Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Автоматизована система (конфігуратор) аналізу та підбору мережевого обладнання з прогнозуванням ефективності роботи
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШИи-21-2
Руслан Абдукарімов
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Євген Павленко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Абдукарімов Руслан Артурович
(прізвище, ім'я, по батькові)

1. Тема роботи Автоматизована система (конфігуратор) аналізу та підбору мережевого обладнання з прогнозуванням ефективності роботи

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25 червня 20 25 р.
3. Вихідні дані до роботи Python, HTML, Tailwind CSS, JavaScript, Visual Studio Code, Scikit-learn, FastAPI, LLM, GeminiAPI, React, Vite, Random Forest

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Проєктування та методи реалізації системи
3) Розробка вебзастосунку
4) Огляд інтерфейсу модулів супроводу і надання рекомендацій

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05 – 28.05	виконано
3	Постановка задачі	29.05 – 01.06	виконано
4	Обґрунтування вибору використаних технологій	02.06 – 03.06	виконано
5	Програмна реалізація	04.06 – 12.06	виконано
6	Оформлення пояснювальної записки	13.06 – 20.06	виконано
7	Підготовка до захисту	21.06 – 24.06	виконано
8	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Євген Павленко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 76 с., 19 рис., 5 табл., 3 дод., 23 джерела.

ВЕБЗАСТОСУНОК, КОНФІГУРАТОР, МЕРЕЖЕВЕ ОБЛАДНАННЯ, ПРОГНОЗ ЕФЕКТИВНОСТІ, РЕКОМЕНДАЦІЙНА СИСТЕМА, API, GEMINI, UX.

Об’єктом дослідження є процес проектування та підбору обладнання для побудови мережевої інфраструктури.

Предметом дослідження є автоматизований вебзастосунок інтелектуального конфігурування мережевого обладнання, що використовує рекомендаційну логіку, методи машинного навчання, моделювання сценаріїв поведінки мережі та вбудованого чат-бота консультанта для покращення взаємодії з користувачем.

Мета роботи – розробити вебзастосунок, що надає автоматизований підбір мережевого обладнання відповідно до вимог користувача, обґрунтовує вибір та прогнозує ефективність роботи в заданих умовах. А також надає знання користувачу в телекомунікаційних технологіях.

Методи дослідження – побудова архітектури клієнт-серверної системи з інтеграцією rule-based підходів та ML-алгоритмів. Моделювання роботи під навантаженням, реалізація REST API та дослідження інтерфейсу. Застосування мовної моделі, що покращує взаємодію з користувачем.

У результаті дослідження було створено вебзастосунок, що дозволяє виконати ефективний підбір обладнання, спрогнозувати поведінку мережі при навантаженнях та підвищити обізнаність.

Дослідження демонструє приклад практичного застосування мовних моделей та пояснювального ШІ у проєктуванні інтелектуальних систем.

ABSTRACT

Bachelor’s thesis contains: 76 pp., 19 fig., 5 tabl., 3 ann., 23 references.

API, CONFIGURATOR, GEMINI, NETWORK EQUIPMENT, PERFORMANCE FORECAST, RECOMMENDATION SYSTEM, UX, WEB APPLICATION.

The object of the study is the process of designing and selecting equipment for building a network infrastructure.

The subject of the study is an automated web application for intelligent configuration of network equipment, which utilizes recommendation logic, machine learning methods, network behavior scenario modeling, and an integrated chatbot assistant to improve user interaction.

The goal of the work is to develop a web application that enables automated selection of network equipment according to user requirements, provides justification for the choice, and forecasts system performance under given conditions. Additionally, it aims to provide users with knowledge in the field of telecommunication technologies.

The research methods include the development of a client-server system architecture with integration of rule-based approaches and machine learning algorithms, simulation of performance under load, implementation of a REST API, and user interface research. A language model is applied to enhance user interaction.

As a result of the study, a web application was developed that enables efficient equipment selection, forecasts network behavior under load, and increases user awareness.

The study demonstrates a practical example of applying language models and explainable AI in the design of intelligent systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ТКМ – телекомунікаційні мережі;

ШІ – штучний інтелект;

API – Application Programming Interface – програмний інтерфейс;

CAPsMAN – Controlled Access Point System Manager – централізоване керування точками доступу;

CRM – Customer Relationship Management – система керування взаєминами з клієнтами;

CSS – Cascading Style Sheets – каскадні таблиці стилів;

GPT – Generative Pre-trained Transformer – генеративна модель на основі трансформера;

HTML – HyperText Markup Language – мова розмітки гіпертексту;

HTTP – Hypertext Transfer Protocol – протокол передавання гіпертексту;

IP – Internet Protocol – інтернет-протокол;

IT – Information Technology – інформаційні технології;

JSON – JavaScript Object Notation – запис об’єктів JavaScript;

JSX – JavaScript XML – розширення синтаксису для React;

LLM – Large Language Model – велика мовна модель штучного інтелекту;

ML – Machine Learning – машинне навчання, підмножина штучного інтелекту;

NGN – Next Generation Network – мережа наступного покоління;

OSI – Open Systems Interconnection – модель мережевої взаємодії;

PoE – Power over Ethernet – передавання живлення через Ethernet;

QoS – Quality of Service – керування якістю обслуговування трафіку;

RAM – Random Access Memory – оперативна пам’ять з довільним доступом;

UI – User Interface – користувацький інтерфейс;

UX – User Experience – досвід користувача;

VLAN – Virtual Local Area Network – віртуальна локальна мережа;

VPN – Virtual Private Network – віртуальна приватна мережа;

VoIP – Voice over IP – передавання голосу через IP-мережі;

Wi-Fi – Wireless Fidelity – бездротова технологія зв’язку;

XML – eXtensible Markup Language – розширювана мова розмітки;

Firewall – мережевий екран, система фільтрації мережевого трафіку для захисту від несанкціонованого доступу;

IP-камера – мережева камера відеоспостереження, що використовує IP-протокол;

IP-телефон – мережева телефонна трубка для VoIP-зв’язку;

Комутація – процес з’єднання мережевих пристроїв на канальному рівні для передавання даних у межах локальної мережі;

Комутатор – пристрій для з’єднання пристроїв у локальній мережі з комутацією на рівні канального рівня;

Локальна мережа – комп’ютерна мережа, що охоплює обмежену географічну область, наприклад, офіс або будівлю;

Маршрутизатор – мережевий пристрій для маршрутизації трафіку між підмережами;

Маршрутизація – процес визначення шляху для передавання даних між різними мережами або підмережами;

Модель OSI – модель взаємодії комп’ютерних мереж яка складається з семі рівнів;

Точка доступу – пристрій для бездротового підключення користувачів до локальної мережі.

ВСТУП

В сучасних реаліях з розвитком телекомунікаційних технологій сильно змінилися способи комунікації та взаємодії. Якщо ще кілька десятиліть тому обмін даними був повільним і ресурсоємним, то сьогодні швидкість і надійність комунікацій стали критичними факторами успішної діяльності бізнесу. Особливу роль у цьому відіграє побудова стабільної, масштабованої та безпечної мережевої інфраструктури.

Разом з тим, процес вибору мережевого обладнання часто є складним, потребує фахових технічних знань і досвіду. Багато підприємств малого та середнього бізнесу не мають у штаті кваліфікованих ІТ-фахівців, здатних об'єктивно оцінити потреби, спрогнозувати майбутні навантаження та підібрати оптимальне обладнання з урахуванням бюджету й перспектив розширення. Це призводить до закупівлі неефективних або надлишкових пристроїв, що ускладнює масштабування, створює «вузькі місця» в мережі та збільшує витрати.

За статистикою необміркований вибір мережевого обладнання є частим явищем. Це в свою чергу призводить до нестабільної і повільної робити мережи, створює вразливості до витоків приватних даних, така мережа є привабливою до спроб зламу і шантажу зі сторони зловмисників.

Одним із рішень такої проблеми є впровадження автоматизованих систем – конфігураторів, які дозволяють здійснювати підбір мережевого обладнання за введеними параметрами. Проте більшість наявних конфігураторів на теренах інтернету розраховані на користувачів із глибокими технічними знаннями, що унеможливлює їх ефективне використання в не підготовленому середовищі.

В умовах, коли малий та середній бізнес часто не можуть дозволити собі висококваліфікованих ІТ-фахівців, актуальним стає створення інтелектуального вебзастосунку, який здатний самостійно проаналізувати

потреби користувача, надати обґрунтовані рекомендації та прогнозувати ефективність обраної конфігурації в реальних умовах.

З розвитком великих мовних моделей (LLM), з’являються нові інструменти і можливості для покращення взаємодії користувача з інтерфейсом та підвищення рівня обізнаності.

В той час коли наявні рішення які можна знайти на теренах інтернету пропонують лише рішення більш схожі на звичайні калькулятори, де користувач повинен розбиратися у багатьох змінних, створення дружнього до користувача вебзастосунку набуває особливої актуальності в бізнес-середовищі.

Дане дослідження присвячене розробці вебзастосунку, який поєднує в собі зручний і зрозумілий інтерфейс, систему рекомендацій та прогнозування поведінки запропонованого обладнання в майбутньому. Також у межах проекту було використано велику мовну модель Gemini, у ролі «мережевого знавця» для покращення взаємодії і допомоги користувачам з незрозумілими термінами та явищами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

В умовах епохи цифровізації, телекомунікаційні мережі (ТКМ) є серцем побудови бізнес-процесів і набувають критичного значення для широкого спектра організацій – від малого бізнесу до великих підприємств. Значення мережевих структур в умовах цифрової трансформації полягає у можливості швидкої адаптації до змін ринкового середовища, зниженні операційних витрат, підвищенні гнучкості та ефективності виробничих процесів. Управління розвитком мережевих структур в умовах цифрової економіки вимагає застосування нових стратегічних підходів, серед яких виділяється розробка гнучких моделей управління, які забезпечують можливість швидкої адаптації компанії до змін у зовнішньому середовищі та забезпечувати ефективну координацію між різними учасниками мережі [1].

Також мережеві структури сприяють розвитку співробітництва між компаніями, що раніше конкурували, та підвищують загальну конкурентоспроможність через створення інноваційних екосистем на засадах розвитку цифрових платформ для організації спільної роботи, обміну знаннями та досвідом, що дозволяє їм швидко реагувати на зміни ринку та впроваджувати нові продукти і послуги [1]. Саме мережеве обладнання забезпечує комунікацію підприємства як всередині так і з ззовні, від нього залежить передача даних, доступ до інтернету, інтеграції з зовнішніми сервісами та безпека приватних даних.

Галузь мережевих технологій охоплює комплексні системи з різними рівнями складності і функціональними особливостями, які включають маршрутизатори, комутатори, точки доступу Wi-Fi. Вимоги до цих систем можна розділити на два основних напрями, апаратний і функціональний. Апаратні вимоги базуються безпосередньо на компонентах «заліза», які

забезпечують стабільну роботу. Основі вимоги стосуються таких складових «заліза» як процесор – керує роботою пристрою та обробляє трафік; оперативної пам’яті (RAM) – відповідає за стабільність під час навантаження; флеш-пам’ять – зберігає налаштування та систему; кількість портів – визначає, скільки пристроїв можна підключити кабелем; покриття сигналу – вказує, яку площу охоплює Wi-Fi; система охолодження – захищає від перегріву; форм-фактор – описує тип корпусу та спосіб монтажу. Апаратні вимоги формуються з урахуванням характеру навантаження та призначення мережі й слугують орієнтиром для вибору обладнання, здатного підтримувати її стабільну та ефективну роботу. Вони дозволяють забезпечити відповідність між можливостями пристроїв і реальними умовами експлуатації, зокрема кількістю підключень, обсягом трафіку та особливостями середовища, в якому розгортається мережа.

Функціональні вимоги ґрунтуються на можливостях мережевого обладнання виконувати конкретні задачі, пов’язані з безпекою, стабільністю та керуванням трафіком. Саме ця функціональність не менш важлива при виборі обладнання і реалізації побудови мереж наступного покоління (NGN) [2].

Основні вимоги охоплюють такі функції, як брандмауер (Firewall) – забезпечує фільтрацію даних і контроль доступу; VPN – створює захищені канали зв’язку між офісами або віддаленими користувачами; QoS – дозволяє пріоритезувати критичний трафік (наприклад, VoIP або відео); VLAN – забезпечує логічне розділення мережі на сегменти; CAPsMAN або Controller Mode – дає змогу централізовано керувати точками доступу; балансування навантаження та резервування каналів – підтримують стабільність роботи при помилках з’єднання. Функціональні вимоги визначають, які можливості має підтримувати обладнання для безпечної, стабільної та керованої роботи мережі. Вони допомагають підібрати пристрої відповідно до сценаріїв використання, масштабу офісу та потреб у розширенні.

Згідно з дослідженнями, проведеними М. Кастельсом [3] та Р. Коглі і Н. Мелвіллом [4], мережеві структури поступово стають новою організаційною формою, що витісняє традиційні ієрархічні моделі. Такі структури дозволяють підприємствам швидко реагувати на зміни ринкового середовища, об’єднувати знання та ресурси, а також ефективно співпрацювати з іншими учасниками економічної екосистеми. У працях Дж. Парнелла [5] та П. Датти [6] мережеві моделі розглядаються як особливий тип організаційних форм, що забезпечують компаніям гнучкість, адаптивність і здатність до інновацій – ключові чинники успіху в умовах постійних змін. Інші автори, зокрема Ф. Ніколс [7] та Д. Альходарі [8], акцентують увагу на тому, що мережеві структури сприяють більш ефективному використанню ресурсів, що є особливо актуальним для промислових підприємств, які функціонують в умовах жорсткої конкуренції. Вони також підкреслюють важливість співпраці в межах мереж, яка забезпечує обмін знаннями, технологіями та інноваціями, підвищуючи загальний рівень конкурентоспроможності.

1.2 Актуальність проблеми

Процес підбору мережевого обладнання є складним і непрозорим. Для виконання такого завдання вимагається розуміння характеристик пристроїв і основ ТКМ. Необхідно враховувати кількість користувачів, типи пристроїв, характер трафіку, вимоги до безпеки, швидкість доступу до інтернету та інші фактори. Такий процес потребує комплексного аналізу, який зараз можуть надати лише профільні фахівці. Користувачу потрібно не лише обрати відповідний пристрій а також і враховувати два основні моменти – розташування пристроїв та бажану швидкість мережі. Існують три типи мережі – дротові, бездротові та гібридні [9].

Проблема ускладнюється тим, що у малому та середньому бізнесі, який є основою економіки багатьох країн, часто відсутні кваліфіковані

фахівці. У таких умовах рішення щодо закупівлі обладнання приймаються людьми без належної технічної освіти. Вони ґрунтуються на загальних описах, цінах або рекомендаціях продавців, які, у свою чергу, не завжди враховують специфіку бізнесу клієнта. У результаті, ТКМ будуються на обладнанні, що не забезпечує належного рівня стабільності при навантаженні, може мати надлишкову функціональність яка підвищує вартість, або не передбачає можливостей масштабування та гнучкого налаштування. Це не лише стримує розвиток компанії, а й може призводити до збоїв критичних систем, втрати конфіденційних даних та підвищення витрат на відновлення і переобладнання.

У ході дослідження в рамках дипломного проєкту були оглянути спеціальні конфігуратори – за допомогою яких можна підібрати обладнання. Проте ці рішення не призначені для масового використання, а орієнтовані на профільних фахівців. Досліджені конфігуратори не містять підказок, не надають пояснень та використовують термінологію, яка є незрозумілою для масового користувача. Таким чином, користувач без відповідних знань стикається з десятками моделей пристроїв, специфікацій, стандартів і скорочень. Це призводить до хибних рішень і користувач «тицяє пальцем у небо».

Варто зазначити, що з появою великих мовних моделей (LLM), зокрема таких як ChatGPT або Gemini, користувачі отримали доступ до інструментів, які здатні в режимі діалогу підібрати обладнання за запитом і пояснити свої рекомендації. Проте LLM не враховують конкретні обмеження інфраструктури через брак даних, викликаний нестачею знань у користувача, через що подані відповіді можуть містити неточності. Більше того, навіть із поясненням користувач без технічного підґрунтя все одно не завжди здатний самостійно інтерпретувати надану інформацію правильно.

У висновку простежується чітка потреба у простому, інтуїтивно зрозумілому, але водночас інтелектуальному інструменті. Такий інструмент має аналізувати потреби користувача за допомогою зрозумілих

параметрів (кількість працівників, площа приміщень, типи пристроїв тощо), надавати аргументовані рекомендації з поясненням, враховувати можливість майбутнього розширення, прогнозувати ефективність роботи запропонованої конфігурації, а також супроводжувати користувача у процесі вибору та надавати відповіді на запитання підвищуючи рівень знань в ТКМ. Наявність такої системи дозволить спростити процес взаємодії для користувачів без належного рівня знань, зменшити ризики під час ухвалення рішень.

1.3 Складність вибору мережевого обладнання

На перший погляд, для створення мережевої інфраструктури підприємства достатньо просто підібрати маршрутизатор, комутатор і кілька точок доступу Wi-Fi. Проте на практиці цей процес є набагато складнішим, ніж здається. Це зумовлено великою кількістю технічних параметрів, які безпосередньо впливають на стабільність, безпеку, надійність і масштабованість мережі.

Найпоширенішою помилкою є підбір обладнання лише на основі кількості співробітників. Проте при виборі конфігурації варто враховувати і характер виконуваної роботи, обсяги використання трафіку, типи підключених пристроїв, наявність відеоспостереження, серверів або хмарних сервісів, а також перспективи розширення офісу. Наприклад, підприємство з власною CRM-системою і IP-камерами потребуватиме потужніших рішень, ніж підприємство, що працює лише з електронною поштою й текстовими документами. Практика показує, що відсутність чіткого уявлення про ці чинники часто призводить до закупівлі або надто слабкого, або надто дорогого обладнання.

Одним із найважливіших елементів є вибір маршрутизатора (router) – серця мережі. Якщо пристрій не здатен обробляти велику кількість паралельних запитів, це призводить до перевантажень, затримок та

нестабільної роботи. Брак обчислювальної потужності спричиняє ситуації, коли пристрої конкурують за ресурси, що негативно впливає на загальну продуктивність. Крім того, дешеві моделі зазвичай не мають гнучких налаштувань, що обмежує можливості контролю та знижує рівень захищеності мережі. Вразливості конфігурації створюють ризики витоку конфіденційної інформації й серйозно ускладнюють подальше масштабування інфраструктури.

Не менш важливою є проблема вибору комутаторів (switch). Тут потрібно врахувати, чи потрібна підтримка PoE для живлення IP-камер або точок доступу Wi-Fi, скільки портів потрібно на перспективу, чи передбачена можливість каскадного з’єднання декількох пристроїв. Якщо ці аспекти не враховуються одразу, то під час розширення офісу доводиться докуповувати або навіть повністю замінювати обладнання, що призводить до додаткових витрат.

Ще одним фактором є рівень шуму, який створює обладнання. Для цього виробники пропонують два типи обладнання – із пасивним та активним охолодженням. Перший працює безшумно, але має нижчу продуктивність, це може спричинити проблеми при високих навантаженнях. Другий витримує великі навантаження, але через наявність вентиляторів може бути досить гучним. Це вимагає додаткового виділення окремого приміщення. Також для першого критично, щоб температура в приміщенні з обладнанням не перевищувала допустимі межі, інакше виникає ризик перегріву, натомість другий менш вразливий до зовнішніх температур.

Підбір точок доступу Wi-Fi також має свою специфіку. Потрібно врахувати очікуване навантаження на одну точку, площу покриття, товщину та матеріал стін, а також підтримку стандартів пристроями працівників. Додатково бажаним є розмежування гостьової та внутрішньої мережі, що позитивно впливає на безпеку. При неправильному виборі точки доступу,

так само як і в випадку з маршрутизатором пристрої конкуруватимуть за ресурси, що негативно вплине на загальну продуктивність.

Крім наведеного, при виборі обладнання варто зважати на споживання електроенергії, наявність вбудованого фаєрволу, VPN-клієнта, централізованого управління, підтримку від виробника та сумісність з вже наявною інфраструктурою. Усі ці параметри створюють багато рівнів складності, які потрібно врахувати одночасно.

Навіть маючи базові знання, користувачу доводиться опрацьовувати великі обсяги інформації: технічні специфікації, таблиці порівнянь, відгуки, огляди, рекомендації. Це перетворює вибір обладнання на трудомісткий, неструктурований і часто інтуїтивний процес, у якому легко припуститися критичної помилки.

У підсумку, стає очевидним, що процес вибору мережевого обладнання – це не банальна покупка роутера з Wi-Fi, а складна інженерна задача. І саме тому виникає потреба у системі, яка здатна автоматизувати ключові етапи цього вибору, підказати логіку рішення, надати пояснення, врахувати сценарії навантажень і допомогти користувачу уникнути хибного рішення навіть.

1.4 Аналіз наявних рішень

З огляду на трудомісткість процесу, природно, що на ринку з’явилися спроби автоматизувати процес підбору обладнання. Деякі виробники і торгові майданчики вже пропонують вебзастосунки, які дозволяють фахівцю підібрати певні моделі комутаторів, маршрутизаторів, точок доступу та навіть цілих серверних шаф залежно від обраних параметрів. Однак реальна ефективність таких рішень для практичного використання масовим користувачем є вкрай обмеженою.

По-перше, більшість конфігураторів які можна знайти в інтернеті є жорстко прив’язаними до каталогу конкретного виробника або платформи.

Вони не дають змоги збирати конфігурацію обладнання з різних брендів, і що найголовніше, вони не оцінюють доцільність вибору в контексті майбутніх навантажень чи можливості масштабування.

По-друге, інтерфейс багатьох таких систем є технічно складним, не інтуїтивним і орієнтованим на фахівців, які вже добре розуміють абревіатури, стандарти та специфікації. Введення параметрів часто вимагає знань. Більше того, система не пояснює, чому запропонована конфігурація є оптимальною. Користувач отримує результат без аргументації, що знижує довіру до такого інструменту і не дозволяє навчитись чомусь у процесі.

По-третє, більшість конфігураторів не враховують реальні фізичні умови підприємств: кількість кімнат і поверхів, типи пристроїв, тепло та шумові характеристики обладнання. Вони ігнорують практичні обмеження, які мають ключове значення при побудові ТКМ в реальному середовищі. У кращому випадку, це формальні підказки на кшталт рекомендується гігабітний порт, без урахування контексту.

Ще однією критичною прогалиною є відсутність прогнозування. Жоден із доступних конфігураторів не пропонує оцінки майбутньої ефективності мережі з урахуванням введених даних. Користувач бачить лише список моделей, але не отримує відповідь на ключове питання: «як ця конфігурація працюватиме за збільшенням навантаження?».

Окремо варто згадати, що такі сервіси фактично не орієнтовані на освітній ефект. Користувач нічого не дізнається про те, як працює мережа, на що впливають ті чи інші параметри, як уникнути типових помилок. Система виконує сухий розрахунок, але не навчає і не пояснює, що особливо важливо.

Підсумок наступний, попри формальну наявність на теренах інтернету великої кількості рішень для підбору обладнання, вони мають суттєві обмеження у функціональності, доступності, гнучкості та корисності для реального, недостатньо обізнаного користувача. Це створює чітку нішу для розробки нової системи, не просто конфігуратора, а

інтелектуального асистента, що поєднує аналітику, рекомендації, навчання та підтримку.

1.5 Перспективи інтелектуалізації підбору

З огляду на складність процесу вибору мережевого обладнання, яка була проаналізована в попередніх підрозділах, цілком очевидним стає необхідність пошуку нових підходів, які дозволяють автоматизувати та водночас спростити цей процес. Одним із таких напрямів є застосування технологій штучного інтелекту, а саме інтелектуалізація конфігурування.

На відміну від звичайних конфігураторів, які працюють за заздалегідь визначеними жорсткими правилами, інтелектуальні системи здатні приймати рішення на основі аналізу вхідних даних, врахування контексту, сценаріїв використання та типу трафіку. Це дозволяє перейти від простого фільтрування параметрів до формування гнучких, аргументованих рекомендацій.

Алгоритми машинного навчання можуть аналізувати сукупність введених параметрів. Таких як кількість пристроїв, типи пристроїв, площа офісу, тип навантаження. На основі пропонуються варіанти, які не просто відповідають параметрам, а є найбільш доцільними з точки зору надійності, безпеки, масштабованості та ціни.

Ще однією важливою перевагою інтелектуального підходу є можливість прогнозування ефективності конфігурації. Завдяки використанню моделей, навчених на симуляціях, система може оцінити, як обране обладнання покаже себе при збільшенні навантаження, додавання нових пристроїв або появи додаткових сервісів. Це дозволяє уникнути перевитрат і зменшити ризики, пов’язані з недооцінкою майбутніх потреб.

Окрему цінність має реалізація функціональності пояснень. Користувач має не просто отримати варіант конфігурації, а й зрозуміти, чому саме ці пристрої рекомендовані.

Інтеграція великої мовної моделі (LLM) у ролі чат-бота консультанта, відкриває новий рівень взаємодії. Замість того, щоб заповнювати складні форми, користувач може вести діалог: пояснювати свої потреби, отримувати пояснення, ставити уточнюючі запитання. Це особливо важливо для користувачів, без технічного підґрунтя, які потребують не лише наявність інструмента, а й супроводу в процесі ухвалення рішення.

Це свідчить про те, що інтелектуалізація процесу підбору мережевого обладнання – це не лише про зручність, а й реальне підвищення точності, надійності та гнучкості системи. Вона дає змогу створити інструмент, який не просто допомагає обрати, а дійсно консультує, аналізує, адаптується та вчить, що є особливо цінним.

1.6 Постановка задачі

В ході дослідження було детально проаналізовано складність вибору мережевого обладнання, обмеження наявних інструментів та перспективи застосування інтелектуальних технологій у цьому процесі. На основі отриманих даних, розробляється інтелектуальна система, призначена для полегшення процесу побудови телекомунікаційних мереж, супроводжуючи користувача на всіх етапах. Актуальність цього проєкту зумовлена потребою у засобах, що не лише надають списки обладнання, але й сприяють підвищенню рівня знань у телекомунікаційних мережах.

Об’єктом дослідження є мережевого обладнання, представлений у вигляді конфігуратора, що надає рекомендації по вибору, прогнозує роботу під навантаженням та вчить.

Об’єктом дослідження є процес проектування та підбору обладнання для побудови мережевої інфраструктури.

Предметом дослідження є автоматизований вебзастосунок інтелектуального конфігурування мережевого обладнання, що використовує рекомендаційну логіку, методи машинного навчання,

моделювання сценаріїв поведінки мережі та вбудованого чат-бота консультанта для покращення взаємодії з користувачем.

Метою даної дипломної роботи є розробити вебзастосунок, що надає автоматизований підбір мережевого обладнання відповідно до вимог користувача, обґрунтовує вибір та прогнозує ефективність роботи мережі в заданих умовах. А також підвищення обізнаності користувача у базових поняттях телекомунікаційних технологій.

1.7 Визначення вимог

Вебзастосунок орієнтований на масового користувача, тому його функціонал повинен бути побудований таким чином, щоб у процесі взаємодії не виникала необхідність звертатися до зовнішніх джерел, вивчати специфікації чи технічну термінологію. Система має забезпечувати виконання низки функціональних завдань, які формуються на основі аналізу актуальних проблем та сценаріїв використання.

Спочатку користувач заповнює послідовну форму конфігуратора, в якій вказує такі параметри: кількість і типи пристроїв, кабінетів і поверхів, швидкість інтернету, потребу в Wi-Fi, VPN, фаєрволах, побажання щодо бренду тощо. Кожен етап супроводжується поясненням, що саме означає параметр і як він вплине на кінцеву конфігурацію.

На основі введених даних система аналізує, та формулює кілька варіантів конфігурацій обладнання. Для цього використовується комбінація логіки з фіксованими правилами (rule-based) та моделей машинного навчання (ML), які надають варіанти надійних конфігурацій. Кожен варіант має супроводжуватися прогнозом ефективності, чому саме це обладнання рекомендовано, які параметри на це вплинули і як це вплине на стабільність телекомунікаційної мережі.

Також у систему має бути інтегрований інтерактивний помічник, який може відповідати на запитання користувача щодо обладнання, технічних

понять і супроводжувати по кожному етапу. Це дозволяє отримати додаткові пояснення у зручній, діалоговій формі, без пошуку інформації на сторонніх ресурсах.

Інтерфейс з адаптацією під користувача який не має повноцінного досвіду роботи з мережами: мінімалізм, підказки, пояснення кожного кроку, візуальні підказки, усе повинно бути направлено на зниження когнітивного навантаження та покращувати отриманий досвід.

Можливість редагування введених параметрів без втрати прогресу: користувач повинен мати змогу повернутися до попередніх кроків, змінити дані та отримати нові результати.

Система не передбачає автоматичного замовлення або закупівлі обладнання, інтеграції з платіжними платформами чи службами доставки. Її призначення полягає саме у допомозі вибору, а не у реалізації повного процесу закупівлі.

2 ПРОЄКТУВАННЯ ТА МЕТОДИ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Архітектура програмної частини

Проєктування автоматизованої системи аналізу та підбору мережевого обладнання ґрунтується на трирівневій архітектурі, яка відповідає сучасним тенденціям, є універсальною і придатною для реалізації масштабованих, динамічних і обслуговуваних інформаційних систем. У межах цієї архітектури кожен рівень виконує чітко визначену функцію, що дозволяє забезпечити незалежність розробки окремих частин системи.

Перший рівень, інтерфейс користувача, виконує функції візуалізації та збору вхідних даних, а також надання результатів, рекомендацій і прогнозів користувачу. Реалізований у вигляді вебінтерфейсу, створеного за допомогою бібліотеки React. Цей рівень спроєктований для спрощення взаємодії користувача з програмним забезпеченням, подання процесу конфігурації логічно структурованою, покроковою і зрозумілою. При цьому всі зміни, які відбуваються у ході взаємодії, синхронізуються з прикладним рівнем системи через API-запити.

Другий рівень, прикладний, зосереджений на обробці введених даних, реалізації логіки підбору конфігурацій мережевого обладнання, оцінки відповідності технічним критеріям, а також виконанні модулів прогнозування ефективності. Саме на цьому рівні відбувається основна обчислювальна діяльність, яка аналізує введені параметри і розбудовує індивідуальні рекомендації. Архітектурно прикладний рівень розділений на кілька модулів, де кожен реалізує окрему підфункцію системи, що дозволяє дотримуватись принципу інкапсуляції та мінімізувати зв’язність між компонентами.

Третій рівень, це дані та машинне навчання. Рівень відповідає за структурованої інформації про доступне мережеве обладнання, його технічні характеристики, типи пристроїв, протоколи та інші параметри, що

використовуються для аналізу. На цьому ж рівні розташовано модель прогнозу, машинне навчання (ML), нетреноване на основі умовно-синтетичних даних. Яка дозволяє зробити попереднє оцінювання ефективності запропонованого рішення за низкою факторів, що не є безпосередньо видимими на момент вибору, а саме стабільність роботи під навантаженням, можливість масштабування, вразливість до помилок користувача тощо.

При проєктуванні вебзастосунку закладено принцип суворого розмежування функціональності між логічними частинами. Це забезпечує структурованість програмного коду, сприяє його повторному використанню та підвищує стабільність під час майбутніх модифікацій. Відмова від монолітного підходу дозволила створити гнучку, модульну систему, яку можна поступово розширювати або адаптувати під нові вимоги, не порушуючи вже реалізовану логіку.

У підсумку, архітектура системи є результатом поєднання сучасних підходів до організації програмного середовища з практичними вимогами до зручності використання, надійності та обґрунтованості рішень. Саме це забезпечує не лише працездатність платформи, а й її структурну якість, відповідність загальним стандартам у галузі створення програмних засобів підтримки прийняття рішень.

2.2 Формалізація критеріїв підбору

В основі системи автоматизованого підбору мережевого обладнання застосовано механізм формалізованого підходу до представлення вимог користувача, що дозволяє перетворити суб’єктивні або описові характеристики телекомунікаційних мереж на чіткі параметри, придатні для автоматизованої обробки. Це є ключовим етапом у процесі прийняття рішень, адже саме за допомогою цих параметрів система здійснює фільтрацію та вибір обладнання з попередньо визначеної бази.

Параметри користувача збираються в окремі логічні блоки, же кожен виконує окрему роль. Ці блоки реалізовані у вигляді полів об'єкта ConfigRequest, структура якого визначена у відповідному модулі схем. Передача даних здійснюється у форматі JSON за допомогою HTTP-запиту до серверної частини системи.

Першу групу параметрів становлять дані о характеристиках фізичного середовища. Зокрема, передбачено введення типу приміщення, кількості поверхів та кількості кімнат на кожному поверсі. Ці відомості є необхідними для оцінювання зон покриття мережі, вибору типу точок доступу та рівня потужності обладнання.

Другу групу параметрів становлять дані про типи кінцевих пристроїв, які планується підключати до мережі. Користувач вказує кількість і типи пристроїв. Ці параметри впливають на розрахунок навантаження, визначення мінімальної пропускної здатності каналів, кількість необхідних портів комутатора або зону дії Wi-Fi.

Третю групу параметрів становлять дані, що стосуються підключення до інтернету. До них належать очікувана швидкість інтернету, зона покриття бездротового з’єднання та характер типового навантаження, а саме: робота в браузері, пошта, документи; Google Docs, відеоспостереження, відеоконференції; IP-телефони, CRM системи, велика кількість IP-камер, хмарні сервіси. Формалізоване подання цих характеристик дозволяє системі правильно обирати конфігурації обладнання за критеріями продуктивності.

Четверту групу параметрів становлять дані про специфічні умови підключення. Зокрема, користувач може вказати необхідність підтримки технологій PoE та кількість необхідних портів, а також визначає тип охолодження обладнання.

П’яту групу і останню групу параметрів становлять дані о базових функціональних вимогах. Зокрема, необхідність підтримки віртуальних приватних мереж (VPN), наявність міжмережевого екрану (Firewall), а

також можливість вибору конкретного виробника або делегування цього вибору системі.

Далі формалізовані дані передаються до модуля логіки рекомендацій, де на їх основі здійснюється відбір пристроїв з бази даних (БД). Фільтрація здійснюється за типом пристрою, виробником та класом продуктивності. Після цього система формує готові конфігурації з кількох пристроїв – маршрутизатора, комутатора та точки доступу – та формує результат у вигляді варіантів, які відповідають заданим критеріям.

Слід зазначити, що процес формалізації критеріїв у межах даної системи не зводиться до пасивного прийому та накопичення вхідних даних від користувача. Формалізація є активною частиною логіки обробки запиту, що охоплює декілька рівнів перетворення вхідної інформації, починаючи від первинної валідації структури і закінчуючи семантичним аналізом вмісту. На першому етапі здійснюється класифікація кожного параметра за типом: числовий, булевий, категоріальний або комбінований. Це дозволяє застосовувати відповідні алгоритми перевірки коректності, відповідності діапазонам значень та логічної сумісності з іншими параметрами.

Паралельно система виконує розділення введених параметрів на обов’язкові та факультативні. Обов’язкові параметри – це ті, без яких неможливо сформувати жодного допустимого технічного рішення, наприклад, тип з’єднання або кількість пристроїв. Їхня відсутність або некоректність блокує подальшу обробку запиту. Факультативні ж параметри не виключають побудови конфігурації, але впливають на її якість, пріоритетність, енергоспоживання чи функціональне навантаження. Наприклад, перевага певного виробника або бажана наявність VPN можуть змінити порядок у ранжуванні результатів.

На наступному етапі система виконує логіко-семантичну перевірку на предмет суперечностей між параметрами. Це забезпечує виключення ситуацій, коли одночасно задані несумісні вимоги, наприклад, підтримка великої кількості портів PoE при вимогах до безшумної пасивної вентиляції.

Такі колізії відслідковуються й обробляються або через винесення пріоритетів, або через надання альтернативних рішень з поясненнями.

Важливим елементом є й те, що на основі формалізованих параметрів система здатна здійснювати узагальнення запиту. Наприклад, якщо користувач не надає повної інформації щодо навантаження мережі, система може здійснити умовну класифікацію за непрямими ознаками: кількість пристроїв, тип приміщення, тип інтернет-з'єднання тощо. Це дозволяє зробити формалізацію не лише засобом представлення даних, а й засобом їхньої інтерпретації в умовах неповноти даних.

Завдяки багаторівневій та узгодженій формалізації критеріїв система здатна не просто відбирати обладнання за жорстко заданими фільтрами, а й здійснювати адаптивний підбір на основі поєднання технічної відповідності, оптимальності та доцільності. Це значно підвищує практичну цінність результатів і наближає функціональність системи до умов реального проєктування мережевої інфраструктури. Приклад спроєктованої схеми мережевої інфраструктури показано на рисунку 2.1.

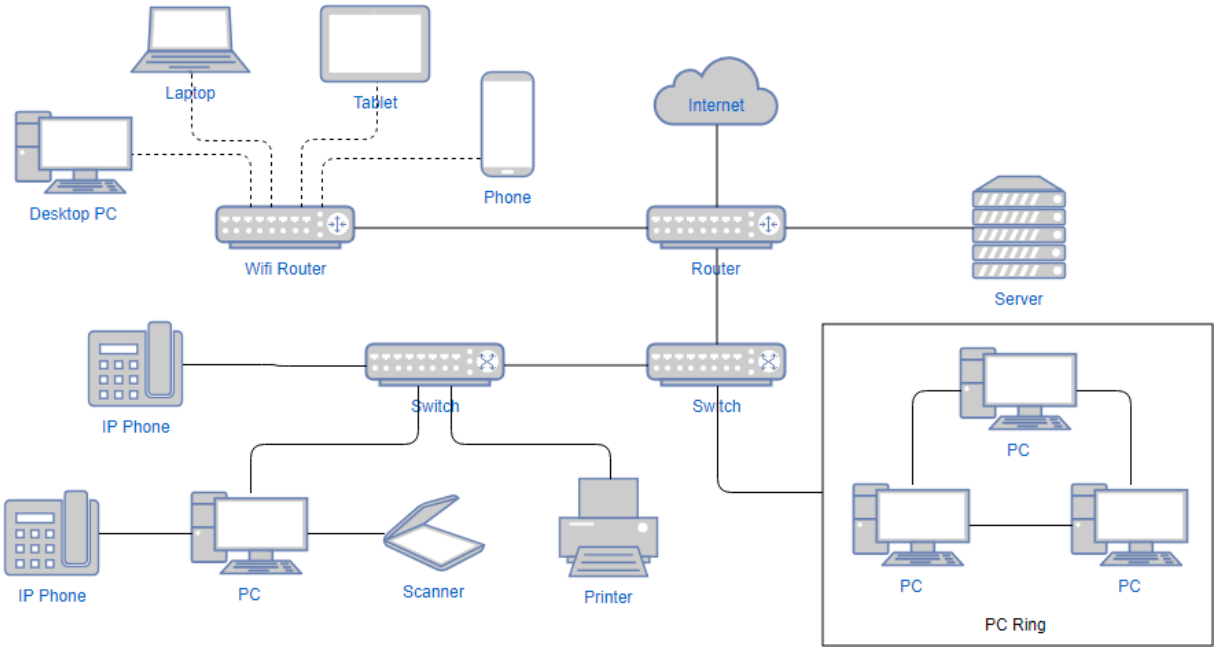

Рисунок 2.1 – Спроєктована схема мережевої інфраструктури

2.3 Алгоритми підбору та ранжування

Алгоритмічна основа підбору мережевого обладнання ґрунтується на комбінованому підході, що поєднує використання попередньо сформованої бази знань (у вигляді структури пристроїв із класифікаторами) та обмежуючих правил для формування відповідних конфігурацій. Результатом роботи цього алгоритму є набір рекомендованих рішень, що розділені за рівнем складності та функціональних можливостей.

Процес підбору реалізовано у вигляді послідовного фільтрування бази обладнання за ключовими критеріями, до яких належать тип пристрою (маршрутизатор, комутатор, точка доступу), виробник та рівень (basic, balanced, pro). На основі значень, переданих у запиті, система застосовує функцію фільтрації пристроїв, що здійснює пошук відповідного пристрою за заданими ознаками у сформованому переліку технічних моделей. У разі відповідності трьом основним атрибутам (vendor, type, tier) та параметрів заданих користувачем, пристрій включається до відповідної конфігурації.

Формування кінцевих варіантів реалізовано через функцію generate_recommendations, яка будує перелік конфігурацій за трьома сценаріями – базовим, збалансованим та професійним. У випадку, якщо користувач не задає виробника, система автоматично створює кілька комбінацій пристроїв із різних брендів, що дозволяє забезпечити вибір серед альтернатив за якістю, вартістю та надійністю.

Особливістю розробленого алгоритму є те, що кожна конфігурація включає повноцінний набір обладнання – мінімум по одному пристрою кожного типу – і побудована на принципі сумісності всередині одного рівня продуктивності. Таким чином, система не лише підбирає окремі пристрої, а формує завершене технічне рішення, придатне для практичного використання в умовах, заданих користувачем. Користувач отримує схему з

кількох конфігурацій обладнання і має варіанти для вибору. Схема наведена на рисунку 2.2.

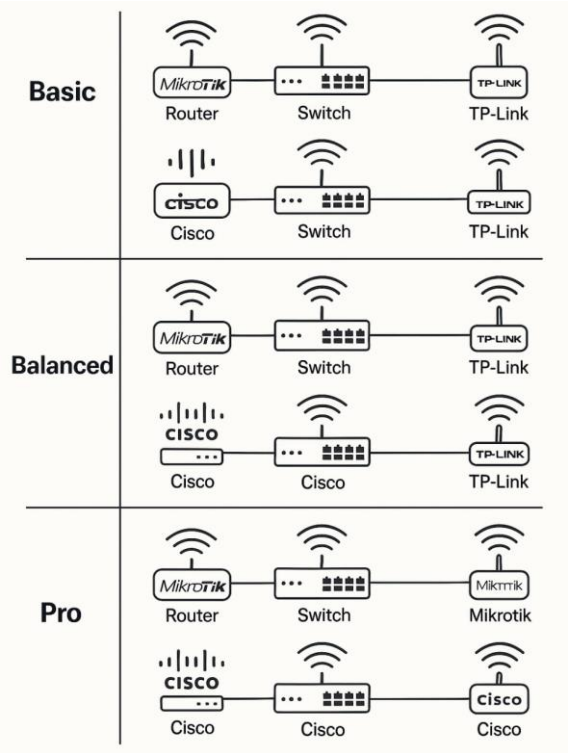

Рисунок 2.2 – Схема з кількох конфігурацій за трьома сценаріями

У підсумку, в системі реалізовано як послідовність рішень із обмежуючими правилами для конфігурацій, що забезпечує швидкість, передбачуваність і можливість масштабування алгоритму для подальших потреб або галузевих специфікацій.

2.4 Прогнозування ефективності

Функціонування мережевої інфраструктури залежить не лише від правильного підбору окремих елементів обладнання, а й від їхньої сукупної здатності ефективно взаємодіяти в умовах реального навантаження. У зв’язку з цим під час розроблення автоматизованої системи підбору було передбачено компонент, відповідальний за прогнозування ефективності

сформованої конфігурації. Цей компонент покликаний дати користувачу не просто набір пристроїв, що формально відповідають параметрам, а також попередню оцінку їхньої поведінки у прогнозованому середовищі експлуатації. Прогнозування реалізовано на основі моделі машинного навчання, яка натренована на умовно-симульованих конфігураціях та враховує взаємозв’язки між характеристиками користувацького середовища, типами обладнання та історичними сценаріями нестабільності. Модель збережена у форматі joblib та завантажується під час запуску модуля.

Загальна логіка роботи прогнозуючого модуля полягає у тому, щоб, спираючись на задані параметри – такі як кількість підключених пристроїв, пропускна здатність інтернет-з’єднання, наявність специфічних вимог – VPN, PoE, бездротовий доступ, а також на дані щодо рекомендованого обладнання, здійснити узагальнення та класифікацію сформованого рішення. У цьому модулі реалізовано модель машинного навчання, яка завантажується на сервері при запуску застосунку. Модель попередньо навчена на наборі штучно згенерованих або емпіричних даних, які відображають типові сценарії експлуатації мережі з різною конфігурацією обладнання.

Зібрані вхідні дані, що надходять від клієнтської частини системи, обробляються модулем predict_efficiency(). У цій функції здійснюється попередній аналіз структури параметрів, агрегація числових значень та трансформація строкових або логічних полів у числові ознаки, придатні для обробки навченою моделлю. Серед них – загальна кількість підключених пристроїв, швидкість інтернету, наявність бездротового інтерфейсу, кількість PoE-портів, вимоги до безпеки, тип навантаження тощо.

Після побудови вектора ознак система передає його на вхід класифікатору. Структура схеми взаємодії між вхідними параметрами, вектором ознак та модельним класифікатором подана на рисунку 2.3. На

виході модель повертає індекс класу, що відповідає одній із чотирьох категорій стабільності:

– клас 0, нестабільна конфігурація, що може призвести до системних збоїв;

– клас 1, допустима, але ризикована, з імовірністю деградації якості обслуговування;

– клас 2, стабільна під поточне навантаження, проте з обмеженим потенціалом до масштабування;

– клас 3, оптимальна з погляду продуктивності, надійності та перспектив подальшого розширення.

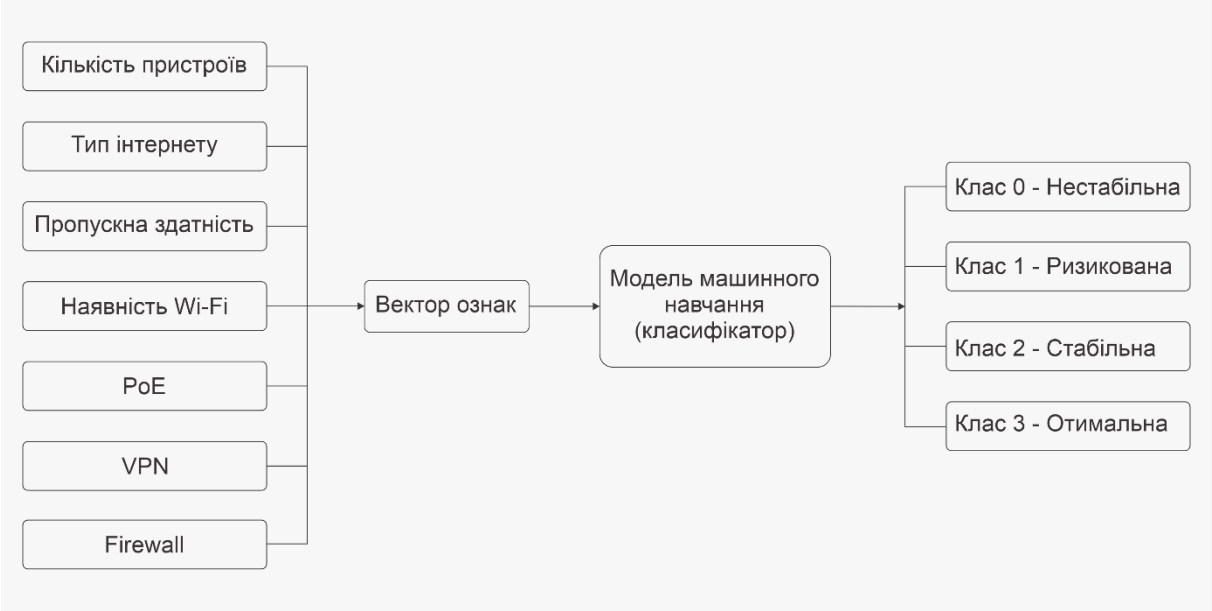

Рисунок 2.3 – Структура модуля прогнозування ефективності

Ці оцінки формуються та виводяться у зручному графічному вигляді для користувача. Такий підхід дозволяє мінімізувати ймовірність технічних помилок у конфігурації, а також спрощує процес ухвалення рішень користувачем, який не має глибоких технічних знань.

Крім того, прогнозуюча модель може бути надалі розширена через залучення нових змінних, таких як: споживання електроенергії, середній час

безвідмовної роботи, вартість обслуговування чи сумісність із майбутніми оновленнями протоколів. Це дає змогу перетворити систему на повноцінний інструмент підтримки проєктування телекомунікаційної інфраструктури.

Підсумовуючи, варто зазначити, що прогнозування ефективності в межах даної системи виконує функцію інтелектуального аудиту – воно зменшує ризики прийняття необґрунтованих рішень, підвищує якість рекомендацій.

3 РОЗРОБКА ВЕБЗАСТОСУНКУ

3.1 Використані технології

Під час розробки вебзастосунку було застосовано сучасний набір інструментів, що поєднує компоненти клієнтської та серверної частин у межах логіки клієнт-серверної архітектури. Даний підхід дозволяє забезпечити функціональну ізоляцію між інтерфейсом користувача, обчислювальними модулями та обробкою даних, що, у свою чергу, сприяє полегшенню підтримки, масштабування та зручності при розробці. Розмежування обов'язків між клієнтом і сервером реалізовано не лише не рівні коду, а й на рівні логіки взаємодії через API-запити.

3.1.1 Технології клієнтської частини

Клієнтська частина побудована з використанням React – бібліотеки JavaScript для створення інтерфейсів користувача. React дозволяє організувати кожен компонент вмісту і керувати візуальним потоком на основі внутрішнього стану застосунку [10]. У межах даного вебзастосунку кожен етап взаємодії – від вітального екрану до підсумку вибору і надання рекомендацій – реалізований як окремий як окремий функціональний компонент, що відповідає за певну частину логіки конфігуратора. Такий підхід забезпечує можливість ізолювати та повторно використовувати код без необхідності змінювати глобальну структуру [11].

Інтерфейс реалізовано у вигляді послідовного покрокового сценарію, де після заповнення кожного етапу відбувається оновлення вмісту без необхідності перезавантаження сторінки у браузері. Такий сценарій реалізований шляхом передачі стану між компонентами та умовної візуалізації в залежності від прогресу користувача. Це безпосередньо

дозволяє створити відчуття цілісного, інтерактивного застосунку, уникаючи зайвого навантаження на мережу чи браузер.

Ключову роль відіграє мова гіпертекстової розмітки HTML, яка формує основу візуального представлення інтерфейсу. HTML є загальноприйнятим стандартом для відображення інформації у вебсередовищі незалежно від типу комп'ютера або операційної системи. Основними перевагами цієї мови є її простота у використанні, платформна незалежність і підтримка всіма сучасними браузерами [23]. В межах React-архітектури використовується JSX, розширення синтаксису JavaScript, що дозволяє записувати HTML-подібні елементи безпосередньо в коді, усі вони в результаті трансформуються у стандартні HTML-теги. Завдяки цьому розробник отримує можливість гнучко створювати й структурувати вебсторінку, використовуючи знайомі концепції: , <section>, <form>, <input>, <button>, <ul>, <li> та інші. У межах цього проєкту, HTML задає ієрархію елементів, визначає порядок їхнього відображення та взаємозв'язки між ними. У поєднанні з Tailwind CSS мова розмітки забезпечує візуальну послідовність, адаптивність інтерфейсу та контроль над стилізацією.

Для стилізації інтерфейсу застосовано утилітарний CSS-фреймворк Tailwind CSS, який дозволяє замінити великі зовнішні таблиці стилів, замінюючи їх компактними класами безпосередньо у JSX-структурі компонентів. Це значно спрощує процес верстки та дозволяє легко масштабувати дизайн. Tailwind CSS працює шляхом сканування всіх наявних у проєкті HTML-файлів, компонентів JavaScript та інших шаблонів на наявність назв класів, що дозволяє згенерувати відповідні стилі, записані у статичний CSS-файл [12].

Збірка клієнтської частини реалізована з використанням інструменту Vite – сучасного збирача, який забезпечує високу продуктивність під час розробки завдяки використанню попередньої обробки модулів ES і миттєвому перезавантаженню при зміні коду [13].

Для забезпечення кросбраузерної сумісності стилів, інтегровано PostCSS у поєднанні з Autoprefixer. PostCSS виконує функцію обробника CSS-файлів, дозволяючи налаштовувати перетворення за допомогою плагінів. Autoprefixer, у свою чергу, автоматично додає необхідні вендорні префікси до CSS-властивостей, що гарантує однакове відображення стилів у всіх актуальних браузерах.

Серед візуальних бібліотек інтегровано react-icons, яка надає зручний інтерфейс для вставки масштабованих іконок із різних пакетів, та tsparticles, яка використовується для створення динамічних графічних ефектів на вітальній сторінці, що підвищує привабливість інтерфейсу для користувача.

Якість коду фронтенду підтримується за допомогою ESLint, який виконує статичний аналіз коду відповідно до заданих правил. Це дає змогу уникнути поширених помилок, підтримувати єдиний стиль проєкту та полегшувати процес перевірки коду [14].

Вся логіка клієнтської частини, реалізована за допомогою високорівневої мови JavaScript. Вона використовується для управління станом, обробки подій, створення запитів до сервера, динамічного оновлення контенту та зв’язування між візуальними компонентами і даними, що надходять у відповідь. Оскільки JavaScript є інтерпретованою мовою, то код виконується безпосередньо в браузері (на рівні клієнта) без потреби компіляції. Це робить її дуже гнучкою і легкою для розробки та тестування [15], [23]. У рамках проєкту JavaScript використовується для керування станом компонентів, обробки подій, обміну даними з API та маршрутизації між кроками конфігуратора. Завдяки підтримці сучасного синтаксису розробка стала зручнішою: застосовано деструктуризацію, стрілкові функції, модулі та функціональні хуки React (useState, useEffect). Це дозволяє підтримувати чисту, читабельну і масштабовану кодову базу.

Компоненти інтерфейсу розроблено з використанням JSX (JavaScript XML) – розширення синтаксису JavaScript, яке дозволяє описувати структуру елементів у вигляді HTML-подібного коду. На відміну від

класичного HTML, JSX дозволяє вбудовувати JavaScript-вирази безпосередньо всередині розмітки, що значно підвищує гнучкість і динамічність елементів інтерфейсу. Кожен компонент у цьому проєкті – це функція, яка повертає фрагмент JSX, що описує відповідний крок конфігуратора. Такий підхід спрощує структурування інтерфейсу, дозволяє створювати багаторазові блоки з власними властивостями та спрощує логіку оновлення стану.

Для обміну даними між клієнтом і сервером обрано формат JSON, що забезпечує зручну та легку передачі структурованої інформації об’єктів. Усі запити, що надсилаються до серверної частини, а також відповіді, що повертаються у відповідь, представляються у вигляді ключ-значення, де ключи є рядками, а значення можуть бути різного типу (рядки, числа, масиви, об’єкти або логічні значення), що робить API зрозумілим і універсальним [16].

3.1.2 Технології серверної частини

Серверна частина системи реалізована з використанням високорівневої мови програмування Python, яка завдяки своїй універсальності стала галузевим стандартом у сфері автоматизації, побудови API та машинного навчання. Python забезпечує високу швидкість розробки, зручний синтаксис, велику екосистему готових бібліотек та активну спільноту підтримки. Ця мова програмування є універсальною і портативною, підтримує декілька парадигм програмування що дозволяє використовувати її на різних Unix-подібних платформах, зокрема Linux, MacOS та Windows [17]. У контексті даного проєкту Python застосовується на всіх етапах логіки серверної частини: у файлі main.py визначено маршрути API та запуск сервера; у recommendation.py реалізовано логіку формування конфігурацій; а у predictor.py інтеграцію з моделлю машинного навчання. Крім того, Python використовується для моделювання схем даних

schemas.py, навчання моделі прогнозування predictor_model_vfinal.joblib, структурування бази пристроїв equipment_db.py, та взаємодії з клієнтом через HTTP-запити. Гнучкість мови дозволила швидко реалізувати як обробку складних об’єктів, так і підключення моделі штучного інтелекту без втрати читабельності коду чи продуктивності. Особливо слід відзначити, що використання Python спрощує адаптацію коду для майбутнього розширення – наприклад, при підключенні нових моделей, сервісів або баз даних.

Фреймворком для побудови програмного інтерфейсу було обрано FastAPI – сучасну технологію, що поєднує в собі простоту реалізації, асинхронність та високу продуктивність. FastAPI дозволяє швидко описувати маршрути запитів, типізувати параметри й автоматично генерувати інтерактивну документацію за стандартом OpenAPI [18]. У проєкті FastAPI використовується як основа для всієї взаємодії з клієнтською частино, саме тут реалізовано маршрути для підбору обладнання, виконання прогнозу ефективності та інтерактивної підтримки. Асинхронна обробка запитів дозволяє обробляти кілька запитів паралельно, що знижує затримки у відповідях і підвищує масштабованість системи. Додатково FastAPI забезпечує інтеграцію з зовнішніми системами моніторингу, збору даних, а також дозволяє будувати мікросервісну архітектуру у майбутньому. У таблиці 3.1 наведено переваги і недоліки FastApi у порівнянні з іншими фреймворками.

Таблиця 3.1 – Порівняння FastApi з іншими фреймворками

Фреймворк	Призначення	Переваги	Недоліки
FastAPI	REST API	Швидкість, валідація	Складність, менше рішень
Flask	Легкий API	Простота, гнучкість	Ручна реалізація

Продовження таблиці 3.1

Фреймворк	Призначення	Переваги	Недоліки
Django REST Framework	API для Django	Інтеграція, адміністрування	Важкий, синхронність
Фреймворк	Призначення	Переваги	Недоліки
Tornado	Багатопоточність, WebSocket	Продуктивність, async	Високий поріг, мало пакетів
Sanic	Асинхронні API	Швидкість, мінімалізм	Слабка підтримка

Для перевірки відповідності структури вхідних і вихідних даних використовується бібліотека Pydantic. Вона дозволяє описувати схеми у вигляді класів Python, автоматично перевіряти типи, діапазони та відповідність формату, і є ключовим інструментом для обробки запитів у FastAPI [19]. У проєкті всі запити від клієнта – конфігурація офісу, характеристики мережі, вимоги до обладнання – надходять у вигляді об’єкта ConfigRequest, який описано у модулі schemas.py. Кожне поле має заданий тип: числовий, логічний, рядковий або список. Pydantic автоматично повертає опис помилки у разі невідповідності, що спрощує налагодження API та захищає сервер від некоректних запитів. Крім цього, Pydantic застосовується також для структурування відповідей, які надсилаються назад на клієнтську частину – у тому числі конфігурацій обладнання та результати прогнозу.

Модель машинного навчання, яка використовується в системі для прогнозування ефективності конфігурації, була навчена з використанням бібліотеки Scikit-learn – одного з найбільш поширених інструментів для реалізації класичних ML-алгоритмів. Завдяки високій гнучкості, доступності стандартних класифікаторів та зрозумілій структурі моделювання, ця бібліотека дозволила швидко сформувати класифікаційну модель для аналізу технічних параметрів мережі [20]. Навчання моделі

здійснювалося на штучно сформованому наборі даних, який охоплює типові варіанти використання, з урахуванням характеристик обладнання та навантаження. У проєкті передбачено можливість заміни або перенавчання моделі без зміни загальної структури програмного інтерфейсу, що робить Scikit-learn зручною платформою для подальшого розвитку інтелектуальної складової вебзастосунку.

У таблиці 3.2 наведено приклади задач та інструменти, які містить бібліотека Scikit-learn для роботи з ними.

Таблиця 3.2 – Основні можливості Scikit-learn

Категорія задач	Приклади алгоритмів
Класифікація	Logistic Regression, SVM, Decision Tree, Random Forest, KNN
Регресія	Linear Regression, Ridge, Lasso, ElasticNet
Кластеризація	KMeans, DBSCAN, Mean Shift, Agglomerative Clustering
Зниження розмірності	PCA, Truncated SVD, t-SNE, Isomap
Ансамблеве моделювання	Random Forest, Gradient Boosting, AdaBoost, Bagging
Оцінювання моделей	Cross-validation, train_test_split, метрики: accuracy, precision, recall
Конвеєри (pipelines)	Pipeline, make_pipeline, зв’язування трансформацій та моделей
Передобробка даних	StandardScaler, MinMaxScaler, LabelEncoder, OneHotEncoder, TfidfVectorizer
Пошук гіперпараметрів	GridSearchCV, RandomizedSearchCV

Продовження таблиці 3.2

Категорія задач	Приклади алгоритмів
Обробка категоріальних змінних	OrdinalEncoder, ColumnTransformer

Для обробки, агрегації та структурування даних у серверній частині проєкту було використано бібліотеку pandas. Вона дозволяє працювати з табличними структурами (DataFrame) та зручно виконувати аналіз числових і категоріальних ознак. У рамках підготовки навчальної вибірки pandas використовувалась для генерації набору штучних конфігурацій, групування зразків за рівнем стабільності та попереднього аналізу розподілу параметрів. Завдяки високій швидкодії та великій кількості вбудованих функцій ця бібліотека стала основним інструментом на етапі попередньої обробки даних.

Для математичних операцій і роботи з масивами чисел застосовувалась бібліотека NumPy, яка є базовим інструментом для наукових обчислень у Python. Саме вона лежить в основі багатьох машинних і статистичних методів, включно зі scikit-learn. У проєкті NumPy використовувалась для перетворення векторів ознак, виконання арифметичних обчислень у процесі генерації датасету та передачі підготовлених масивів у моделі класифікації. Її інтеграція забезпечила ефективність на всіх етапах роботи з числовими даними.

Для автоматизованого підбору гіперпараметрів прогнозуючій моделі було використано бібліотеку Optuna. Вона дозволяє ефективно шукати оптимальні конфігурації параметрів за допомогою евристичних алгоритмів, зокрема Tree-structured Parzen Estimator (TPE). У проєкті Optuna допомогла обрати оптимальну глибину дерев, кількість базових моделей та функцію поділу при використанні алгоритму Random Forest. Це дало змогу покращити точність класифікації без надлишкового часу навчання, а також зробило процес оптимізації прозорим, відтворюваним і гнучким для подальших змін.

Для збереження та завантаження навченої моделі використовується бібліотека joblib, яка дозволяє забезпечити легке і ефективне відтворювання об'єктів Python, зокрема таких, що містять великі масиви NumPy-даних [21]. У рамках цього проєкту навчену модель, створену у scikit-learn, було експортовано у формат .joblib та завантажено під час ініціалізації сервера. Це дозволяє уникнути повторного навчання моделі на кожен запит і значно прискорює відповіді системи при прогнозуванні. Додатковою перевагою використання joblib є її сумісність із іншими інструментами Python, що дає змогу в майбутньому інтегрувати складніші моделі без зміни підходу до їхнього збереження.

Обмін даними між клієнтом і сервером організовано за допомогою REST API, що підтримує різні мови програмування та підтримує різноманітні формати даних, що дозволяє створити єдиний інтерфейс, ефективну розв’язку клієнт-сервер і підтримувати багаторівневу архітектуру [22]. У проєкті кожна функція (підбір, прогнозування, взаємодія з чат-ботом) винесена в окремий маршрут (/recommend, /predict, /chat). Це забезпечує логічне розділення функціоналу, простоту розширення й можливість підключення сторонніх клієнтів. REST API у поєднанні з JSON забезпечує читабельність та простоту інтеграції як з frontend, так і з потенційними зовнішніми сервісами.

3.2 Реалізація інтерфейсу користувача

Клієнтська частина у вебзастосунку реалізована у вигляді покрокового конфігуратора, який поетапно збирає всі необхідні параметри для формування рекомендацій з підбору мережевого обладнання. Загальний підхід передбачає послідовне відображення окремих компонентів React залежно від поточного стану конфігурації. Це дозволяє уникнути перевантаження екрану інформацією, спростити користувацький досвід і надати чітку логічну структуру процесу вибору.

Усі стилі реалізовано з використанням утилітарної бібліотеки Tailwind CSS, що дозволяє уникнути створення великих окремих CSS-файлів. Більшість стилів задаються безпосередньо в атрибутах className JSX-елементів, що забезпечує високу прозорість і локалізацію стилізації. У проєкті використовуються як базові утиліти Tailwind (відступи, шрифти, розміри, кольори), так і адаптивні класи для підтримки респонсивного дизайну. Крім цього, до застосунку підключено глобальні стилі у файлах index.css та App.css, де зберігаються кастомні правила, а також конфігурація Tailwind. Такий підхід забезпечує швидку верстку, зручне масштабування інтерфейсу та високу сумісність між компонентами.

Перший крок, WelcomePage, який виконує презентаційну функцію та коротко описує користувачу призначення вебзастосунку. У центрі розміщено короткий опис, та кнопку для старту. Цей крок не містить функціонального навантаження, однак виконує роль вступу до інтерфейсу. На рисунку А.1 в додатку А, можна побачити вигляд інтерфейсу першого кроку конфігуратора, WelcomePage.

Другий крок, представлений компонентом OfficeStep, користувач вводить базові параметри, що стосуються офісного приміщення. Зокрема, визначається тип простору (відкритий або закритий), кількість приміщень і поверхів. Ці параметри мають критичне значення для подальшого вибору обладнання – наприклад, для оцінки потреб у покритті Wi-Fi сигналом або кількості комутаційних пристроїв. Усі дані збираються через зручну форму з логічним поділом на секції, що дозволяє користувачу швидко орієнтуватися у вхідних параметрах. На рисунку А.2 в додатку А, можна побачити вигляд інтерфейсу дургого кроку конфігуратора, OfficeStep.

Третій крок, який реалізовано через компонент DeviceStep, користувач вказує типи та кількість пристроїв, що мають бути підключені до мережі. Серед доступних категорій: комп’ютери, IP-камери, IP-телефони, принтери, IoT-пристрої тощо. Інтерфейс компонента передбачає використання перемикачів, прапорців і випадаючих списків для зручного

введення. Зібрана інформація дозволяє системі оцінити навантаження на мережу та підібрати обладнання з відповідною пропускною здатністю та кількістю портів. На рисунках А.3 та А.4 в додатку А, можна побачити вигляд інтерфейсу третього кроку конфігуратора, DeviceStep.

Четвертий крок, InternetStep, призначений для збору параметрів інтернет-з'єднання. Користувач вказує тип підключення очікувану швидкість та характер навантаження. Ці дані використовуються не лише для вибору маршрутизатора, але й для подальшого прогнозування ефективності конфігурації – зокрема, її стійкості до перевантаження або втрати зв'язку. На рисунку А.5 в додатку А, можна побачити вигляд інтерфейсу четвертого кроку конфігуратора, InternetStep.

П'ятий крок, ConnectionStep, система уточнює інформацію про типи підключення, можливість живлення пристроїв через PoE і обирає тип системи охолодження. Користувач вводить кількість пристроїв, які потребують PoE, та кількість портів з такою підтримкою. Це дозволяє системі підібрати відповідні комутатори або маршрутизатори, які здатні одночасно передавати живлення і дані. Також користувач може обрати тип охолодження (пасивне або активне), що впливає на енергоспоживання та рівень шуму в приміщенні. На рисунку А.6 в додатку А, можна побачити вигляд інтерфейсу п'ятого кроку конфігуратора, ConnectionStep.

Шостий крок, ExtrasStep, дозволяє вказати додаткові вимоги до майбутньої мережі. Серед опцій – підтримка віддаленого доступу (наприклад, VPN), наявність вбудованого або зовнішнього firewall, а також вибір бажаного виробника обладнання. Цей крок є необов'язковим, однак він значно підвищує точність рекомендацій, оскільки враховує індивідуальні переваги користувача й специфічні сценарії використання мережі. На рисунку А.8 в додатку А, можна побачити вигляд інтерфейсу шостого кроку конфігуратора, ExtrasStep.

Сьомий крок, майже фінал, користувач переходить до компонента ResultPage, де виводиться короткий звіт за всіма введеними параметрами.

Інформація згрупована по розділах: офіс, пристрої, інтернет, підключення та додаткові налаштування. Цей екран дозволяє перевірити правильність вибору та, за потреби, повернутися до будь-якого етапу для внесення змін. Таким чином, конфігуратор надає повний цикл взаємодії – від введення до підсумкового перегляду, забезпечуючи гнучкість, контроль і прозорість усіх кроків.

Восьмий крок, RecommendationsPage, є фінальним кроком, який надає користувачу рекомендовані конфігурацію обладнання, сформованих по даним які вказав користувач на попередніх етапах. На цьому кроці користувач також отримує прогноз ефективності запропонованих конфігурацій обладнання. Цей блок містить повідомлення про те, наскільки стабільною буде мережа при вказаному навантаженні. Інформація отримується з серверної частини, де модель машинного навчання класифікує параметри за стабільністю та повертає результат.

Не менш важливим елементом є компонент ChatBot, який супроводжує на всіх кроках конфігуратора та дозволяє користувачу отримати відповіді на питання не звертаючись до зовнішніх джерел. Інтерфейс чату інтегровано у нижню праву частину екрана.

Загальне керування інтерфейсом реалізовано в основному компоненті App.jsx, який виконує роль ядра всієї клієнтської частини. Саме тут ініціалізується загальний стан конфігурації (formData) за допомогою React-хука useState, а також зберігається значення поточного кроку (step). Компонент використовує масив steps для умовного відображення відповідного етапу конфігуратора – від OfficeStep до RecommendationsPage. Усі дочірні компоненти отримують спільний об’єкт даних та функцію оновлення, що забезпечує централізоване управління даними без необхідності дублювання логіки. Запуск усього React-застосунку відбувається у файлі main.jsx, де компонент App підключається до HTML-елемента з ідентифікатором root. Для кращого контролю за поведінкою компонентів під час розробки використовується обгортка StrictMode, яка

виявляє потенційні проблеми у коді. На рисунку А.7 додатку А, можна побачити вигляд елементів навігації між кроками конфігуратора.

Взаємодія інтерфейсу користувача із сервером здійснюється через HTTP-запити до REST API, реалізованого за допомогою FastAPI. У кожному компоненті, де передбачено надсилання даних формується запит методом POST, який надсилає об’єкт конфігурації у форматі JSON. Для цього у більшості випадків використовується функція fetch, яка асинхронно звертається до маршруту (/recommend, /predict, /chat) та отримує відповідь у вигляді JSON-об’єкта. Отримані дані аналізуються та використовуються для оновлення стану або виводу результатів на екран. Такий механізм забезпечує повну відокремленість інтерфейсу від серверної логіки, дозволяючи масштабувати або оновлювати серверну частину без змін у компонентах користувача. На рисунку 3.1 наведена структура компонентів клієнтської частини.

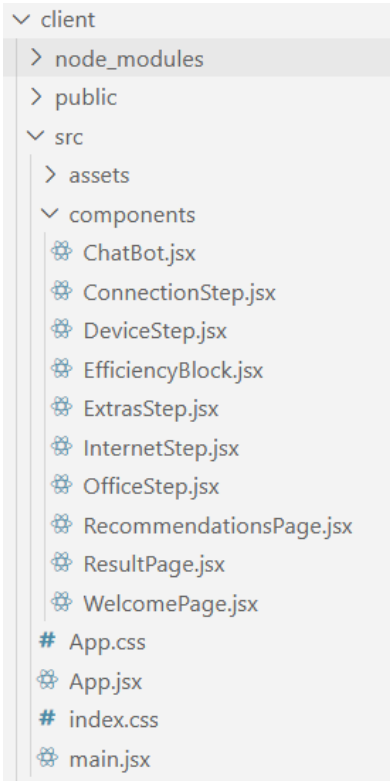

Рисунок 3.1 – Структура компонентів клієнтської частини

3.3 Реалізація логіки підбору обладнання

Логіка підбору мережевого обладнання реалізована на стороні серверної частини у модулі recommendation.py. Вона відповідає за формування комбінацій пристроїв, що найкраще відповідають потребам користувача, на основі введених параметрів. Основою є локальна база даних (БД) equipment_data, яка містить словникову структуру з описом усіх доступних пристроїв – маршрутизаторів, комутаторів та точок доступу, згрупованих за виробниками, призначенням і класами потужності (tier).

Центральна функція generate_recommendations(user_vendor) виконує побудову трьох типів конфігурацій – базової, збалансованої та професійної. Для кожного рівня система формує комплект за точно прописаними правилами із трьох компонентів – router, switch і access point. Якщо користувач вказує бажаного виробника, система підбирає конфігурації обладнання які містять виключно пристрої від цього виробника. У випадку, якщо вказано «не має значення (any)», перебираються комбінації з кількох виробників із заданим рівнем потужності.

Також для формування конфігурацій, що включають обладнання від різних виробників, було застосовано чітко визначені обмеження (rule-based), які підвищують точність підбору сумісних комбінацій. Приклад одного з таких правил – комбінація пристроїв, що містить маршрутизатор від Mikrotik, комутатор від Cisco та точку доступу Wi-Fi від Mikrotik, хоча й є технічно можливою, однак вважається небажаною. Це пов’язано з тим, що кожен виробник реалізує власну, часто несумісну або частково сумісну екосистему. Тому, використання пристроїв одного виробника забезпечує кращу інтеграцію та зручність у налаштуванні, що позитивно впливає на продуктивність та загальний досвід використання. У лістингу 3.1 наведено програмний код одного з правил підбору пристроїв, а саме від виробника Mikrotik. Повний програмний код правил підбору пристроїв, можна побачити у лістингу Б.1 додатку Б.

Лістинг 3.1 – програмний обладнання для виробника Mikrotik

```
    elif user_vendor == "MikroTik":
        result["basic"].append({
            "router": filter_device("Mikrotik", "router",
"basic"),
            "switch": filter_device("Mikrotik", "switch",
"basic"),
            "access_point": filter_device("Mikrotik",
"access_point", "basic")
        })
        result["balanced"].append({
            "router": filter_device("Mikrotik", "router",
"balanced"),
            "switch": filter_device("Mikrotik", "switch",
"balanced"),
            "access_point": filter_device("Mikrotik",
"access_point", "balanced")
        })
        result["pro"].append({
            "router": filter_device("Mikrotik", "router",
"pro"),
            "switch": filter_device("Mikrotik", "switch",
"pro"),
            "access_point": filter_device("Mikrotik",
"access_point", "pro")
        })
```

Для пошуку конкретного пристрою застосовується допоміжна функція filter_device(), яка здійснює фільтрацію за трьома критеріями: типом пристрою, виробником та рівнем потужності. Якщо відповідний варіант знайдено в базі даних, він додається до конфігурації. Для уникнення помилок, у разі невідповідності пристрої він пропускається. Програмний код функції надано у лістингу 3.2.

Лістинг 3.2 – Програмний код функції filter_device()

```
def filter_device(vendor: str, device_type: str, tier:
str) -> Dict:
    for device in equipment_data:
        if (
            device["vendor"] == vendor and
            device["type"] == device_type and
            device["tier"] == tier
        ):
            return device
    return None
```

Сформовані конфігурації структуруються у вигляді словника, де кожен ключ містить список можливих комбінацій обладнання. Кожна конфігурація є об’єктом, що містить повний опис трьох пристроїв і мета-дані для пояснення вибору. Цей результат представляється у форматі JSON і передається на клієнтську частину, де виводиться у вигляді структурованого набору рекомендацій.

Реалізована архітектура є гнучкою та масштабованою. Структура коду дозволяє легко розширити базу обладнання, додати нові типи пристроїв або вдосконалити алгоритм відбору, включивши додаткові критерії. Завдяки зрозумілій, модульній реалізації на Python, логіка підбору залишається прозорою, контрольованою та зручною для подальшої підтримки.

3.4 Навчання моделі прогнозування

Для реалізації функціональності оцінки ефективності обраної конфігурації мережевого обладнання в системі було впроваджено окрему модель машинного навчання. Вона дозволяє класифікувати параметри заданої мережі на основі вхідних ознак і передбачити рівень її стабільності за попередньо навченим алгоритмом. Такий підхід дає змогу користувачеві

отримати не лише технічну рекомендацію, але й оцінку перспектив експлуатації створеної конфігурації в реальному середовищі.

Під час розробки моделі було сформульовано задачу багатокласової класифікації: необхідно віднести конфігурацію до одного з чотирьох класів ефективності – від нестабільної до оптимальної. В основу моделі покладено дев’ять ключових ознак. Усі ці параметри подаються у числовому вигляді, що спрощує навчання і виключає потребу в додатковому кодуванні або нормалізації. У таблиці 3.3 наведено пояснення до кожної з ознак.

Таблиця 3.3 – Пояснення ключових ознак

Ознака	Пояснення
Devices_total	Загальна кількість підключених пристроїв
Bandwidth_mbps	Пропускна здатність інтернет-каналу
Wifi_flag	Наявність бездротового доступу (0/1)
Poe_flag	Використання PoE-живлення (0/1)
Vpn_flag	Потреба у VPN-каналі (0/1)
Firewall_flag	Наявність вимог до firewall (0/1)
Power_ratio	Відношення сумарного навантаження до паспортної потужності обладнання
Tier_score	Рівень продуктивності конфігурації (basic = 1, balanced = 2, pro = 3)
Brand_rank	якісна оцінка бренду (0/1)

Оскільки у відкритому доступі відсутні релевантні датасети для заданої задачі, було згенеровано власну вибірку конфігурацій. Всього створено понад 5000 зразків із випадковими, але фізично допустимими

поєднаннями ознак. Для кожного зразка клас визначено емпірично – відповідно до порогових значень навантаження, пропускної здатності та сумісності компонентів. Зібрані дані були розділені на навчальну та тестову вибірки у співвідношенні 80 до 20.

Для реалізації класифікації було обрано алгоритм Random Forest, який показав найкращі результати порівняно з логістичною регресією (Logistic Regression) та бустингом (Gradient Boosting). Конфігурацію моделі RandomForestClassifier можна побачити у лістингу 3.3.

Лістинг 3.3 – Програмний код конфігурації моделі Random Forest

```
RandomForestClassifier(
    n_estimators = 100,
    criterion    = "gini",
    max_depth    = None,
    max_features = "sqrt",
    random_state = 42,
    bootstrap    = True
)
```

Базова модель містить 100 дерев рішень і була оптимізована за допомогою крос-валідації (GridSearchCV). Найкращу узгодженість метрик показала модель RandomForestClassifier. Результати навчання наведено у таблиці 3.4.

Таблиця 3.4 – Результати навчання моделі Random Forest

Метрика	Train	Test
Accuracy	0,93	0,89
Macro-F1	0,92	0,87

Особливу цінність представляє прозорість цієї моделі: важливість ознак оцінюється автоматично, що дає змогу інтерпретувати внесок

кожного параметра. Найбільший вплив мали пропускна здатність каналу, співвідношення навантаження до потужності пристроїв та загальна кількість клієнтів у мережі. У таблиці 3.5 наведено відсоток впливу найважливіших ознак.

Таблиця 3.5 – Найважливіші ознаки за значенням

Назва ознаки	Значення важливості	Опис
Bandwidth_mbps	42 %	Пропускна здатність каналу
Power_ratio	21 %	Навантаження на обладнання
Devices_total	11 %	Кількість клієнтських пристроїв

Модель було збережено у вигляді об’єкта збереженого у форматі joblib, що дає змогу завантажувати її без необхідності повторного навчання. Навчання виконано за допомогою бібліотеки scikit-learn, версія 1.3, у середовищі Python 3.11. Збережений файл predictor_model_vfinal.joblib завантажується при запуску серверної частини й використовується безпосередньо в модулі predictor.py, який формує вектор ознак на основі запиту користувача, викликає метод predict() та обробляє результат. У лістингу 3.4 наведено код побудови пайплайна збереження моделі.

Лістинг 3.4 – Програмний код побудова пайплайна і збереження моделі

```
preproc = ColumnTransformer([
    ("num", num_transformer, num_cols),
    ("cat", cat_transformer, cat_cols)
])

model = RandomForestRegressor(
```

Продовження лістингу 3.4

```
    n_estimators=240,
    max_depth=18,
    min_samples_leaf=2,
    max_features="sqrt",
    bootstrap=True,
    random_state=42
)

pipeline = Pipeline([
    ("preprocessing", preproc),
    ("rf", model)
])

pipeline.fit(X_train, y_train)
preds = pipeline.predict(X_val)
print("MAE:", mean_absolute_error(y_val, preds))
joblib.dump(pipeline, "predictor_model_vfinal.joblib")
```

Результатом роботи моделі є числове значення рівня ефективності, яке супроводжується текстовим поясненням для користувача. У разі виявлення критичних перевищень (наприклад, коли кількість пристроїв перевищує допустиму межу для певного комутатора), результат навмисно знижується до найнижчого рівня. Це дозволяє уникнути хибного оптимізму у прогнозі та попередити можливі проблеми в експлуатації.

3.5 Реалізація прогнозуючого модуля

Для забезпечення оцінки ефективності обраної конфігурації реалізовано прогнозуючий модуль, розміщений у серверній частині в модулі predictor.py. Основна його функція полягає у визначенні рівня ефективності запропонованої конфігурації, яке складається з трьох ключових пристроїв: маршрутизатора, комутатора та точки доступу. Кожен

запит надходить у вигляді словника, що містить детальний опис як структури обладнання, так і параметрів мережі, офісного середовища та додаткових вимог.

Усередині модуля спочатку виконується завантаження попередньо натренованої моделі машинного навчання з файлу predictor_model_vfinal.joblib. Модель побудована за допомогою бібліотеки Scikit-learn і виконує класифікацію на чотири рівні стабільності: від нестабільної конфігурації до оптимальної та масштабованої. Модель приймає на вхід вектор з дев’яти ознак, які обчислюються на основі вхідних параметрів користувача.

Цей вектор включає: загальну кількість підключених пристроїв, пропускну здатність інтернету, наявність підтримки Wi-Fi, технології PoE, вимоги до VPN і firewall, потужність обраного обладнання (оцінювана як частка від максимальної межі навантаження), рівень передбачуваного навантаження (tier), а також рейтинг бренду. Для кожного з цих параметрів використовується просте числове кодування (0 або 1 для логічних параметрів, оцінка потужності у вигляді дробу тощо), що забезпечує сумісність з моделлю.

Після формування ознак модель виконує прогноз, що представляє собою ціле число від 0 до 3. Результат перетворюється у відповідну текстову мітку (label) за допомогою словника LABELS. Для більшої інформативності модуль також формує докладне пояснення, у якому вказано кількість пристроїв, пропускну здатність, потужність кожного елементу конфігурації та чи задовольняє вона потреби користувача. Усі ці дані об’єднуються у зручне для читання текстове повідомлення.

Крім оцінки моделі, модуль додатково перевіряє, чи не перевищує фактичне навантаження технічні межі пристроїв. Для цього порівнюється загальна кількість пристроїв з максимальною кількістю користувачів, які підтримує кожен елемент, ці значення вказані в DEVICE_LIMITS. У випадку перевантаження оцінка знижується до найнижчого рівня, а у

пояснення додається відповідне попередження. На лістингу 3.3 наведено програмний код DEVICE_LIMITS.

Лістинг 3.5 – Програмний код DEVICE_LIMITS

```
        router = structure.get("router", {})
        access_point = structure.get("access_point", {})
        switch = structure.get("switch", {})

        router_limit = router.get("users",
DEVICE_LIMITS["router"])
        access_point_limit = access_point.get("users",
DEVICE_LIMITS["access_point"])
        switch_ports = switch.get("ports",
DEVICE_LIMITS["switch_ports"])
```

На виході створюється JSON-об’єкт, що містить три поля: числовий рейтинг, текстову інтерпретацію та пояснення. Ці дані виводяться у блоці EfficiencyBlock клієнтського інтерфейсу, що дозволяє користувачу оцінити доцільність конфігурації і зробити правильний вибір. Завдяки цьому модуль відіграє роль експертної системи, яка мінімізує ризики недооцінки навантаження або некоректного вибору обладнання.

3.6 Реалізація інтеграції з Gemini

У межах вебзастосунку реалізовано інтерактивний модуль чат-бота, що виконує функцію консультанта під час проходження етапів конфігурації. Чат-бот інтегровано як невід’ємну частину інтерфейсу користувача та серверної логіки, з підтримкою обміну повідомленнями у реальному часі.

На клієнтському боці чат реалізовано у вигляді окремого React компонента, що відображається в правій нижній частині екрану на всіх кроках конфігуратора. Користувач має змогу вводити запитання у текстовому полі і отримувати відповіді у вигляді коротких підказок.

Повідомлення надсилаються до серверної частини за допомогою HTTP POST-запитів, сформованих через вбудований метод fetch. Формат повідомлення включає не лише сам текст, а й контекст – інформацію про поточний етап конфігуратора, яка дозволяє серверу формувати більш точну відповідь.

На серверній частині логіку обробки чат-запитів реалізовано у маршруті /chat, створеному засобами FastAPI. У відповідному обробнику chat_with_gemini() запит обробляється: до повідомлення користувача додається контекстна частина, після чого весь prompt передається функції get_gemini_response() у модулі gemini_chat.py. У цій функції реалізовано взаємодію або з зовнішнім API великої мовної моделі Gemini. Після обробки запиту результат у форматі рядка повертається назад клієнту як відповідь чат-бота.

Комунікація між клієнтом і сервером організована через стандартний формат JSON, що дозволяє зручно передавати як повідомлення, так і супутні метадані. Такий підхід забезпечує повну відокремленість інтерфейсу від логіки генерації відповідей, спрощує налагодження та дозволяє у майбутньому замінити мовну модель або інтегрувати зовнішні сервіси без зміни клієнтської частини. Крім того, реалізована структура підтримує просту масштабованість – можна додавати нові типи запитів або контекстів, пов’язаних із певними етапами конфігурації, без порушення існуючої архітектури.

У підсумку, модуль чат-бота забезпечує супровід і надає підтримку користувачу, зменшуючи потребу в технічній документації або звернення до зовнішніх джерел. Його інтеграція стала органічним доповненням до системи та покращила досвід взаємодії.

4 ОГЛЯД ІНТЕРФЕЙСУ МОДУЛІВ СУПРОВОДУ І НАДАННЯ РЕКОМЕНДАЦІЙ

4.1 Інтерфейс ResultPage

Після заповнення всіх етапів конфігуратора користувач потрапляє на підсумовуючий екран на конфігуратора ResultPage, який відображає остаточну конфігурацію, сформовану на основі введених даних. Цей компонент є підсумковим у логіці використання конфігуратора й виконує роль візуального звіту.

На сторінці виводяться результати виборів користувача в ході проходження попередніх кроків конфігуратора. Користувач отримує можливість перевірити введені значення етапом і відредагувати їх у разі потреби, що покращує досвід взаємодії з вебзастосунком і підвищує точність рекомендацій. На рисунку 4.1 наведено інтерфейс ResultPage.

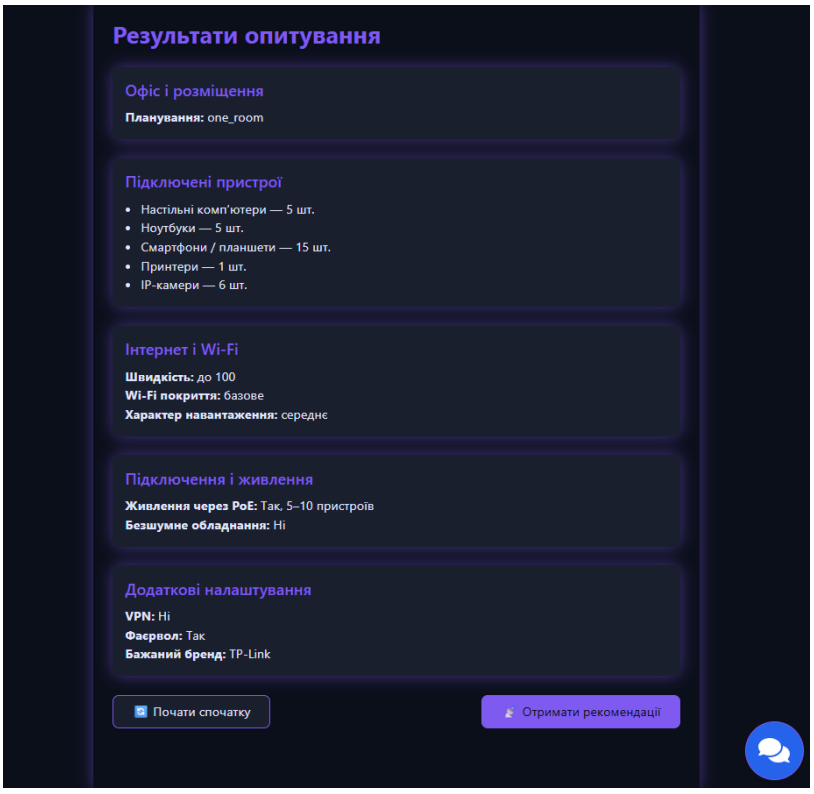

Рисунок 4.1 – інтерфейс ResultPage

Інтерфейс реалізовано у вигляді кількох блоків з логічним групуванням. Кожен елемент виводиться у стилізованій картці з використанням Tailwind CSS. У нижній частині сторінки міститься кнопка переходу до фінального етапу.

4.2 Інтерфейс DeviceStep

Другий крок конфігуратора, DeviceStep, є одним з ключових у процесі, оскільки саме тут визначається, які пристрої будуть підключені до мережі і у якій кількості. Його реалізовано як зручний і гнучкий модуль динамічного введення, що дозволяє користувачеві зосередитися на реальних потребах свого середовища без зайвого технічного навантаження.

При виборі типу пристрою із заздалегідь підготовленого списку, користувач супроводжується короткою підказкою, яка пояснює його призначення. На рисунку А.4 додатку А, можна побачити приклад роботи інтерактивних підказок.

Після вибору типу пристрою користувач вводить його кількість. Для усунення випадку введення неправильних даних, реалізовано автоматичну перевірку – користувач не може ввести від’ємне значення, усі неправильні значення автоматично нівелюються. Додати інший тип пристрої можна при натисканні на кнопку «+ Додати пристрій», також можна і видалити зайві поля, все це реалізовано безе перезавантаження сторінки. На рисунку 4.2 можна побачити поле вводу пристроїв.

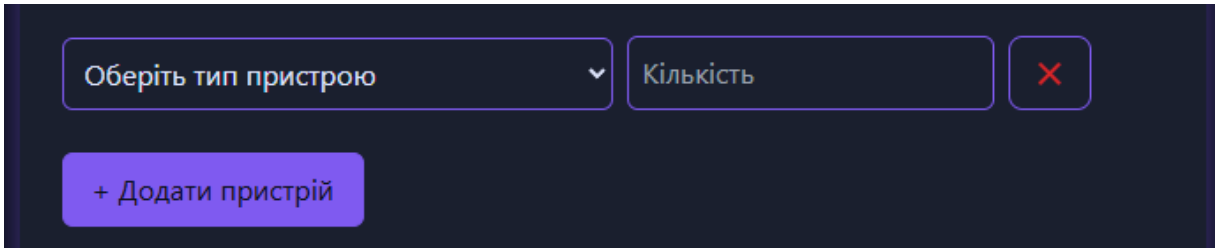

Рисунок 4.2 – Поле вводу пристроїв

Окремою перевагою цього інтерфейсу є підрахунок загальної кількості пристроїв, який автоматично оновлюється у режимі реального часу. Це не лише інформативна опція, а й важливий аналітичний елемент, що впливає на розрахунок навантаження в подальших етапах. В нижній частині екрану виводиться перелік усіх доступних типів пристроїв разом з поясненнями, що слугує одночасно як підказка, так і інструкція.

На рисунку 4.3 можна побачити елементи блоку підрахунку пристроїв і інструкції.

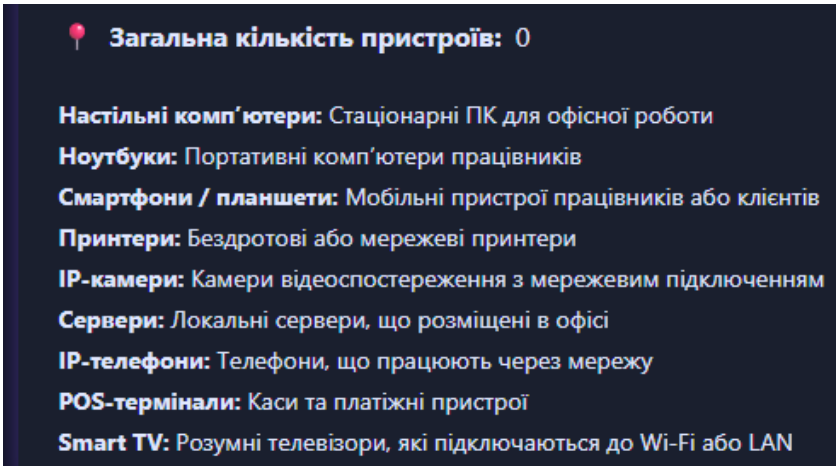

Рисунок 4.3 – Блок підрахунку пристроїв і інструкцій

4.3 Інтерфейс RecommendationsPage та EfficiencyBlock

На фінальному етапі, RecommendationsPage, користувач отримає відображення сформованих системою варіантів конфігурацій, які різняться за рівнем потужності, ціною та ступенем масштабованості. Усі рекомендовані конфігурації групуються за трьома категоріями – базова, збалансована і професійна. Кожна конфігурація представлена у вигляд окремого блоку з трьома пристроями і коротким опису кожного пристрою. Окрема увага приділена оформленню – кожен блок візуально вирізняється за допомогою анімованих рамок, тіней і кольорових заголовків.

На рисунку 4.2 наведено вигляд інтерфейсу RecommendationsPage.

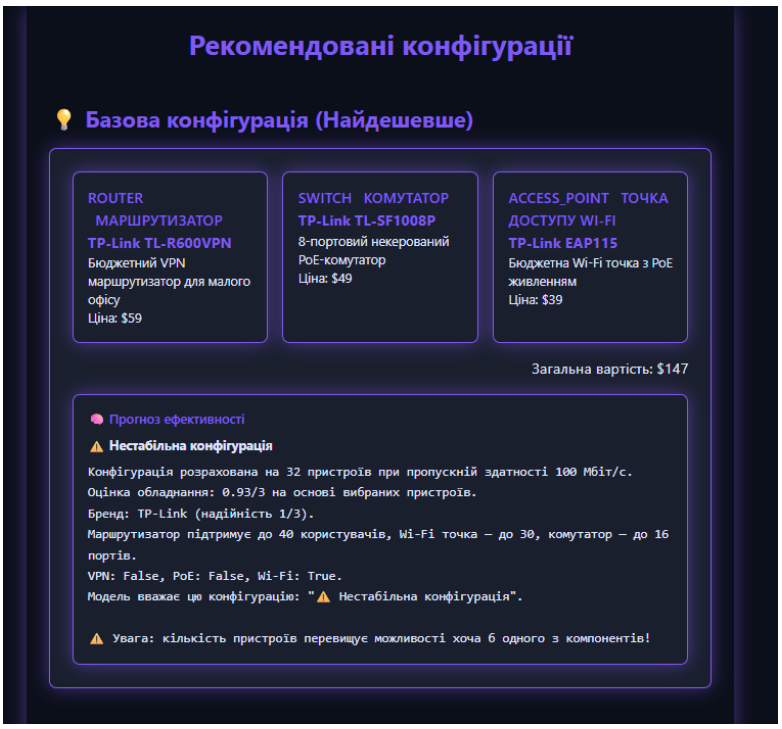

Рисунок 4.2 – Інтерфейс RecommendationsPage та EfficiencyBlock

Під кожною конфігурацією відображається компонент EfficiencyBlock, який виводить результат прогнозу ефективності. Користувач отримує коротке пояснення, сформоване на основі оцінки моделі машинного навчання: ступінь стабільності, коментар щодо перевантаження, технічні висновки. Блок прогнозу виконано у вигляді кольорового індикатора якості, який підсвічується відповідно до рівня ефективності, що продемонстровано на рисунку 4.2 під блоком запропонованої конфігурації з трьох пристроїв.

4.4 Інтерфейс ChatBot

Інтерфейс чат-бота є невід’ємною частиною вебзастонку. Він інтегрований для супроводу користувача на кожному етапі, від перших кроків до фінальних результатів. Головне завдання – допомогти, підтримати та пояснити все, що відбувається у процесі конфігурації.

Візуально чат-бот реалізований у вигляді невеликої кнопки з іконкою повідомлення, що завжди залишається на видноті в нижньому правому куті екрану. Це не заважає основній роботі з інтерфейсом і доповнює його. При натисканні відкривається невелике і зручне вікно діалогу, яке органічно вписується поверх основного інтерфейсу. Вікно має інтерфейс схожий на месенджер, що сприяє кращому сприйняттю з боку користувача.

На рисунку 4.3 наведено вигляд кнопки виклику чат-боту.

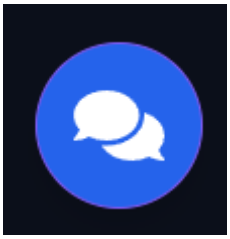

Рисунок 4.3 – Кнопка виклику чат-бота

Користувач бачить свої повідомлення праворуч, а відповіді чат-бота з’являються зліва, супроводжувані зображенням робота. Діалог виглядає природно, майже як листування в сучасному месенджері, але з легким технологічним шармом. Важливо, що бот не перевантажує текстом – його відповіді лаконічні, доброзичливі й до теми. Також чат-бот наділений здатністю бачити де саме знаходиться користувач зараз, він розуміє контекст і може надати пояснення що саме треба робити на певному кроку конфігуратора.

Одразу після відкриття вікна чат-бот першим вітається, запрошуючи поставити запитання або підказати, з чого почати. Якщо користувач вагається на певному кроці, він може звернутись до чат-бота, і той пояснить, наприклад, що таке VPN, чим відрізняється базова конфігурація від професійної. Користувач може отримати точні і швидкі відповіді на різноманітні питання в сфері телекомунікаційних технологій, завдяки інтеграції з великою мовною моделлю Gemini.

На рисунку 4.4 наведено вигляд інтерфейсу чат-бота.

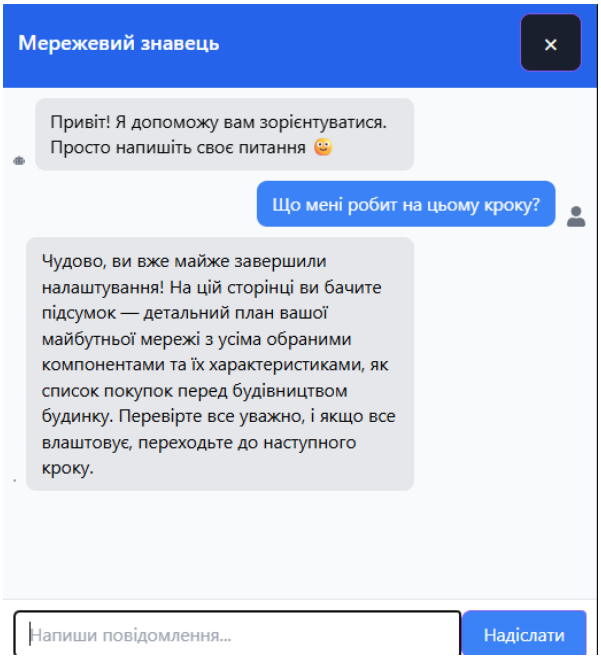

Рисунок 4.4 – Інтерфейс чат-бота

Цей інтерфейс поєднує простоту, функціональність і привітність. Він не відволікає, не тисне й не виглядає формальним. Навпаки, бот стає ідеальним цифровим консультант, що вміє слухати, розуміє ситуацію і завжди готовий допомогти.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було реалізовано повнофункціональний вебзастосунок, призначений для підбору конфігурації мережевої інфраструктури з урахуванням технічних параметрів середовища, потреб користувача та прогнозування ефективності. Проєкт охоплює повний цикл розробки – від аналізу предметної області та постановки задачі до проєктування, реалізації, тестування й документування програмної системи.

У теоретичній частині роботи було досліджено актуальні труднощі, пов’язані з вибором мережевого обладнання. Було встановлено, що користувачі часто стикаються зі складністю у врахуванні технічної сумісності компонентів, прогнозуванні навантаження та відсутністю обґрунтованих рекомендацій. Проведений аналіз ринку конфігураторів виявив, що більшість з них зосереджуються лише на фільтрації за характеристиками, не враховуючи подальшу експлуатаційну стабільність або контекст використання. Це підтвердило необхідність створення системи, що поєднує аналітичну логіку та доступний інтерфейс взаємодії.

У результаті виконання практичної частини було розроблено вебзастосунок із клієнтською частиною на базі React та серверною частиною, реалізованою за допомогою FastAPI. Інтерфейс побудовано у вигляді покрокового конфігуратора з логічною послідовністю введення даних. Для формування рекомендацій система використовує заздалегідь підготовлену базу мережевого обладнання, яка аналізується за допомогою спеціалізованих алгоритмів добору. Особливу увагу приділено реалізації модуля прогнозування ефективності, що базується на моделі машинного навчання Random Forest, навченої на синтетичному датасеті з урахуванням факторів навантаження, потужності, пропускної здатності й стабільності.

Окремим функціональним блоком впроваджено інтерактивний чат-бот, який інтегрований із великою мовною моделлю Gemini. Завдяки

контекстно-чутливому промпту та дружній поведінці та зрозумілому інтерфейсу, чат-бот забезпечує персоналізовану підтримку користувача протягом усього процесу конфігурації, пояснюючи технічні параметри та допомагаючи приймати обґрунтовані рішення.

Система демонструє приклад поєднання традиційної логіки вибору з інтелектуальними підходами – як у вигляді машинного прогнозування, так і через пояснювальний ШІ-супровід. Результати показали, що навіть за умов штучно згенерованих даних модель ефективно класифікує мережеві сценарії та знижує ризики хибного рішення користувачем.

У ході роботи було досягнуто основну мету – створити інтуїтивно зрозумілу, гнучку та аналітично обґрунтовану систему для підбору мережевого обладнання. Виконані завдання підтвердили технічну доцільність та практичну значущість такого підходу. Отриманий досвід охоплює проєктування інтерфейсів, реалізацію моделей машинного навчання, роботу з REST API, а також інтеграцію великих мовних моделей у реальне програмне середовище, що підкреслює комплексність і навчальну цінність виконаного дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Македон В. Управління розвитком мережевих структур промислових компаній в умовах цифрової економіки. *European journal of management issues*. 2024. Т. 32, № 3. С. 195–205. URL: https://www.ssoar.info/ssoar/bitstream/handle/document/97931/ssoar-ejmi-2024-3-.pdf (дата звернення: 20.05.2025).

2. Алексин В. Відмовостійка маршрутизація з підтримкою балансування навантаження. *Радіоелектроніка і молодь у XXI столітті* : Матеріали 23-го Міжнар. молодіж. форуму, м. Харків, 11 квіт. 2019 р. Харків, 2019. С. 122. URL: https://nure.ua/wp-content/uploads/workshop/konferentsiia-perspektyvy-rozvytku-infokomunikatsij-ta-informatsijno-vymiriuvalnykh-tekhnolohij-.pdf (дата звернення: 21.05.2025).

3. Castells M. Globalisation, networking, urbanisation: reflections on the spatial dynamics of the information age. *Urban studies*. 2010. Vol. 47, no. 13. P. 2737–2745. URL: https://ideas.repec.org/a/sae/urbstu/v47y2010i13p2737-2745.html (date of access: 21.05.2025).

4. Kohli R., Melville N. P. Digital innovation:areview and synthesis. *Information systems journal*. 2018. Vol. 29, no. 1. P. 200–223. URL: https://doi.org/10.1111/isj.12193 (date of access: 22.05.2025).

5. Parnell J. A. Strategic management: theory and practice. SAGE Publications, Incorporated, 2013. 664 p. URL: https://us.sagepub.com/en-us/nam/strategic-management/book238755 (date of access: 22.05.2025).

6. Datta P. M. Global technology management 4. 0: concepts and cases for managing in the 4th industrial revolution. Springer International Publishing AG, 2022. URL: https://link.springer.com/book/10.1007/978-3-030-96929-5 (date of access: 23.05.2025).

7. Nickols F. Strategy, strategic management, strategic planning and strategic thinking. *Management journal*. 2016. P. 4–7. URL: https://www.nickols.us/strategy_etc.pdf (date of access: 24.06.2025).

8. Alkhodary D. Integrating sustainability into strategic management: a path towards long-term business success. *International journal of professional business review*. 2023. Vol. 8, no. 4. P. e01627. URL: https://doi.org/10.26668/businessreview/2023.v8i4.1627 (date of access: 25.05.2025).

9. Set up your small business network - Windows Client. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/set-up-your-small-business-network (date of access: 26.05.2025).

10. Початок роботи – React. *React – JavaScript-бібліотека для створення користувацьких інтерфейсів*. URL: https://uk.legacy.reactjs.org/docs/getting-started.html#learn-react (дата звернення: 29.05.2025).

11. How to use react, the javascript framework. *Tania's Website*. URL: https://www.taniarascia.com/getting-started-with-react/ (date of access: 30.05.2025).

12. Installing with vite - installation. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/docs/installation/using-vite (date of access: 02.06.2025).

13. Getting started. *vitejs*. URL: https://vite.dev/guide/ (date of access: 03.06.2025).

14. Getting Started with ESLint - ESLint - Pluggable JavaScript Linter. *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*. URL: https://eslint.org/docs/latest/use/getting-started (date of access: 05.06.2025).

15. JavaScript. *Сучасний підручник та довідник з JavaScript*. URL: https://javascript.tutorial.in.ua/ (дата звернення: 07.06.2025).

16. Json. *Сучасний підручник та довідник з JavaScript*. URL: https://javascript.tutorial.in.ua/json/ (date of access: 08.06.2025).

17. Python Software Foundation. General python FAQ. *Python Documentation*. URL: https://docs.python.org/uk/3/faq/general.html (date of access: 10.06.2025).

18. FastAPI. *FastAPI Documentation*. URL: https://fastapi.tiangolo.com/ (date of access: 11.06.2025).

19. Welcome to pydantic. *Pydantic Documentation*. URL: https://docs.pydantic.dev/latest/ (date of access: 11.06.2025).

20. Scikit-learn: machine learning in Python. *Scikit-learn Documentation*. URL: https://scikit-learn.org/stable/index.html (date of access: 12.06.2025).

21. Joblib documentation. *Joblib: running Python functions as pipeline jobs*. URL: https://joblib.readthedocs.io/en/stable/ (date of access: 12.06.2025).

22. IBM. What Is a REST API (RESTful API). *IBM - United States*. URL: https://www.ibm.com/think/topics/rest-apis (date of access: 10.06.2025).

23. Павленко Є. П., Бутенко В. М., Головко О. В. Інженерія програмного забезпечення. WEB-програмування : навч. посіб. Харків : УкрДУЗТ, 2019. 120 с.