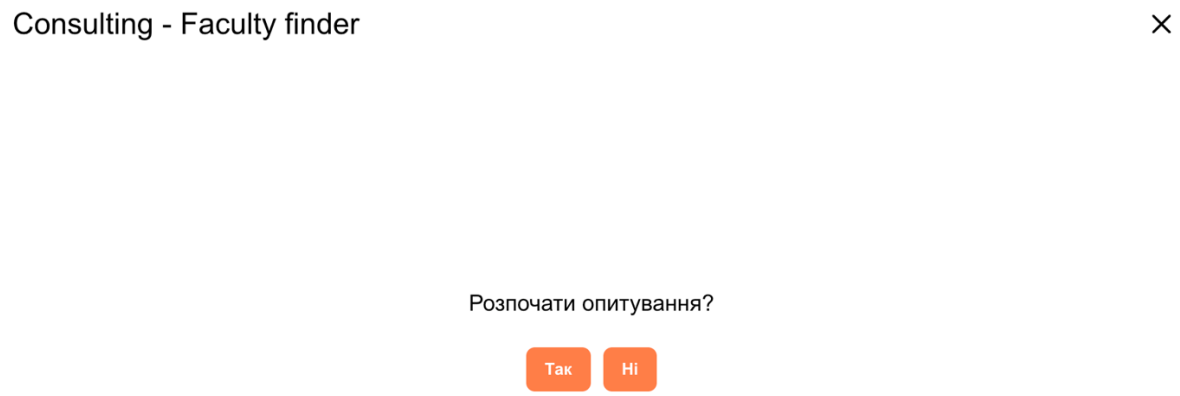


Рисунок Г.3 – Приклад питання при проходженні консультації

Потім усі відповіді збираються по принципу “condition_key: варіант відповіді користувача” та оброблюються у блоці “rules”. Якщо у блоці “rules” усі відповіді користувача співпадають з правилами у блоці “conditions”, то користувач отримає відповідний результат(рис. Г.4) з блоку “action”, а саме “result_key”.

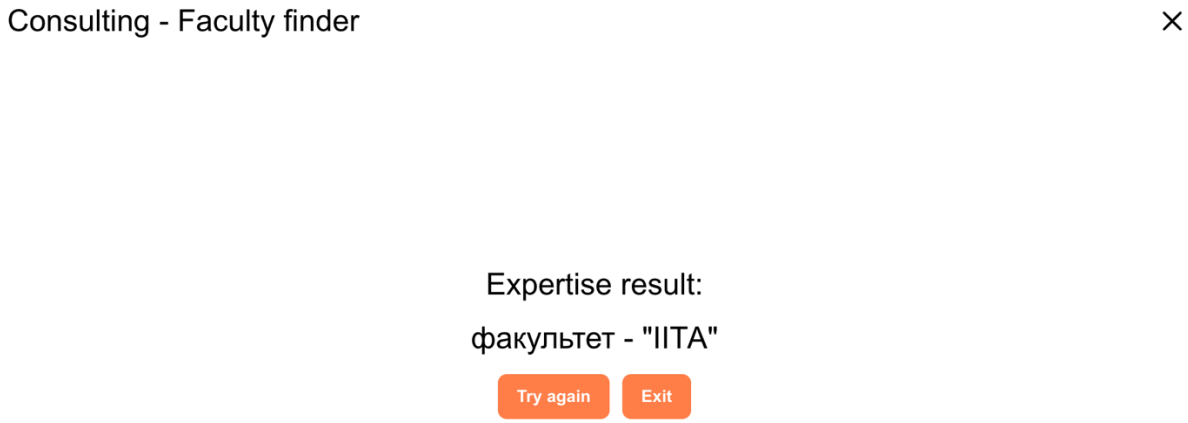


Рисунок Г.4 – Результат проходження консультації

Додаток Д

Акт впровадження

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

Довідка дана студенту групи 1КН-21м Шептякову Ігорю Олександровичу 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