Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка вебсайту путівника для абітурієнтів ХНУРЕ
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТСІ-21-1

Дмитро ЛУКІЄНКО
(власне ім'я, прізвище)

Спеціальність 151 Автоматизація та комп'ютерно інтегровані технології
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Системна інженерія
(повна назва освітньої програми)

Керівник старший викладач Дмитро ГУРІН
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Лукієнко Дмитро Вікторович, як здобувачка вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

"18" червня 2025 р.  Дмитро ЛУКІЄНКО

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

Кафедра Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Системна Інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«28» квітня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Лукієнку Дмитру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту путівника для абітурієнтів ХНУРЕ затверджена наказом університету від 19.05.2025 №391 Ст.
2. Термін подання здобувачем роботи до екзаменаційної комісії 24.06.2025р
3. Вихідні дані до роботи Інформація з офіційного сайту ХНУРЕ, презентацій факультетів, а також консультаційних матеріалів для абітурієнтів; Дані про існуючі способи швидкого пошуку інформації на вебсайтах; Технології побудови сучасних вебзастосунків: Next.js, Tailwind CSS, Framer Motion; середовище розробки Visual Studio Code; мови програмування JavaScript та TypeScript.
4. Перелік питань, що потрібно опрацювати в роботі Аналіз функціональних та технічних вимог до вебсайту-путівника; Проєктування та реалізація структури вебсайту-путівника ХНУРЕ; Розробка та інтеграція чат-віджета з підтримкою embedding пошуку; Розробка адаптивного тесту для профорієнтації абітурієнтів з динамічним формуванням рекомендацій;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакати комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Графічний матеріал у вигляді презентації формату pptx, кількість сторінок 14 с.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
1	Отримання завдання до кваліфікаційної роботи	28.04.2025	виконано
2	Написання вступу	30.04-05.05.2025	виконано
3	Аналіз функціональних та технічних вимог до вебсайту-путівника	4.05-10.05.2025	виконано
4	Проєктування та реалізація структури вебсайту-путівника ХНУРЕ	11.05-18.05.2025	виконано
5	Розробка та інтеграція чат-віджета з підтримкою embedding пошуку	20.05-26.05.2025	виконано
6	Розробка адаптивного тесту для профорієнтації абітурієнтів з динамічним формуванням рекомендацій	26.05-3.06.2025	виконано
7	Оформлення пояснювальної записки	06.06.2025	виконано
8	Подавання роботи на нормоконтроль		

Дата видачі завдання 28.04.2025р.

Здобувач _____ Дмитро ЛУКІЄНКО
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ старший викладач Дмитро ГУРІН
(підпис) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка має: 56 с., 24 рис., 3 дод., 29 джерел.

ІНТЕРАКТИВНИЙ ТЕСТ, ПРОФОРІЄНТАЦІЯ, ЧАТ-БОТ, ВЕКТОРНИЙ ПОШУК, EMBEDDING, POSTGRESQL, SUPABASE, PGVECTOR, NEXT.JS, TAILWIND TYPECRIPT, ПЕРСОНАЛІЗАЦІЯ, ІНТЕЛЕКТУАЛЬНИЙ ПОШУК.

Об'єктом розробки є процес розробки та впровадження вебсайту-путівника для абітурієнтів Харківського національного університету радіоелектроніки.

Предметом розробки є функціональний вебсайт з інтегрованим адаптивним профорієнтаційним тестом для допомоги у виборі спеціальності, а також вбудованим чат-виджетом, який здійснює інтелектуальний пошук відповідей на основі embedding-векторів.

Мета роботи – розробити сучасний, зручний та адаптивний вебсайт для абітурієнтів ХНУРЕ, який включає інтелектуальний чат-бот із векторним пошуком та інтерактивний тест для підвищення якості комунікації та сприяння в обранні освітнього напрямку.

У роботі розглянуто основи веброзробки з фокусом на інтерактивність інтерфейсів, проаналізовано актуальні підходи до реалізації чат-ботів з векторним пошуком на базі Supabase. Детально описано процес розробки та інтеграції профорієнтаційного тесту, що динамічно адаптується до відповідей користувача, забезпечуючи персоналізовані рекомендації. Розроблене рішення поєднує сучасні технології та зручність використання, що підвищує ефективність інформаційної підтримки абітурієнтів університету.

Результати кваліфікаційної роботи також можна віднести до реалізації Цілі сталого розвитку 4 "Якісна освіта", зокрема п.4.4 – сприяння розвитку навичок і можливостей у молоді шляхом створення інноваційного інструменту профорієнтації та підтримки освітнього вибору.

ABSTRACT

The explanatory note: 56 pages, 24 images, 3 additions, 29 sources.

INTERACTIVE TEST, CAREER GUIDANCE, CHATBOT, VECTOR SEARCH, EMBEDDING, POSTGRESQL, SUPABASE, PGVECTOR, NEXT.JS, TAILWIND, TYPESCRIPT, PERSONALIZATION, INTELLIGENT SEARCH.

The object of the development is the process of designing and implementing a guide website for applicants to the Kharkiv National University of Radio Electronics.

The subject of the development is a functional website with an integrated adaptive career guidance test to assist in choosing a specialty, as well as a built-in chat widget that performs intelligent answer search based on embedding vectors.

The purpose of the work is to develop a modern, user-friendly, and adaptive website for applicants to NURE, which includes an intelligent chatbot with vector-based search and an interactive test to improve communication and assist in selecting an educational direction.

The work explores the fundamentals of web development with a focus on interactive interfaces, analyzes current approaches to implementing chatbots with vector search using Supabase, and provides a detailed description of the development and integration process of a career guidance test that dynamically adapts to user responses to generate personalized recommendations. The developed solution combines modern technologies and ease of use, enhancing the effectiveness of informational support for university applicants.

The results of the qualification work also contribute to the implementation of Sustainable Development Goal 4 "Quality Education", specifically item 4.4 – promoting the development of skills and opportunities among youth by creating an innovative tool for career guidance and educational decision-making support.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СУБД – Система управління базами даних

ХНУРЕ – Харківський національний університет радіоелектроніки.

ШІ – Штучний інтелект

AI – Штучний інтелект (Artificial Intelligence)

LLM – Великі мовні моделі (Large Language Models)

ML – Машинне навчання (Machine Learning)

QUIZ – Вікторина, тест

SEO – Пошукова оптимізація (Search Engine Optimization)

SSR – Серверний рендеринг (Server-Side Rendering)

SSG – Генерація статичних сайтів (Static Site Generation)

UI – Користувацький інтерфейс (User Interface)

UX – Користувацький досвід (User Experience)

ВСТУП

Сучасний світ ставить перед молоддю дедалі складніші вимоги щодо вибору професійного шляху. Одним із найвідповідальніших рішень у житті кожного випускника школи є вибір майбутньої спеціальності та закладу вищої освіти. Проте на практиці це рішення часто викликає розгубленість і сумніви. Багато абітурієнтів не мають чіткого уявлення про свої інтереси, здібності, життєві пріоритети та можливості. Їм важко оцінити власні сильні сторони, визначити галузі, до яких вони мають схильність, і спрогнозувати, де саме зможуть реалізувати себе найкраще.

В умовах стрімкого розвитку технологій, різноманіття спеціальностей та змін на ринку праці, випускники шкіл часто відчувають інформаційне перевантаження. Існуючі джерела інформації – сайти вищих навчальних закладів, офіційні довідники, тематичні форуми – здебільшого мають складну навігацію, перевантажені термінами й не адаптовані під типового користувача, який шукає швидкі та прості відповіді. Це особливо критично в умовах обмеженого часу, психологічного тиску та високої ставки помилки при виборі професії.

Одним із завдань сучасної освіти має бути не тільки надання якісної академічної підготовки, а й ефективне консультування молоді щодо вибору спеціальності. Такий вибір має ґрунтуватися не лише на оцінках чи конкурсному балі, але й на глибшому аналізі особистісних характеристик, інтересів, нахилів та потенціалу кожного окремого абітурієнта.

Зважаючи на це, особливої актуальності набуває створення інструментів, які б сприяли осмисленому вибору освітнього напряму, орієнтованому на індивідуальні особливості користувача. Серед таких інструментів доцільним є використання вебзастосунків нового покоління – зручних, інтуїтивних, інтерактивних і персоналізованих.

Мета роботи – розробити сучасний, зручний та адаптивний вебсайт для абітурієнтів ХНУРЕ, який включає інтелектуальний чат-бот із векторним

пошуком та інтерактивний тест для підвищення якості комунікації та сприяння в обранні освітнього напрямку.

Об'єктом розробки є процес розробки та впровадження вебсайту-путівника для абітурієнтів Харківського національного університету радіоелектроніки.

Предметом розробки є функціональний вебсайт з інтегрованим адаптивним профорієнтаційним тестом для допомоги у виборі спеціальності, а також вбудованим чат-виджетом, який здійснює інтелектуальний пошук відповідей на основі embedding-векторів.

Одним із основних елементів проєкту є чат-бот, що дозволяє зменшити інформаційне навантаження на користувача, надаючи відповіді у зручній діалоговій формі. Завдяки використанню embedding-пошуку, реалізованого через Supabase та PgVector, забезпечується швидкий і точний доступ до необхідної інформації без складної навігації.

Крім того, адаптивний профорієнтаційний тест побудований за принципом динамічної логіки: запитання формуються відповідно до попередніх відповідей, що дозволяє формувати персоналізовані рекомендації щодо вибору спеціальності. Такий підхід враховує особисту траєкторію користувача, його інтереси та потенціал.

Таким чином, розробка вебсайту-путівника з інтегрованими інтелектуальними інструментами є не лише доцільною, а й необхідною ініціативою для підтримки абітурієнтів у процесі прийняття одного з найважливіших рішень у їхньому житті – вибору професійного шляху.

Кваліфікаційну роботу оформлено згідно ДСТУ 3008:2015 [1], а також з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» [2].

Матеріали роботи опубліковані у [3].

1 АНАЛІЗ ФУНКЦІОНАЛЬНИХ ТА ТЕХНІЧНИХ ВИМОГ ДО ВЕБСАЙТУ-ПУТІВНИКА

1.1 Функціональні вимоги

У сучасному інформаційному середовищі абітурієнти стикаються з величезним потоком даних, що часто ускладнює прийняття усвідомленого рішення про вибір майбутньої спеціальності та закладу вищої освіти.

Пошук достовірної, актуальної та структурованої інформації вимагає від користувача значних зусиль, особливо якщо ресурс не адаптований під сучасні вимоги UX/UI.

Під час власного дослідження офіційного сайту ХНУРЕ було виявлено, що наразі відсутні інтерактивні інструменти, такі як чат-боти або системи обробки природної мови (NLP), які могли б допомогти абітурієнтам швидко знаходити потрібну інформацію в зручній формі [4] (рис. 1.1). Традиційний пошук не підтримує контекстуальне розпізнавання, а інтерфейс не адаптований під персоналізовані запити, що значно ускладнює процес орієнтації в інформації.

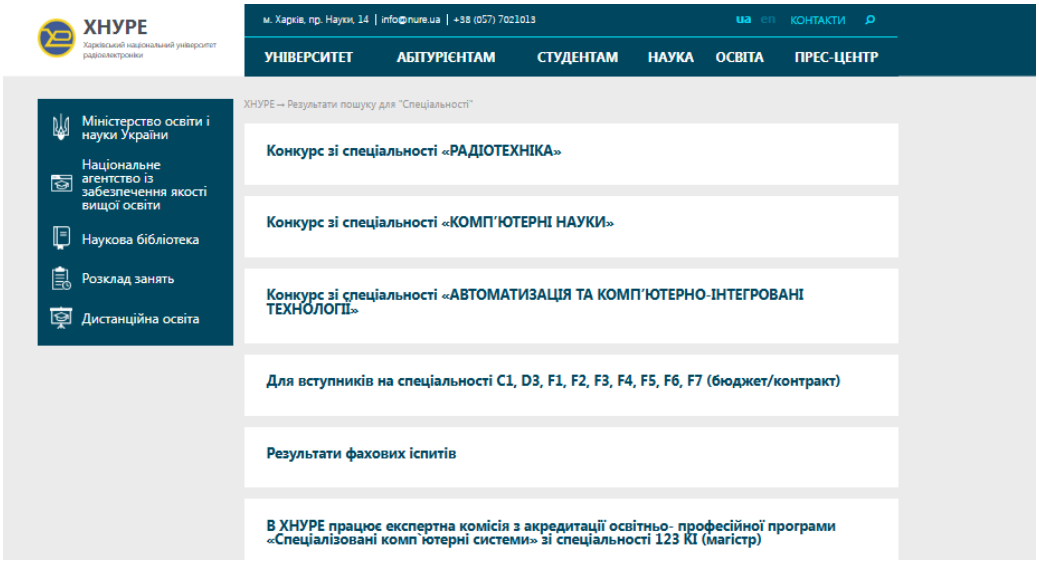


Рисунок 1.1 – Спроба пошуку за ключовим словом «Спеціальності» на вебсайті ХНУРЕ

В Україні, зокрема у ХНУРЕ, офіційні сайти традиційно орієнтовані на подання великої кількості текстового контенту, що часто не структуровано для зручного сприйняття та навігації.

Ситуація не унікальна: навіть топові університети світу – MIT, Stanford, Oxford – стикаються з викликами у створенні інтуїтивних та ефективних систем пошуку профорієнтаційної інформації [5,6]. Це підтверджує приклад спроби пошуку (рис. 1.2).

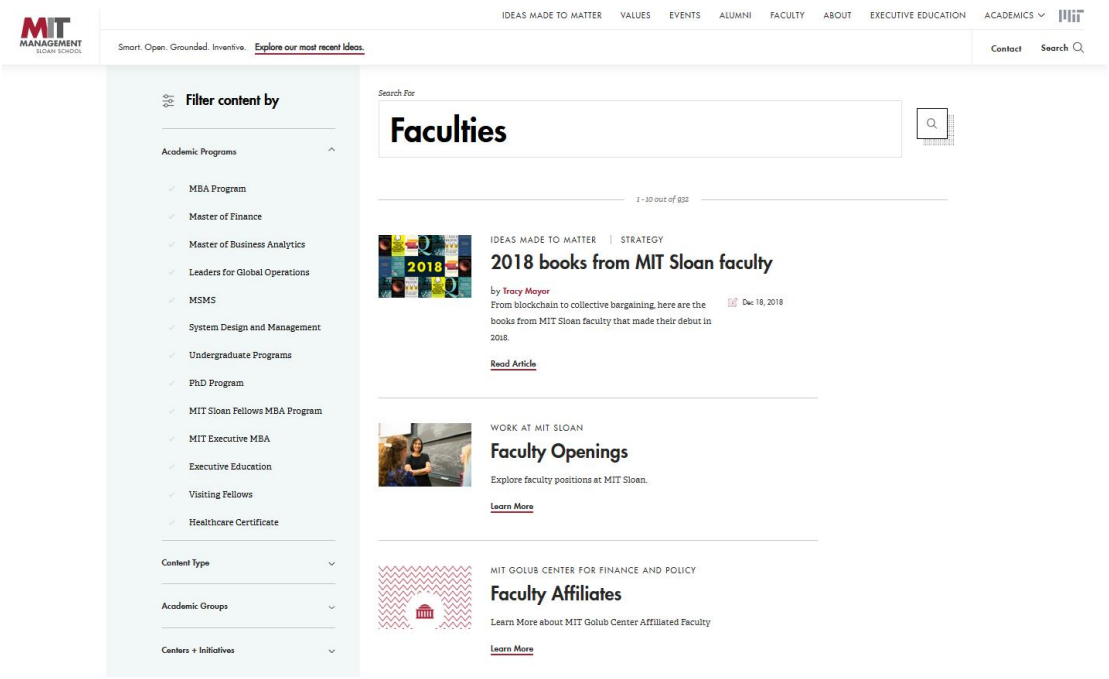


Рисунок 1.2 – Спроба пошуку за ключовим словом «Faculties» на вебсайті MIT

Часто функція пошуку обмежується простим текстовим полем, яке не завжди коректно обробляє запити, не розпізнає синоніми чи контекст, що призводить до низької релевантності результатів. Це змушує користувачів вручну переглядати розділи сайту, що значно збільшує час і зусилля, необхідні для отримання потрібної інформації.

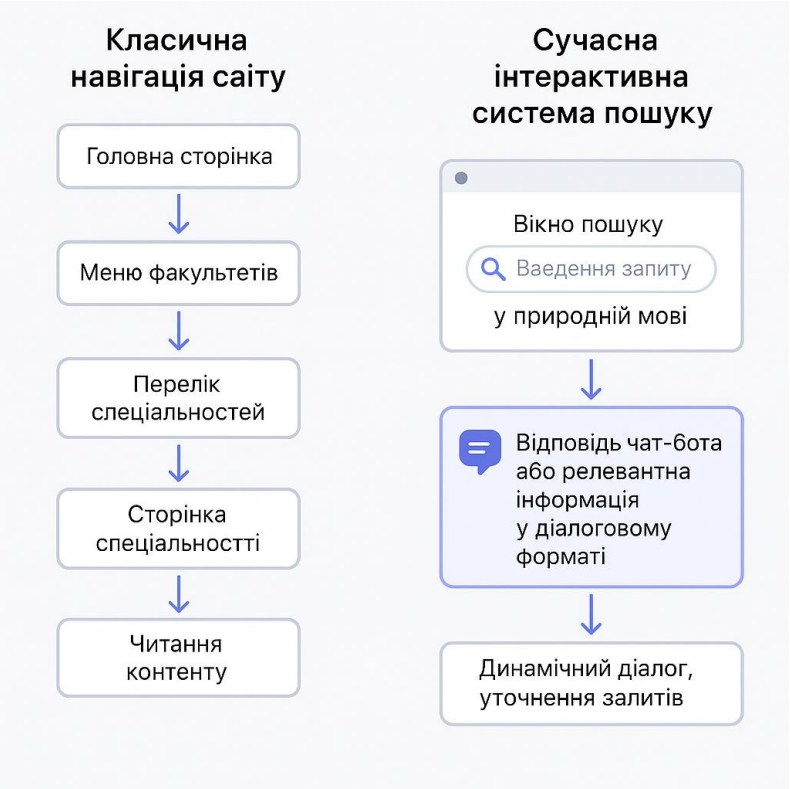


Рисунок 1.3 – Схема класичної навігації сайту проти сучасної інтерактивної системи пошуку.

Інноваційним і практичним рішенням у цьому контексті є використання чат-ботів з підтримкою embedding-векторного пошуку. Такий підхід дозволяє розпізнавати запити користувачів у природній мові, розуміти контекст та надавати відповіді у діалоговому форматі, що значно покращує взаємодію [7,8]. Для ХНУРЕ інтеграція такого чат-виджета є особливо актуальною, оскільки дозволяє суттєво знизити інформаційне навантаження на абітурієнтів та надати їм релевантні відповіді швидко і просто.

За дослідженнями UX-фахівців, інтерфейси з інтегрованими чат-ботами підвищують рівень задоволеності користувачів, скорочують час пошуку інформації на 30-50% та покращують загальне враження від ресурсу [9,10]. Особливо це стосується молоді, яка звикла до комунікації в месенджерах і цінує зручність та швидкість.

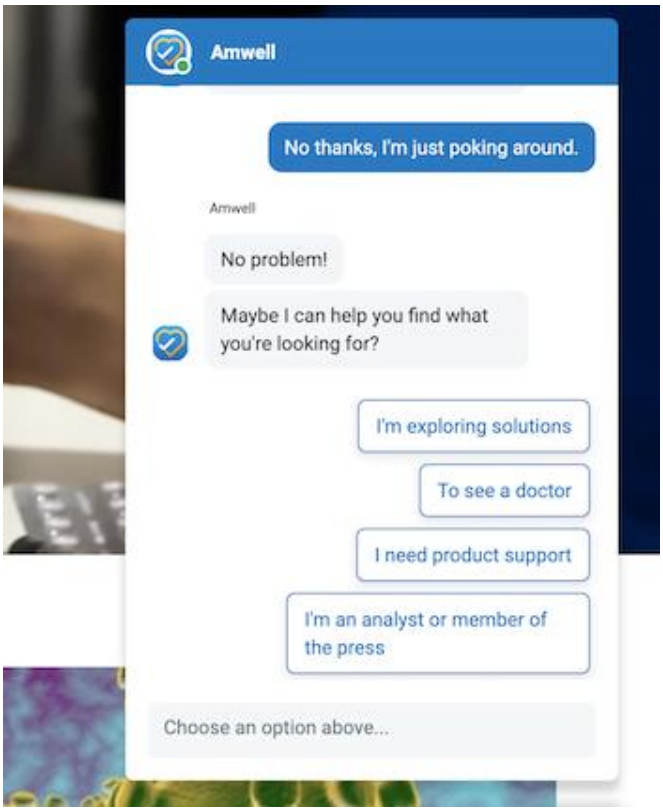


Рисунок 1.4 – Приклад діалогового вікна чат-бота з релевантними відповідями.

Додатково важливим компонентом є реалізація адаптивного профорієнтаційного тесту. Сучасні підходи до профорієнтації визнають, що ефективність рекомендацій залежить від урахування індивідуальних особливостей особистості, інтересів і схильностей. Традиційні тести часто мають статичну структуру, що обмежує їх точність і актуальність [11].

Адаптивний тест, розроблений у межах цього проєкту, базується на динамічній логіці формування питань: відповіді користувача впливають на подальші запитання, що забезпечує персоналізований підхід і підвищує якість рекомендацій [12]. Така методика активно застосовується у провідних світових платформах профорієнтації і навчання, підтверджена багатьма науковими дослідженнями [13].

Крім того, для розширення аудиторії та забезпечення доступності, вебсайт має підтримувати мультимовність, насамперед українську та англійську мови.

Це відповідає світовим тенденціям інклюзивності та глобалізації освіти, що дозволить залучити як українських, так і іноземних вступників [14].

Для забезпечення актуальності та оперативності контенту передбачено створення зручної адміністративної панелі. Вона дає змогу співробітникам університету без глибоких технічних знань швидко оновлювати інформацію, редагувати базу знань чат-бота та керувати профорієнтаційним тестом [15]. Такий підхід зменшує залежність від розробників та пришвидшує впровадження змін.

Відповідно до сучасних стандартів веброзробки, інтерфейс повинен бути адаптивним, забезпечуючи коректне відображення та зручність роботи на різних пристроях – від десктопів до мобільних телефонів. Це особливо важливо з огляду на те, що значна частина молоді користується мобільним інтернетом для пошуку інформації [16].

1.2 Технічні вимоги

Розробка вебсайту-путівника для абітурієнтів ХНУРЕ передбачає дотримання низки технічних вимог, які забезпечать стабільну роботу системи, її масштабованість, зручність обслуговування та інтеграцію з іншими ресурсами.

Сайт має бути реалізований із використанням сучасного стеку технологій, зокрема Next.js як основного фреймворку. Next.js забезпечує високу продуктивність, рендеринг на сервері (SSR) та генерацію статичних сторінок (SSG), що критично важливо для освітніх платформ, які мають бути швидкими, доступними та добре індексованими пошуковими системами. Завдяки Next.js розробники мають можливість повністю контролювати метадані, Open Graph-інформацію, структуру sitemap і robots.txt, що безпосередньо впливає на SEO-оптимізацію, зручність пошуку сайту через Google та інші платформи.

Необхідна підтримка адаптивної верстки, що забезпечить коректне відображення сайту на пристроях з різними розмірами екрана: від смартфонів до

великих моніторів. Інтерфейс має залишатися зручним і функціональним незалежно від типу пристрою.

Для забезпечення широкої доступності ресурсу потрібно реалізувати мультимовну підтримку (мінімум українська та англійська мови), з можливістю розширення мовного пакета в майбутньому. Всі текстові матеріали мають бути відокремлені від коду для зручного перекладу через систему керування локалізаціями.

Ключовим елементом є інтеграція чат-бота, що має бути реалізована у вигляді окремого віджета з можливістю вбудовування в інші сайти та системи за допомогою iFrame. Це дозволить ХНУРЕ або іншим партнерам розміщувати консультативний функціонал на зовнішніх платформах – наприклад, на головному сайті університету, в особистому кабінеті абітурієнта або на партнерських порталах.

Важливо також передбачити адміністративну панель, яка дозволить уповноваженим співробітникам зручно редагувати інформацію на сайті та в базі знань чат-бота. Через неї має бути можливо додавати або змінювати спеціальності, опис факультетів, відповіді на типові запитання, а також налаштовувати логіку адаптивного тесту. Панель повинна мати простий і безпечний інтерфейс із розмежуванням прав доступу.

Система має підтримувати семантичний пошук із використанням технологій обробки природної мови (Natural Language Processing), що дозволить користувачу вводити запити в довільній формі (наприклад: "я хочу стати айтішником" або "я добре знаю математику – куди вступати?") та отримувати релевантні відповіді.

Уся інформація на сайті має зберігатися у централізованій базі даних, з можливістю регулярного резервного копіювання та відновлення. Технології, що використовуються, повинні підтримувати хмарну інфраструктуру, щоб забезпечити масштабованість, високу доступність та безпеку.

Для забезпечення конфіденційності та захисту персональних даних користувачів слід реалізувати авторизацію та автентифікацію (для

адміністративної частини), SSL-шифрування та базові механізми захисту від ботів і DDoS-атак.

Загалом, реалізація всіх технічних вимог має забезпечити надійність, гнучкість та готовність платформи до подальшого розвитку, розширення функціоналу й інтеграції з іншими освітніми або консультаційними сервісами.

1.3 Вимоги до чат-віждета

Чат-віджет є одним із найважливіших та ключових елементів вебсайту-путівника для абітурієнтів. Він має забезпечувати максимально зручну, інтерактивну та персоналізовану консультацію, яка допоможе молоді швидко і легко отримувати відповіді на запитання, що виникають у процесі вибору майбутньої спеціальності та навчального закладу. Для досягнення цієї мети віджет повинен відповідати низці важливих технічних та функціональних вимог.

Передусім, чат-віджет має бути адаптивним – коректно відображатися на різних пристроях: від великих екранів комп’ютерів до мобільних телефонів. Він повинен розташовуватися збоку екрану таким чином, щоб не заважати основному контенту сайту, але водночас бути завжди помітним і доступним для користувача. Для підвищення юзер-френдлі досвіду віджет повинен підтримувати можливість згортання і розгортання за допомогою кнопки, що є звичною функцією для більшості сучасних чат-ботів. Такий підхід дозволяє зручно керувати видимістю віджета і не відволікатися від основної інформації.

Крім того, важливо, щоб віджет супроводжувався плавними, приємними анімаціями – наприклад, віджет повинен з’являтися одразу після відкриття сайту, щоб привернути увагу користувача і повідомити про наявність інтерактивної допомоги. Анімації створюють більш сучасний та привабливий інтерфейс, покращують взаємодію і враження від користування.

Однією з ключових особливостей є те, що чат-віджет має зберігати повну історію переписки користувача навіть після його закриття або переходу між різними сторінками сайту. Це надзвичайно важливо, адже дозволяє користувачу

легко повертатися до раніше отриманої інформації, не втрачати контекст діалогу і не повторювати питання. Для реалізації цього можуть використовуватися такі технології, як локальне сховище браузера (localStorage) або інші методи збереження даних на стороні клієнта.

Усі відповіді, які надає чат-бот, мають містити посилання на первоісточник інформації – наприклад, на офіційний сайт університету, розділ з описом факультету чи спеціальності. Це дає користувачу можливість, за потреби, глибше вивчити цікаву тему, отримати розширену або офіційну інформацію безпосередньо зі справжнього джерела.

Віджет повинен підтримувати контекстне розуміння діалогу. Якщо користувач у розмові зазначив певну спеціальність або напрямок, то всі наступні питання і відповіді, навіть без повторного уточнення, мають автоматично бути прив’язані саме до цієї теми. Для цього застосовуються сучасні методи обробки природної мови (Natural Language Processing) і відповідні ембеддинги (векторні уявлення), які дозволяють системі правильно і точно визначати тематику і контекст запитань, підбираючи релевантні відповіді.

Крім основної консультації, віджет повинен мати функціонал для пропозиції пройти адаптивний профорієнтаційний тест. Наприклад, коли користувач запитує поради щодо вибору напрямку навчання, система має надати кнопку або посилання на тест. При натисканні на цю кнопку віджет плавно трансформується в інтерактивний особистий інтерфейс, призначений для проходження тестування.

Тест побудований як динамічна послідовність питань, де кожне наступне питання залежить від відповідей користувача на попередні. Якщо абітурієнт уже визначився з основним напрямком, наприклад, програмуванням, то всі питання будуть зосереджені на уточненні, яке саме піднапрямок чи спеціалізація йому цікавіша. Такий підхід дозволяє максимально індивідуалізувати результати і зробити рекомендації більш точними та корисними.

Після завершення тестування користувачу буде надано детальний підсумок з рекомендаціями щодо спеціальностей, які найбільше відповідають

його інтересам, здібностям та навичкам. Це допоможе більш усвідомлено та впевнено обирати подальший освітній шлях.

Для покращення взаємодії у віджеті обов’язково має бути реалізований лоудер (індикатор завантаження), який інформуватиме користувача про обробку запиту або очікування відповіді від системи. Це зробить інтерфейс більш зручним і зрозумілим, знизить рівень фрустрації в користувача.

Отже, чат-віджет повинен бути сучасним, гнучким, інтерактивним та максимально орієнтованим на індивідуальні потреби кожного абітурієнта. Він має забезпечувати швидку, точну і персоналізовану допомогу у виборі спеціальності, роблячи процес прийняття рішення більш комфортним і ефективним.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СТРУКТУРИ ВЕБСАЙТУ-ПУТІВНИКА ХНУРЕ

2.1 Ініціалізація вебзастосунку

На етапі початкового проєктування було обрано сучасний фреймворк Next.js, який є одним із найпопулярніших засобів для створення масштабованих, швидких та SEO-оптимізованих вебзастосунків. Його функціональність включає серверний рендеринг, підтримку статичної генерації, вбудовану маршрутизацію та гнучку архітектуру. Це робить його ідеальним вибором для реалізації вебсайту-путівника для абітурієнтів, де важливу роль відіграє швидкий доступ до інформації, локалізація та динамічний інтерфейс.

Ініціалізація застосунку була здійснена за допомогою офіційної CLI-команди: npx create-next-app@latest

При створенні проєкту було одразу обрано використання App Router – нової маршрутизаторної моделі в Next.js, яка базується на файловій системі й дозволяє будувати інтерфейс із гнучкою композицією layout-ів, segment-ів, nested route-ів, шаблонів тощо. Завдяки App Router стало можливим організовувати логіку застосунку за допомогою вкладених директорій, таких як @layout, @page, @template, @error та інших, що значно спрощує підтримку великого проєкту й забезпечує масштабованість [17] (рис. 2.1).

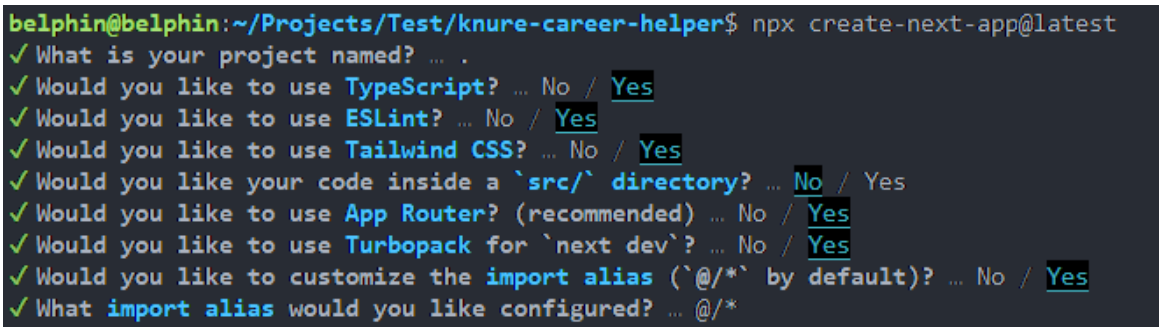


Рисунок 2.1 – Налаштування стартового темплейту

Під час генерації застосунку було автоматично встановлено низку корисних інструментів:

– PostCSS – для обробки Tailwind CSS та інших CSS-плагінів;

– Tailwind CSS – сучасна утилітарна CSS-бібліотека, яка значно прискорює процес створення інтерфейсів завдяки використанню готових класів без потреби писати кастомні стилі з нуля;

– ESLint – для автоматичної перевірки синтаксису, дотримання стилістичних правил коду та уникнення помилок під час розробки;

– TypeScript – для забезпечення статичної типізації, що особливо важливо в проєктах середньої та великої складності.

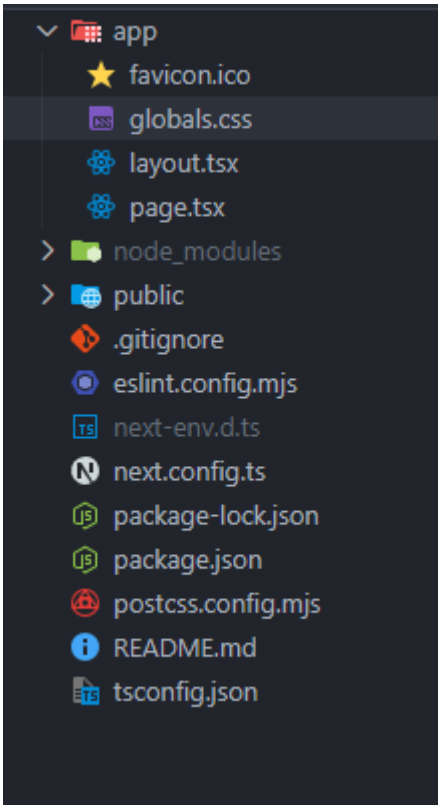


Рисунок 2.2 – Стартовий темплейт Next.js

У якості пакетного менеджера було свідомо обрано pnpm. Це сучасна альтернатива npm і yarn, яка виділяється високою швидкістю роботи, ефективним кешуванням пакетів і можливістю спільного використання залежностей між різними проєктами. Його використання дозволяє значно зменшити загальний розмір директорії node_modules, уникнути дублювання

пакетів, а також пришвидшити встановлення та оновлення залежностей, що позитивно впливає на швидкість збірки і CI/CD-процесів [17].

Таким чином, ще на етапі ініціалізації було закладено фундамент сучасного, продуктивного та зручного в розробці вебзастосунку, що відповідає сучасним стандартам frontend-розробки.

2.2 Налаштування SCSS і Tailwind CSS

Для досягнення високої якості оформлення інтерфейсу вебсайту-путівника було прийнято рішення поєднати переваги двох потужних технологій: Tailwind CSS та SCSS. Tailwind CSS – це сучасна утилітарна CSS-бібліотека, яка надає готові класи для швидкого створення адаптивних і стильних інтерфейсів без необхідності писати власні стилі з нуля. Однак у великих проєктах часто виникає потреба в більш гнучкому повторному використанні стилів, групуванні класів і створенні змінних, що полегшує підтримку та масштабування коду.

Для цього до проєкту було додано підтримку SCSS – CSS-препроцесора, який надає розширені можливості, такі як змінні, вкладеність, міксіни, імпорти і директиви. У поєднанні з Tailwind CSS, SCSS дозволяє систематизувати стилі, уникати дублювань і створювати читабельний та підтримуваний код. Встановлення SCSS здійснювалось за допомогою команди: pnpm add sass

Щоб організувати стилі, було створено структуру файлів у папці styles/, яка включає:

– colors.scss – файл, де визначені кольорові змінні для різних тем, зокрема світлої та темної. Це дозволяє швидко змінювати колірну палітру проєкту централізовано, не змінюючи численні CSS-класи по всьому коду;

– components.scss – файл, що містить стилі для окремих компонентів, створені за допомогою директиви @apply. Ця директива Tailwind дозволяє об’єднувати кілька утилітарних класів у один клас SCSS, що спрощує повторне використання і покращує організацію стилів;

– global.scss – глобальні стилі, які застосовуються на рівні всього застосунку, наприклад, для базових тегів, скидання стилів, типографіки та інших загальних налаштувань;

– theme.scss – спеціальний файл, який відповідає за зв’язок кольорових змінних SCSS з кольорами Tailwind. Це дозволяє інтегрувати SCSS-перемінні у конфігурацію теми Tailwind та забезпечує гнучке керування темізацією проєкту.

Використання директиви @apply у SCSS значно підвищило ефективність коду, оскільки дозволяє логічно об’єднувати групи класів Tailwind у повторно використовувані класи, зменшуючи дублювання та покращуючи читабельність стилів. Це особливо корисно в проєктах зі складною структурою компонентів, де однакові набори класів часто застосовуються в різних місцях [18].

Варто також звернути увагу на важливі зміни в останніх версіях Tailwind CSS, що стосуються темізації. Раніше кольорові схеми (світла, темна, та інші) визначалися переважно у конфігураційному файлі tailwind.config.js, що накладало певні обмеження на динамічне перемикання тем. У нових версіях Tailwind підтримується призначення тем безпосередньо у DOM за допомогою класів, наприклад, class="dark", які можна динамічно додавати чи видаляти на рівні HTML-елементів. Такий підхід значно підвищує гнучкість керування темами в реальному часі, полегшує реалізацію адаптивних інтерфейсів та дає змогу швидко реагувати на зміни налаштувань користувача без необхідності перезавантаження чи зміни конфігурації проєкту [19].

Завдяки такому поєднанню SCSS і Tailwind CSS, а також новому підходу до темізації, проєкт вебсайту-путівника отримав потужний та зручний інструментарій для підтримки сучасного дизайну з урахуванням різних уподобань користувачів.

2.3 Підключення мультимовності з next-intl

З метою підтримки інтерфейсу для українських та іноземних абітурієнтів у проєкт було додано мультимовність. Для цього інтегровано бібліотеку next-intl (версія 4.x), яка дозволяє реалізувати локалізацію з використанням URL-префіксів (/ua, /en тощо).

Встановлення здійснювалося відповідно до офіційного гайду [20]:

```
pnpm add next-intl
```

У рамках реалізації підтримки мультимовності було прийнято рішення використовувати префікси мов у структурі URL. Наприклад, для української мови використовується /ua, а для англійської – /en. Попри те, що загальноприйняте скорочення для української – uk, у цьому проєкті було свідомо обрано ua, оскільки саме ця форма більш звична й зрозуміла для цільової аудиторії сайту, зокрема абітурієнтів та їхніх батьків.

Для організації мовної маршрутизації було створено динамічну директорію [locale], яка відповідає за вибір мови інтерфейсу на основі URL. У Layout-компоненті реалізована перевірка, чи є обрана мова серед підтримуваних, та відображення вмісту відповідною мовою. У разі недійсної локалі користувача перенаправляє на сторінку помилки (рис. 2.3).

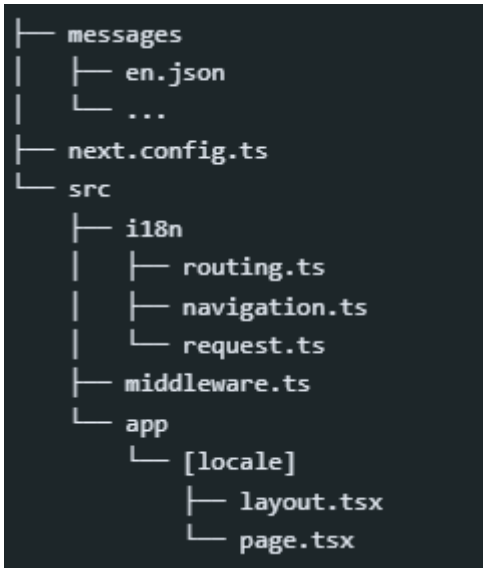


Рисунок 2.3 – Архітектура підключення бібліотеки next-intl

Такий підхід дозволяє зручно керувати мовною логікою на рівні маршрутизації та чітко визначати сторінки, які мають підтримувати локалізацію. Завдяки використанню окремої динамічної директорії можна легко масштабувати систему перекладів та уникати дублювання коду для кожної мовної версії [20].

Також для централізованого керування мовами, навігацією та текстами за допомогою гайду було створено окрему директорію i18n, яка містить кілька допоміжних файлів [21]:

– routing.ts – визначає доступні локалі, базові маршрути та логіку для перемикання мови;

– navigation.ts – містить конфігурацію навігаційних пунктів з урахуванням локалей;

– requests.ts – описує шаблони запитів, які залежать від мови або використовуються в багатомовному середовищі.

Цей підхід дозволив забезпечити зручну масштабовану структуру для додавання нових мов у майбутньому та забезпечити стабільну роботу сайту для різних категорій користувачів.

2.4 Інтеграція Supabase у проєкт та організація захищених сторінок

Для забезпечення надійної системи аутентифікації користувачів, а також організації захищених зон вебсайту-путівника було обрано сервіс Supabase. Ця платформа дозволяє легко реалізувати автентифікацію, керування сесіями, а також взаємодію з базою даних за допомогою сучасного API і SDK, що робить її ідеальним вибором для проєктів на базі Next.js.

У проєкт було інтегровано Supabase, використовуючи публічний анонімний ключ (anon key) для ініціалізації клієнта. Цей ключ дозволяє здійснювати безпечні запити до бази даних і серверних функцій без необхідності додаткової складної конфігурації, одночасно забезпечуючи захист через правила

доступу на стороні сервера. Завдяки цьому у майбутньому можна буде легко реалізувати такі функції, як пошук релевантних ембеддингів на основі запиту користувача у чат-виджеті [22].

Однією з головних причин вибору Supabase є її підтримка зберігання векторних представлень (ембеддингів) і пошуку за векторними відстанями безпосередньо у базі даних. Це дозволяє реалізувати високопродуктивний пошук найбільш релевантних ембеддингів без необхідності завантажувати всі дані у пам’ять для подальшого аналізу. Такий підхід значно оптимізує роботу пошукової системи чат-бота, роблячи її швидшою та ефективнішою [22].

Архітектурно проект організовано так, що всі захищені сторінки розміщені у папці protected і обгорнуті у спеціальний layout-компонент – ProtectedLayout. Цей компонент перевіряє наявність авторизації користувача, і у разі відсутності авторизації перенаправляє його на сторінку входу. Таким чином, реалізується базовий захист контенту, що доступний лише авторизованим користувачам. Цей підхід значно спрощує підтримку безпеки та дозволяє централізовано керувати доступом до приватних частин вебсайту [23].

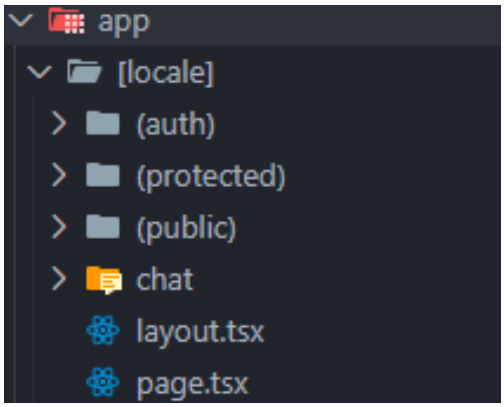


Рисунок 2.4 – Архітектура розділення сторінок на public та protected

Сторінка входу Login.tsx реалізована як окремий маршрут у системі App Router Next.js. Вона відповідає за прийом даних користувача, виклик методів аутентифікації Supabase, а також обробку помилок і повідомлень. При успішній аутентифікації користувач автоматично перенаправляється до дашборду –

головної захищеної сторінки. У випадку, якщо користувач вже авторизований і намагається зайти на сторінку входу, він буде автоматично перенаправлений назад на дашборд, що покращує UX.

Важливо зазначити, що сесії та інформація про авторизацію зберігаються у cookie браузера, що дає змогу використовувати їх як на клієнті, так і під час серверного рендерингу (SSR). Це дозволяє реалізувати такі функції, як попередня перевірка сесії на сервері та динамічне формування контенту без зайвих запитів після завантаження сторінки, що підвищує безпеку [23].

Адміністративна панель у проєкті не виділена в окремий репозиторій або окремий додаток, а інтегрована безпосередньо в основний код вебсайту-путівника. Таке рішення було прийнято свідомо для спрощення демонстрації функціоналу та зручності розробки. Інтеграція адмінпанелі у той же проєкт дозволяє зменшити складність розгортання, усунути необхідність в окремих точках аутентифікації та уникнути дублювання коду. Крім того, це спрощує підтримку та розвиток функціоналу, адже всі частини системи перебувають у єдиній кодовій базі.

Серед ключових переваг використання Supabase в цьому проєкті варто виділити:

– проста інтеграція з Next.js та React, що дозволяє швидко реалізувати аутентифікацію та захищені маршрути без необхідності складної серверної конфігурації;

– підтримка SSR через збереження сесій у cookie, що дозволяє робити серверний рендеринг із врахуванням стану користувача;

– безпека: завдяки політикам доступу Supabase (RLS) і використанню анонімного ключа, запити до бази даних виконуються лише з дозволом, що підвищує безпеку;

– можливість зберігання та пошуку векторних ембеддингів безпосередньо у базі даних, що оптимізує роботу пошукової системи чат-бота та дозволяє ефективно знаходити найбільш релевантні відповіді;

– масштабованість: легко можна розширити функціонал, додаючи нові API або серверні функції без зміни архітектури;

– уніфікована кодова база завдяки інтеграції адмінпанелі у той же проєкт, що і основний сайт, що полегшує розробку та демонстрацію.

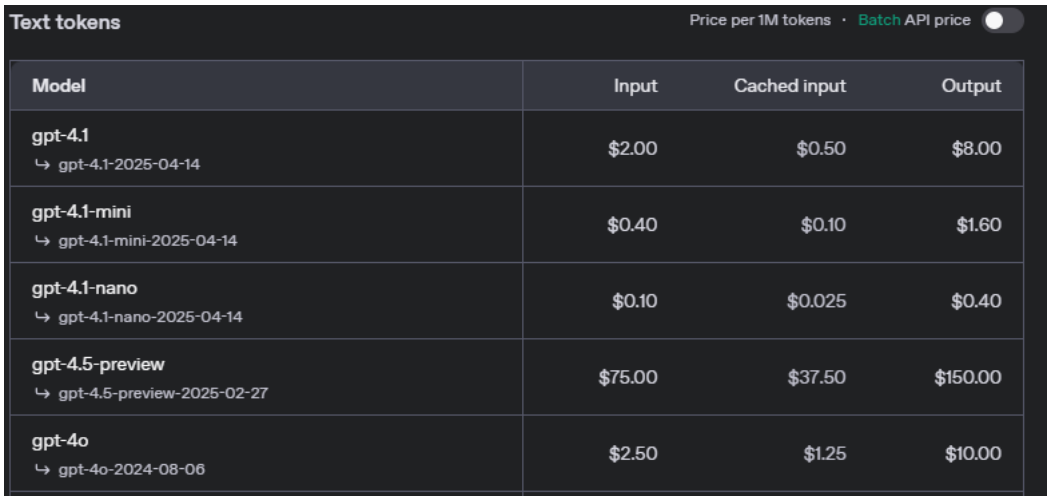
Таким чином, обраний підхід інтеграції Supabase забезпечив не лише зручний і безпечний механізм аутентифікації, а й створив гнучку платформу для подальшого розвитку функціоналу вебсайту-путівника ХНУРЕ, включно з розумним пошуком інформації за допомогою векторних ембеддингів.

3 РОЗРОБКА ТА ІНТЕГРАЦІЯ ЧАТ-ВІДЖЕТА З ПІДТРИМКОЮ EMBEDDING ПОШУКУ

3.1 Загальна ідея чат-виджета

Чат-виджет є ключовим інструментом усього вебсайту-путівника для абітурієнтів ХНУРЕ. Його основна мета полягає у забезпеченні інтерактивної, швидкої й максимально точної консультації в діалоговому форматі, який є звичним та зручним для сучасної молоді. Через цей інтерфейс користувач найчастіше вперше взаємодіє з цифровим середовищем університету, тому від його якості залежить загальне враження про сервіс. Важливими є швидкість, релевантність відповідей і простота взаємодії, особливо в умовах великого обсягу інформації та психологічного навантаження у період вступної кампанії.

Початково може здатися логічним використання великих мовних моделей (LLM), таких як GPT-4 або Claude, які здатні генерувати довгі та змістовні відповіді на природні запити користувача. Проте в реальному застосуванні такий підхід виявляється неефективним. По-перше, звернення до LLM потребує формування складного системного промпту – інструкції, яка задає модель поведінки, стилю, базових знань. Вона займає кілька сотень токенів і витрачає ресурси незалежно від складності самого питання. Крім того, до цього промпту додається сам запит користувача та очікувана відповідь. У підсумку одне звернення до GPT-4 може складати понад 700–900 токенів, вартість яких сягає приблизно 0.01 долара США. За умовами масового використання (наприклад, 1000 запитів на день) це обертається щомісячними витратами у 300 доларів навіть без пікових навантажень. Для освітнього некомерційного проєкту подібна модель є фінансово невиправданою. Ціни на найновіші LLM моделі OpenAI зображено на рис. 3.1.



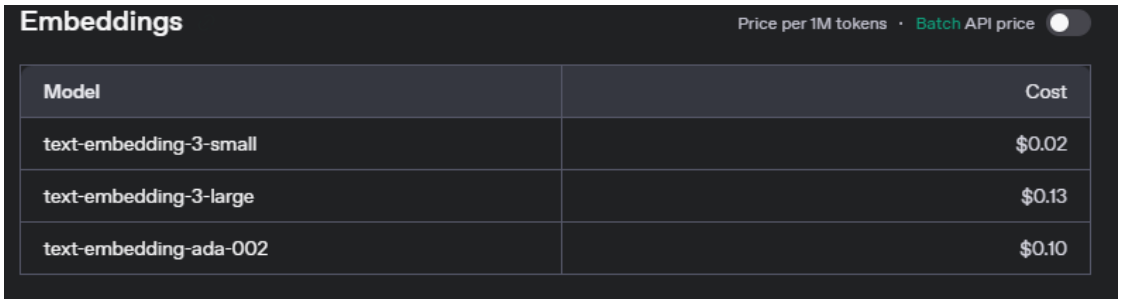
Text tokens — Price per 1M tokens · Batch API price

Model	Input	Cached input	Output
gpt-4.1 ↳ gpt-4.1-2025-04-14	$2.00	$0.50	$8.00
gpt-4.1-mini ↳ gpt-4.1-mini-2025-04-14	$0.40	$0.10	$1.60
gpt-4.1-nano ↳ gpt-4.1-nano-2025-04-14	$0.10	$0.025	$0.40
gpt-4.5-preview ↳ gpt-4.5-preview-2025-02-27	$75.00	$37.50	$150.00
gpt-4o ↳ gpt-4o-2024-08-06	$2.50	$1.25	$10.00

Рисунок 3.1 – Таблиця цін на найновіші LLM моделі OpenAI

До того ж, відкритий доступ до LLM породжує ризик зловживань – користувачі нерідко використовують чат для сторонніх завдань, таких як написання творів, вирішення задач або побутових порад. Це збільшує навантаження на систему, спотворює статистику та не виконує основну функцію довідника для абітурієнтів. Ще одним критичним недоліком є неможливість повного контролю над змістом відповіді: LLM може згенерувати неточну або вигадану інформацію, що в контексті вступної кампанії є вкрай небезпечним.

Альтернативою виступає embedding-пошук, який реалізований у розробленому чат-виджеті. Замість генерації відповіді з нуля, система виконує пошук заздалегідь підготовлених текстових фрагментів на основі їхніх векторних представлень. Коли користувач вводить запит, він перетворюється на числовий вектор за допомогою embedding-моделі, після чого порівнюється з базою векторів, створеною на основі матеріалів університету. Це дозволяє миттєво знаходити найбільш релевантну відповідь, що суттєво пришвидшує процес, знижує вартість і гарантує достовірність відповіді. До того ж, embedding-підхід забезпечує повний контроль над джерелами знань – усі відповіді походять виключно з авторизованої бази знань, сформованої та перевіреної адміністрацією. Ціни на Embeddings моделі OpenAI зображено на рис. 3.2.



Embeddings Price per 1M tokens · Batch API price

Model	Cost
text-embedding-3-small	$0.02
text-embedding-3-large	$0.13
text-embedding-ada-002	$0.10

Рисунок 3.2 – Таблиця цін на Embeddings моделі OpenAI

Для реалізації embedding-пошуку було обрано модель text-embedding-3-large від OpenAI. Вона вирізняється високою точністю, підтримкою української мови, стабільною документацією та прозорою ціною – лише 0.0001 долара США за 1000 токенів. Враховуючи, що середній запит абітурієнта складається з 10 токенів, вартість обробки одного embedding-запиту є мізерною – всього 0.000001 долара. Для порівняння: 1000 запитів на день обійдуться у 0.001 долара, що в 10 000 разів дешевше за LLM-підхід. Водночас, якість пошуку залишається на дуже високому рівні, особливо завдяки попередній підготовці бази знань та використанню PgVector в Supabase – рішення, яке дозволяє швидко індексувати, зберігати та порівнювати великі масиви embedding-векторів у хмарній базі даних.

На ринку існують інші провайдери embedding-моделей – зокрема, Cohere, Mistral, або open-source рішення на базі Sentence-BERT чи BGE. Проте ці моделі мають свої обмеження: деякі не підтримують українську мову, інші потребують локального хостингу, що ускладнює розгортання. Модель text-embedding-3-large, навпаки, забезпечує оптимальний баланс між якістю, швидкістю, вартістю та простотою інтеграції, особливо в середовищі Next.js та Supabase [24] [25].

3.2 Налаштування бази даних Supabase для embedding-пошуку

Для збереження та обробки embedding-векторів у рамках чат-виджета було обрано платформу Supabase – сучасну backend-платформу, побудовану на базі PostgreSQL. Вона ідеально підходить для інтеграції з frontend-додатками,

зокрема завдяки підтримці таких важливих функцій, як авторизація, обробка даних у реальному часі та наявність серверних функцій (stored procedures). Проте вирішальним аргументом на користь Supabase стало її розширення PgVector – спеціальний модуль PostgreSQL, який надає повноцінну підтримку векторного пошуку без необхідності звертатися до сторонніх сервісів або використовувати окремі ML-платформи [26] [27].

PgVector дозволяє зберігати векторні представлення безпосередньо у самій базі даних, як звичайні поля типу vector, що значно спрощує архітектуру проєкту. При цьому пошук за векторами реалізується через прості SQL-запити з використанням спеціалізованих операторів. Це означає, що жодна частина системи не потребує завантаження всіх ембеддингів у пам’ять додатку або окремого векторного сервера – увесь пошук відбувається безпосередньо на рівні СУБД. Такий підхід забезпечує надзвичайно швидку обробку запитів, що вимірюється мілісекундами, та водночас дає змогу масштабувати рішення до великої кількості запитань без суттєвих витрат ресурсів.

Крім того, інтеграція пошуку прямо в базу дозволяє централізовано управляти всією логікою: усі запити, їхня актуальність, відповідність ембеддингів і запитів користувача – все зберігається в єдиному середовищі, що робить систему стабільною, прозорою та контрольованою. Завдяки такій архітектурі Supabase виступає не лише як зручний API-рівень, а як повноцінна векторна база знань, яка об'єднує у собі дані, пошук і безпеку [27] [28].

У базі даних для забезпечення роботи чат-виджета з embedding-пошуком було створено дві ключові таблиці: Questions та Embeddings.

Таблиця Questions містить усі запитання, які можуть бути поставлені абітурієнтами. Вона слугує джерелом інформації для генерації векторних представлень (ембеддингів) і побудови логіки відповіді на запити користувачів. Кожен запис у цій таблиці містить не лише формулювання запитання, а й його відповідь, ключові слова, а також технічні поля, які дозволяють контролювати синхронізацію та оновлення ембеддингів. Додатково передбачено зберігання метаданих для більш гнучкої обробки запитів.

Поля таблиці Questions:

– question – формулювання запитання користувача;

– answer – детальна відповідь, яка пояснює суть питання;

– keywords – список пов’язаних ключових слів, які покривають варіації формулювання;

– metadata – додаткові поля для віджету;

– created_at – дата створення запитання;

– updated_at – дата останнього оновлення запису;

– synchronized_at – дата останньої синхронізації ембеддинга з OpenAI.

Таблиця Embeddings відповідає за зберігання векторних представлень запитань. Кожен вектор пов’язаний з конкретним записом з таблиці Questions і використовується для швидкого пошуку найбільш релевантного запиту в базі через векторне порівняння (рис. 3.3) [26] [27].

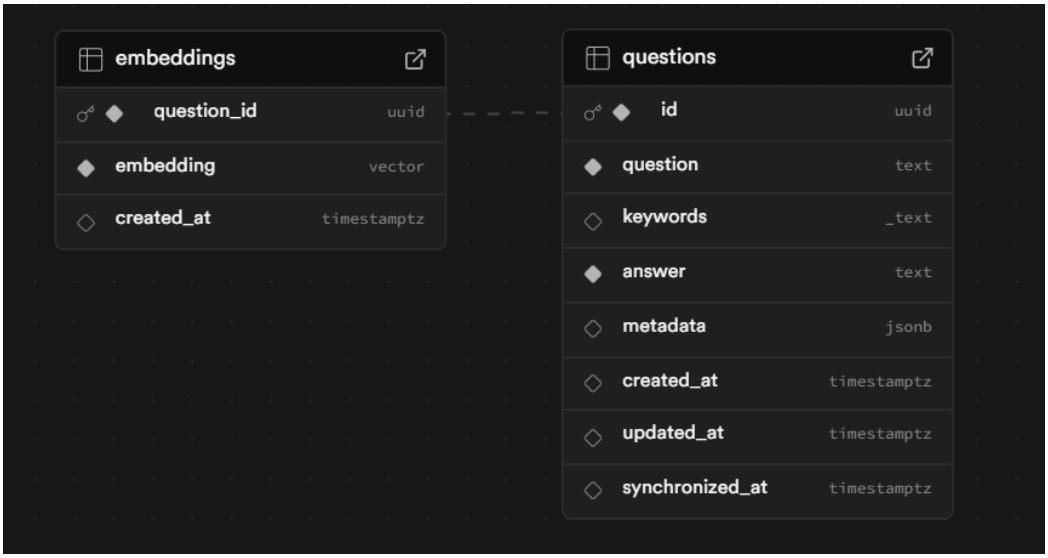


Рисунок 3.3 – Структура бази даних Supabase

Для створення ембеддингів використовується не лише саме питання (question) і ключові слова (keywords), але й відповідь (answer). Це важливо, оскільки користувач зазвичай формулює запит, виходячи не з конкретного питання, а з інформації, яку він хоче отримати. Наприклад, замість прямого запитання "Які документи потрібні для вступу?" користувач може написати:

"Хочу подати документи – що потрібно?". Тому включення поля answer у вхідний текст для embedding-моделі є більш ефективним і правильною практикою – це підвищує точність пошуку.

У Supabase була створена серверна функція find_most_similar_question, яка приймає на вхід векторне представлення запиту користувача (embedding) та необов’язковий параметр обмеження кількості результатів (за замовчуванням 1). Ця функція виконує пошук найбільш схожих запитань у базі даних, порівнюючи вектори embedding за допомогою оператора <=>, що визначає відстань між векторами. В результаті вона повертає відповідь на запитання, метадані та показник схожості (розрахований як 1 мінус відстань), сортує результати за зростанням відстані (тобто від найбільш релевантних) і обмежує кількість повернутих записів згідно з параметром result_limit. Така реалізація дозволяє швидко знаходити найбільш релевантні відповіді без необхідності завантажувати всі embedding-и у пам’ять, виконуючи векторний пошук безпосередньо в базі даних.

3.3 Інтерфейс адміністратора: керування питаннями та синхронізація

Опис адміністративного інтерфейсу та його функцій є критично важливою частиною всієї архітектури чат-виджета. Щоб забезпечити зручне та надійне керування embedding-питаннями, у приватній частині проєкту, що захищена через ProtectedLayout, була створена адмінпанель у вигляді сторінки Dashboard. Усі її функції доступні виключно авторизованим користувачам з відповідними правами, і вхід на цю сторінку реалізований через окрему сесію, яка перевіряється за допомогою авторизації через Supabase. Для цього було створено спеціальну обгортку withAuth, яка використовується для бекенд-ендпоінтів. Кожен такий ендпоінт спочатку перевіряє, чи присутній авторизаційний токен у cookie, чи він валідний, та чи відповідає користувач встановленим умовам

доступу. Це дозволяє повністю контролювати безпеку доступу до ключових адміністративних функцій.

Основним функціональним центром цієї сторінки є кнопка «Синхронізувати питання», яка викликає серверну функцію. Вона проходиться по таблиці Questions, і відбирає лише ті питання, у яких значення поля synchronized_at менше, ніж значення поля updated_at. Таким чином, синхронізуються лише нові або змінені записи, що дозволяє значно зекономити токени OpenAI API та зменшити час очікування. Після цього для кожного відповідного запису викликається функція, яка створює векторне представлення (embedding) з тексту question, keywords, та обов'язково – answer. Як уже згадувалося раніше, включення answer у вхідний текст є важливою практикою, оскільки користувач зазвичай шукає не буквальний текст запитання, а відповідь, яка містить потрібну йому інформацію.

Щоб зробити процес синхронізації наочним, до кнопки було додано індикатор завантаження (loader). Це дозволяє адміну бачити, що запит виконується і не натискати її повторно. Всі подібні елементи на сторінці Dashboard – включно з іншими кнопками та формами – мають аналогічні індикатори активності, що значно покращує UX у межах адміністративного функціоналу.

Крім того, для зручного додавання нових питань була реалізована функція автоматичного збору запитань з будь-якої сторінки сайту. Для цього створена друга ключова кнопка на Dashboard – «Додати запитання», яка відкриває модальне вікно, у якому адміністратор може ввести URL-адресу сторінки (наприклад, сторінки підготовчого відділення на офіційному сайті університету). Додатково можна ввести користувацький запит (user prompt), у якому конкретизується, які саме питання повинна згенерувати нейромережа. Наприклад, адмін може вказати: «Згенеруй 5 типових питань від абітурієнтів щодо умов вступу на підготовче відділення» (рис. 3.4).

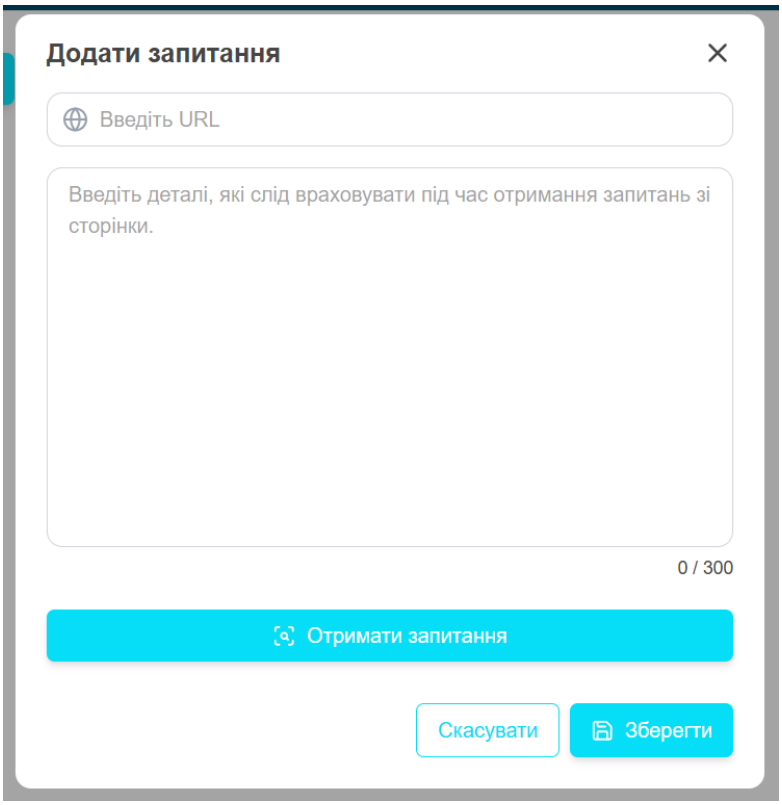


Рисунок 3.4 – Форма створення запитань

Ця інформація передається до спеціального серверного ендпоінта, який викликає модель GPT-4 (4.0-all) у streaming-режимі, щоб відповіді поверталися поступово, рядок за рядком, без очікування повного завершення генерації. Це суттєво пришвидшує роботу та дає змогу адміну бачити хід генерації в реальному часі. Модель також отримує системний промпт, який вказує їй, у якому форматі необхідно надсилати відповідь (зазвичай – масив об'єктів з ключами question, answer, keywords, тощо).

Після завершення генерації, отримані JSON-об’єкти розбираються на фронтенді та виводяться у вигляді попереднього перегляду. Адміністратор може переглянути їх та натиснути кнопку «Зберегти», яка відправляє об'єкти безпосередньо до таблиці Questions у Supabase (рис. 3.5).

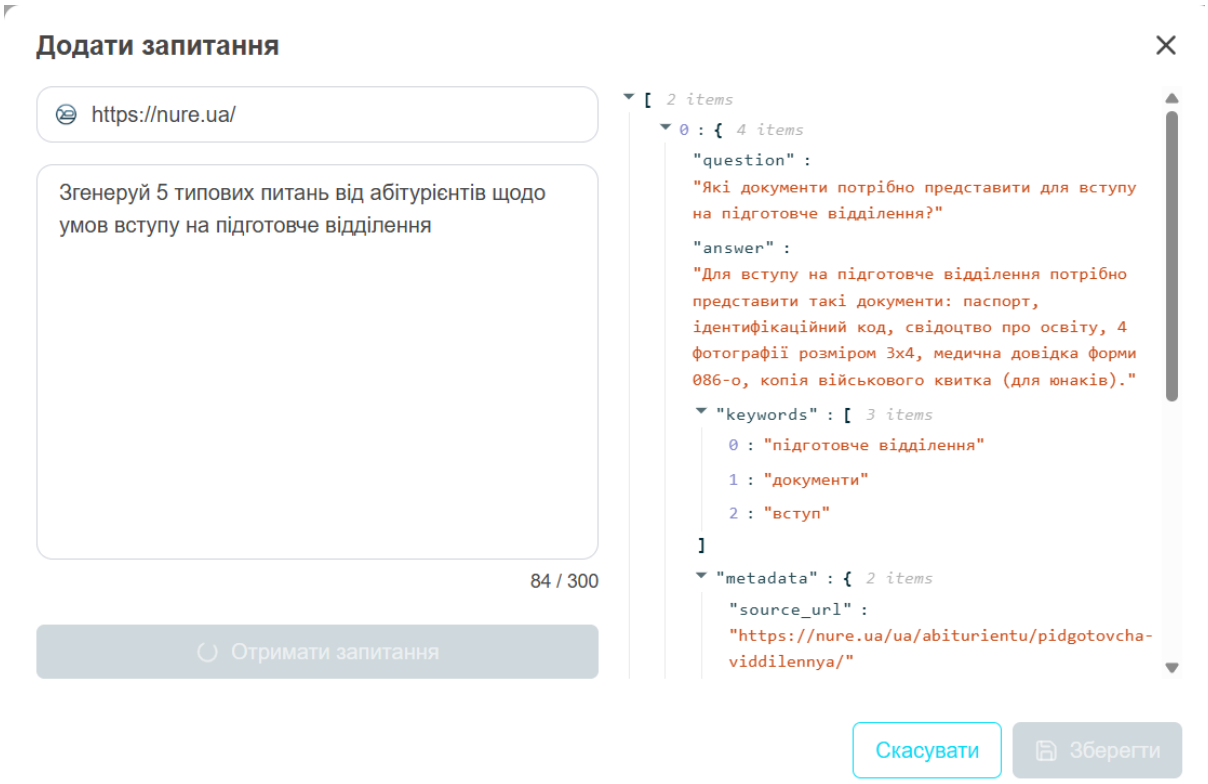


Рисунок 3.5 – Результат створення запитань

Усі записи одразу отримують позначку created_at, а також значення updated_at, яке потім використовується для перевірки необхідності синхронізації.

Важливо підкреслити, що адмінпанель є інтегрованою частиною основного застосунку – вона не винесена в окремий проєкт, що дозволяє спростити архітектуру, спільно використовувати компоненти, контексти та сесію, а також продемонструвати повний цикл створення та обробки контенту без перемикання середовищ або репозиторіїв.

Ще одним важливим елементом безпеки та персоналізації є реалізація механізму авторизації через єдину формулу Login, яка інтегрується з Supabase Auth (рис. 3.6).

Завдяки цьому кожен запит до адміністративних функцій не лише перевіряє наявність токена, а й фіксує, який саме користувач його здійснив. Це дає змогу формувати лог аудиту дій, зберігати інформацію про те, хто додав або змінив конкретне запитання, та у разі потреби – відновити хронологію змін.

Таким чином, система не лише захищає доступ, а й дозволяє відстежувати внесок кожного адміністратора в загальну базу знань.

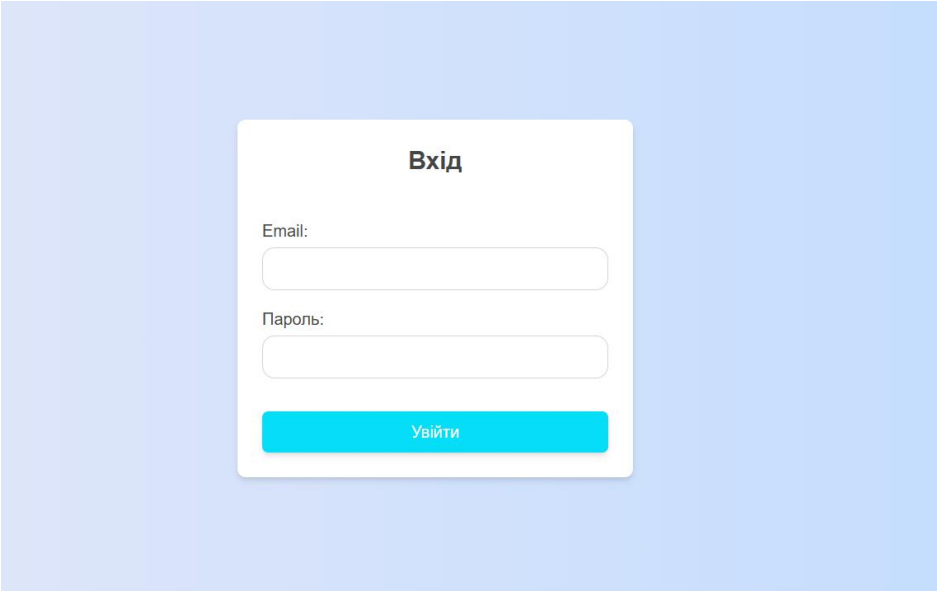


Рисунок 3.6 – Форма Login

У підсумку, реалізація панелі адміністратора з таким функціоналом дозволяє ефективно керувати базою знань, швидко оновлювати інформацію, реагувати на зміни в інформаційних матеріалах університету та створювати нові питання буквально за кілька хвилин. Це рішення повністю відповідає вимогам до сучасної AI-системи з орієнтацією на продуктивність, безпеку та масштабованість.

3.4 Реалізація ChatWidget

Для реалізації інтерактивного чат-виджета було створено ключові компоненти: ChatInput – поле введення запитань користувачем, та ChatMessageDisplay – компонент відображення повідомлень. При відкритті сторінки користувачу анімовано показується кнопка відкриття віджета помічника, що одразу привертає увагу (рис. 3.7).

Рисунок 3.7 – Кнопка для відкриття віджета помічника

Натискання на неї запускає плавну анімацію відкриття самого чат-вікна, створюючи відчуття живої взаємодії.

ChatMessageDisplay використовує концепцію PolyRole для визначення автора повідомлення: якщо повідомлення від асистента – воно розміщується зліва, якщо від користувача – справа, що робить інтерфейс зрозумілішим і зручнішим.

ChatInput динамічно відображається – поки бот формує відповідь, поле введення приховується, щоб уникнути відправки нових повідомлень раніше часу. Під час очікування відповіді видно лоудер. Хедер чат-вікна містить назву помічника і кнопку закриття (рис. 3.8).

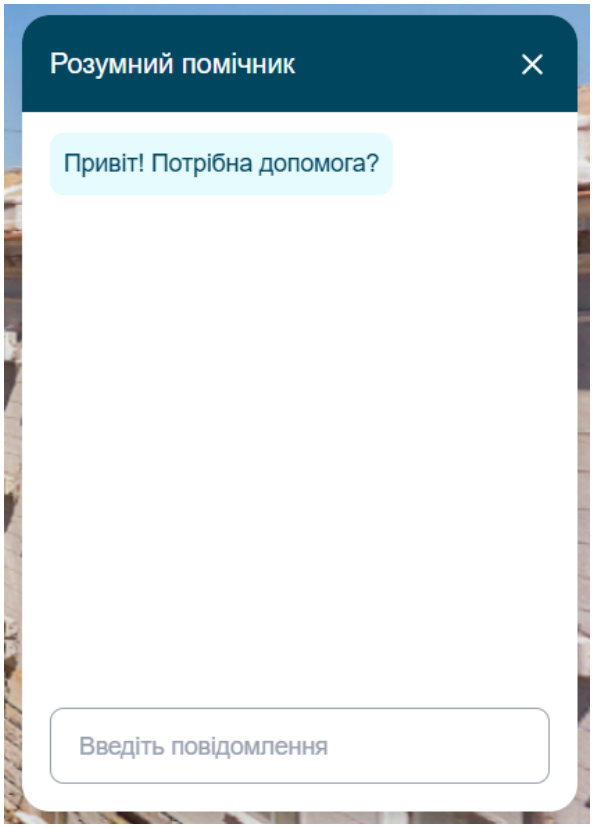


Рисунок 3.8 – Віджет чату помічника

Запити користувача обробляються через SSR-бекенд-функцію, яка виконує пошук релевантних ембеддингів у базі Supabase, забезпечуючи швидкість і точність.

Для кращої реалістичності відповіді виводяться поступово – за допомогою тайм-аутів текст дописується по частинах, створюючи ефект «друкування». Після завершення друкування може викликатися callback-функція для плавного переходу.

Для покращення взаємодії з користувачем також була реалізована логіка автоматичного скролу донизу чат-вікна при появі нових повідомлень або під час очікування відповіді – наприклад, коли з’являється лоудер чи повідомлення від асистента. Вікно автоматично прокручується вниз із плавною анімацією, щоб користувач завжди бачив актуальний стан діалогу. Водночас враховано, що користувач може самостійно прокручувати чат угору – якщо прокрутка піднімається вище певного порогу, автоматичний скролл тимчасово вимикається, дозволяючи вільно переглядати історію без "перескакувань".

Щойно користувач повертає прокрутку донизу, скролл знову "прикріплюється" до останнього повідомлення і стежить за оновленням переписки. Уся ця логіка винесена в окремий кастомний хук useScrollToBottom, який повторно використовується не лише у ChatWidget, але й у адміністративній панелі, де необхідно автоматично прокручувати відповідь від ШІ при генерації запитань для сайту.

Важливою частиною стало збереження всієї історії повідомлень у полі conversation. Це поле зберігає весь діалог між користувачем і ботом. Коли користувач закриває чат-виджет, у кожного повідомлення в цій історії ставиться прапорець read: true. Це дозволяє при наступному відкритті чату уникнути повторного відтворення анімації друкування старих повідомлень – користувач бачить історію миттєво, без зайвих затримок.

Ще одна важлива функція – це відображення індикації точності відповіді. Якщо оцінка релевантності ембеддинга низька, поруч із повідомленням

з’являється жовта іконка. При наведенні на неї користувач бачить пояснення, що чат не впевнений у точності відповіді, і вона може не зовсім відповідати на запитання. Це додає прозорості в роботі системи і допомагає користувачеві критично сприймати отриману інформацію.

У процесі тестування виявилась проблема: якщо користувач у першому питанні уточнює спеціальність, а в подальших питаннях цього контексту немає, то ембеддинги не можуть правильно зрозуміти, до якої саме спеціальності належить нове запитання. Адже схожих питань існує багато для кожної спеціальності.

Для розв’язання цієї задачі була реалізована логіка збереження контексту, що зберігається у metadata повідомлення. Якщо у питання є контекст, то під самим повідомленням з’являється іконка у вигляді «Reply» (аналогічно популярним месенджерам). Клік на цю іконку активує додаткову панель над полем введення, де показується історія спілкування в контексті вибраного питання. Цю панель можна закрити, щоб позбавитися контексту і задавати нові питання незалежно.

Також у metadata повідомлення зберігається source_link – посилання на першоджерело інформації. Під повідомленням відображається відповідна кнопка, яка дає змогу користувачу перейти до детальнішої інформації на оригінальному сайті чи сторінці. Це підвищує довіру до відповіді і дозволяє отримати більше деталей за потреби.

Окрім цього, для деяких питань була реалізована додаткова логіка уточнення через поле clarifyingQuestion. Наприклад, якщо користувач запитує: «Який прохідний бал на спеціальність?», але не уточнює, про яку саме спеціальність йдеться, то система активує clarifyingQuestion, який просить користувача уточнити спеціальність або пов’язану інформацію. Відповідь на це уточнення автоматично використовується для формування контексту – вона прикріплюється до початкового питання і дозволяє точніше сформулювати відповідь. Таким чином, кожне наступне питання враховує уточнений контекст, що значно підвищує релевантність відповідей (рис. 3.9)

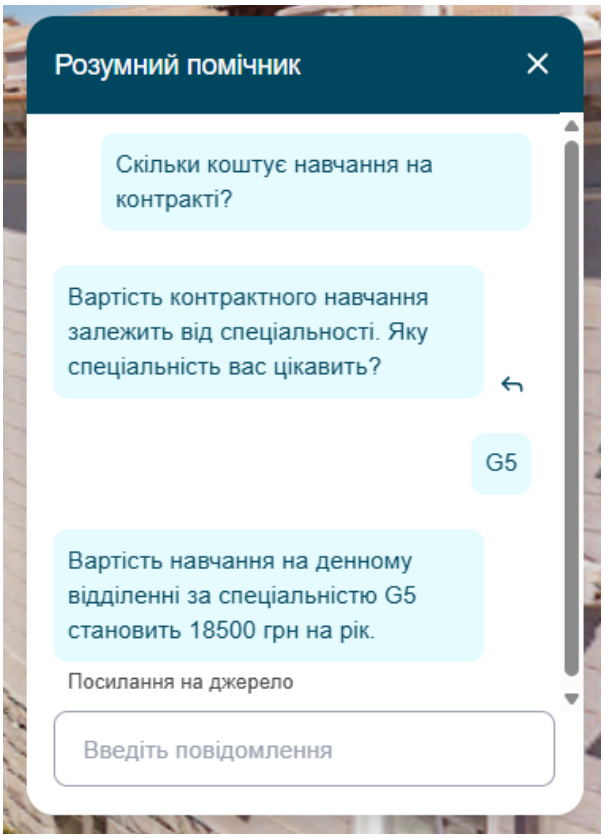


Рисунок 3.9 – Уточнююче питання

Щоб уникнути ситуації, коли після відповіді на clarifyingQuestion система знову повертає той самий попередній ембеддинг, було впроваджено механізм подвійного пошуку. Ми повертаємо два найбільш релевантні ембеддинги.

У таких випадках загальний ембеддинг (наприклад, питання про «вартість контрактного навчання») зазвичай має більший обсяг тексту і вищу вагу, тому він потрапляє першим. Але ми перевіряємо, який ембеддинг уже був показаний користувачу, і якщо перший збігається з попереднім – повертаємо другий, специфічніший (наприклад, той, що сформований на основі відповіді «G5»).

Таким чином, коли користувач відповідає на уточнююче питання (наприклад, вказує спеціальність G5), система може правильно знайти і показати відповідь саме про вартість навчання на обраній спеціальності, а не повторити загальну інформацію. Це дозволяє уникнути циклів і підвищує точність та релевантність діалогу.

Наприкінці вікно чату було винесено в окремий компонент, що робить його зручним для повторного використання. Такий підхід дозволяє виводити чат не лише безпосередньо у проєкті, але й розміщувати його на окремому роуті, а також вбудовувати як iframe у інші додатки чи сайти, що значно розширює можливості інтеграції ChatWidget.

3.5 ChatWidgetContext: глобальне керування станом віджета чату

У процесі побудови масштабованої архітектури чат-виджета постала важлива проблема: як зробити так, щоб стан відкритості виджета (open) можна було контролювати з будь-якої точки застосунку, не передаючи цей стан через довгі ланцюжки пропсів між батьківськими й дочірніми компонентами. Наприклад, якщо ми хочемо, щоб натискання кнопки у хедері або на сторінці спеціальностей відкривало чат, а сам ChatWidget розміщений у футері або в глобальному layout, – це ставало складною задачею без централізованого керування станом.

Для вирішення цієї проблеми був створений глобальний контекст ChatWidgetContext, який став єдиним джерелом істини для всіх аспектів, пов’язаних з ChatWidget. Цей контекст дозволяє не лише контролювати відкриття або закриття виджета з будь-якої частини застосунку, а й забезпечує централізоване зберігання всіх повідомлень, логіку роботи з контекстами запитань, оновленнями повідомлень, і навіть їх маркування як прочитаних.

У межах цього контексту зберігається історія спілкування користувача з помічником – об’єкт conversation. Він дозволяє не лише відтворити повідомлення після оновлення сторінки або переходу між маршрутами, а й реалізувати складну логіку, таку як поступове відображення повідомлення (ефект "написання"), збереження повідомлень у локальному або серверному сховищі, додавання реплаїв із контекстом тощо. Кожне повідомлення в conversation має окреме поле read, яке оновлюється після закриття чату, щоб при

повторному відкритті не запускати знову анімацію друкування вже прочитаних відповідей. Це створює відчуття "живого" та послідовного діалогу для користувача, без повторів і затримок.

Окремо важливо зазначити, що інтеграція ChatWidgetContext безпосередньо в layout застосунку дозволила зберігати увесь стан навіть при переході між сторінками. Це особливо важливо для застосунків, де користувач часто переходить із загальної інформації до вузько спеціалізованих сторінок (наприклад, зі стартової сторінки на сторінку певної спеціальності або програми навчання). Користувач може розпочати діалог на одній сторінці, а потім – без переривання сесії – продовжити діалог на іншій, бачачи всю історію та контекст розмови.

Ще одним важливим аспектом стало те, що завдяки ChatWidgetContext стало можливим викликати функції для керування чатом, змінювати стан відкритості, додавати або видаляти повідомлення, керувати контекстами запитів з будь-якого місця.

Щоб зробити використання цього контексту максимально зручним, був реалізований кастомний хук useChatWidget(), який дозволяє витягувати усі потрібні змінні та функції одним викликом. Це значно спрощує інтеграцію ChatWidget у будь-який компонент застосунку.

Завдяки цій архітектурі, чат став не просто додатковим інструментом підтримки, а повноцінним інтегрованим помічником, який "живе" у застосунку постійно, зберігає контекст, запам’ятовує історію розмов і забезпечує справжню безшовну взаємодію з користувачем.

4 РОЗРОБКА АДАПТИВНОГО ТЕСТУ ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ З ДИНАМІЧНИМ ФОРМУВАННЯМ РЕКОМЕНДАЦІЙ

4.1 Інтеграція тесту у ChatWidget та UI-реалізація

Для реалізації адаптивного тесту профорієнтації у межах уже існуючого чат-інтерфейсу було важливо чітко розділити логіку звичайного спілкування з чат-ботом та проходження тесту. З цією метою в контекст ChatWidget було додано окрему змінну quizOpen, яка відповідає за стан: відкрито зараз тест чи звичайний чат. Це дозволило гнучко управляти інтерфейсом та відображенням компонентів, в залежності від режиму.

Коли quizOpen активується, в інтерфейсі автоматично відбувається плавна анімація розширення стандартного чат-виджета до тестового режиму. Такий перехід реалізовано з використанням анімованого зміщення висоти і padding, що створює враження живого, гнучкого інтерфейсу. У разі завершення або закриття тесту – відбувається зворотна анімація, і чат повертається до звичайного вигляду.

Оскільки логіка тесту принципово відрізняється від класичної бесіди, у коді ChatWidget реалізовано окремий масив quizConversation, який відображається замість стандартного conversation, коли відкрито тест. Кожне повідомлення у цьому масиві містить не тільки текст запитання й відповіді, а й додаткові метадані, зокрема поле answers, у якому визначено можливі варіанти відповідей.

Для взаємодії з цими варіантами створено окремий компонент QuizChoices, який плавно з’являється знизу при кожному новому питанні. У разі трьох варіантів відповіді вони відображаються в один ряд, що підходить, наприклад, для питань типу «Так / Ні / Не впевнений» (рис. 4.1).

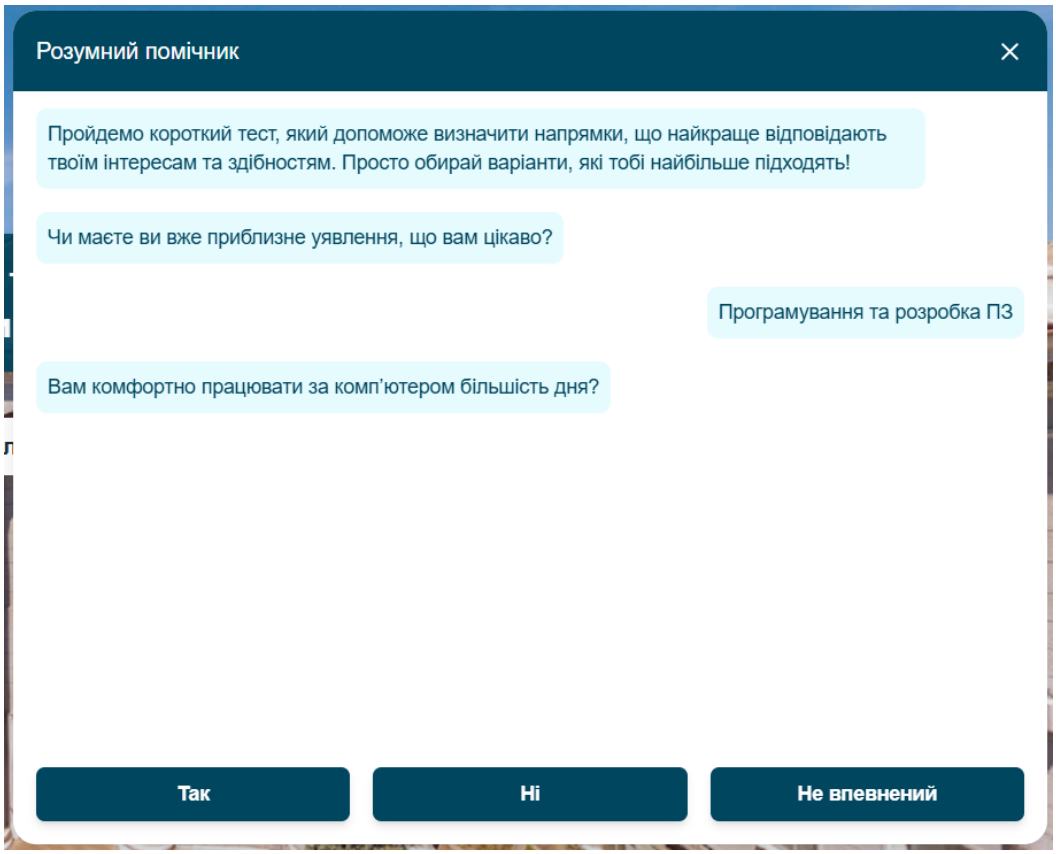


Рисунок 4.1 – Відображення трьох варіантів відповідей

Якщо ж варіантів більше трьох, то вони автоматично рендеряться у вигляді сітки по два в ряд, що забезпечує як естетичність, так і зручність взаємодії. Візуально варіанти відповіді мають іконку – або порожній квадратик, або галочку, що з'являється при виборі. Це дозволяє реалізувати як одиночний вибір (single-select), так і множинний (multi-select).

Щойно користувач обирає хоча б одну відповідь, з'являється анімована кнопка Submit, натискання якої надсилає відповідь (рис. 4.2).

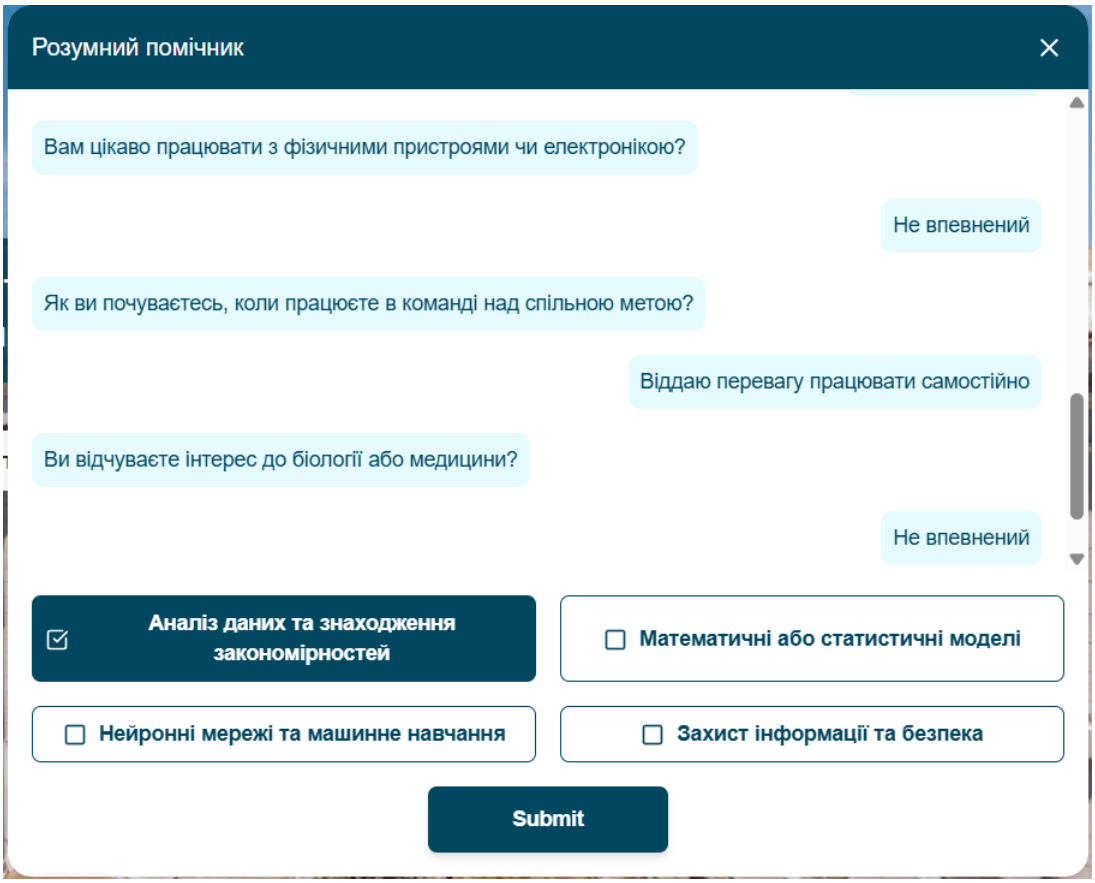


Рисунок 4.2 – Відображення multi-select варіантів відповідей

Після надсилання, інтерфейс очікує на нове запитання або результат – аналогічно до роботи зі звичайним чат-ботом, однак логіка при цьому ізольована.

Для деяких питань у метаданих було додано поле testTrigger, яке дозволяє відображати кнопку для початку тесту. Наприклад, якщо користувач запитує, як визначити спеціальність, яка спеціальність йому підходить або як обрати спеціальність, система пропонує пройти тест і показує відповідну кнопку для його запуску (рис. 4.3).

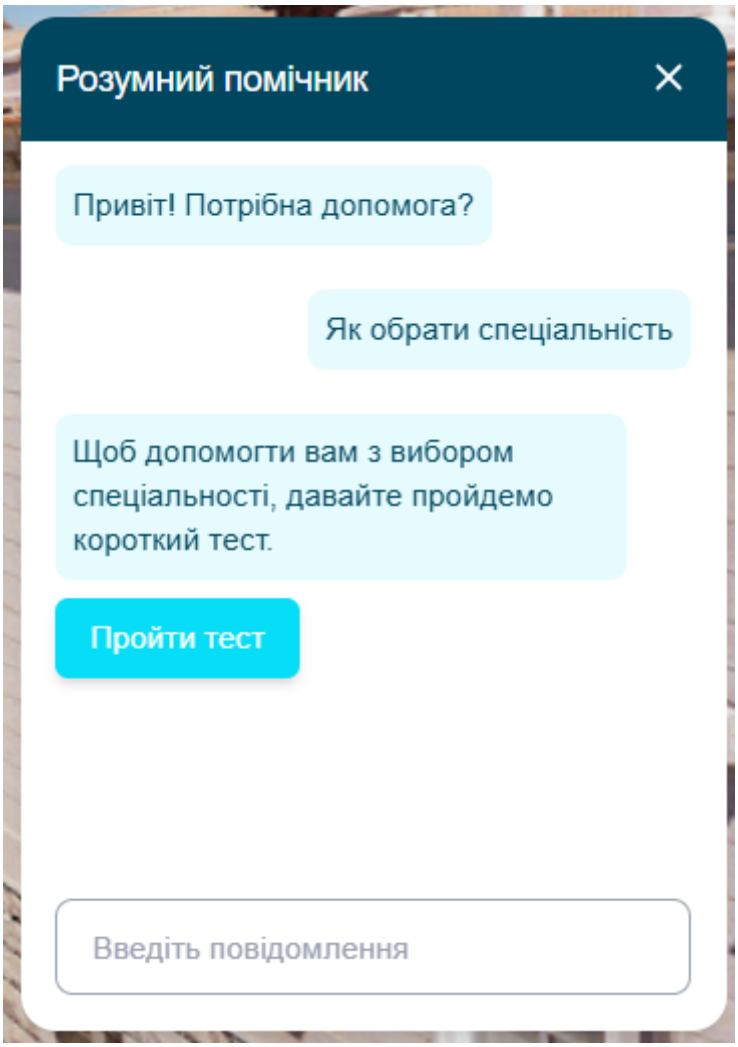


Рисунок 4.3 – Кнопка пройти тест

Коли користувач закриває тест, усі повідомлення в quizConversation позначаються як read: true, щоб при повторному відкритті не анімувати вже отримані відповіді повторно. Одночасно з цим закривається й сам чат-виджет. При його наступному відкритті користувач потрапляє до звичайного режиму чат-бота.

Щоб зберегти логіку взаємодії та не втратити прогрес, було реалізовано механізм «нагадування» про незавершений тест. Якщо quizConversation зберігся, але quizOpen неактивний, бот автоматично вставляє у кінець звичайної розмови повідомлення з текстом: «Ви почали проходити тест, але не завершили його. Бажаєте продовжити?» разом з кнопкою продовження (рис. 4.4). При цьому враховано, що таке повідомлення не повинно дублюватися, якщо воно вже було вставлено раніше. Це робить чат розумнішим і персоналізованішим.

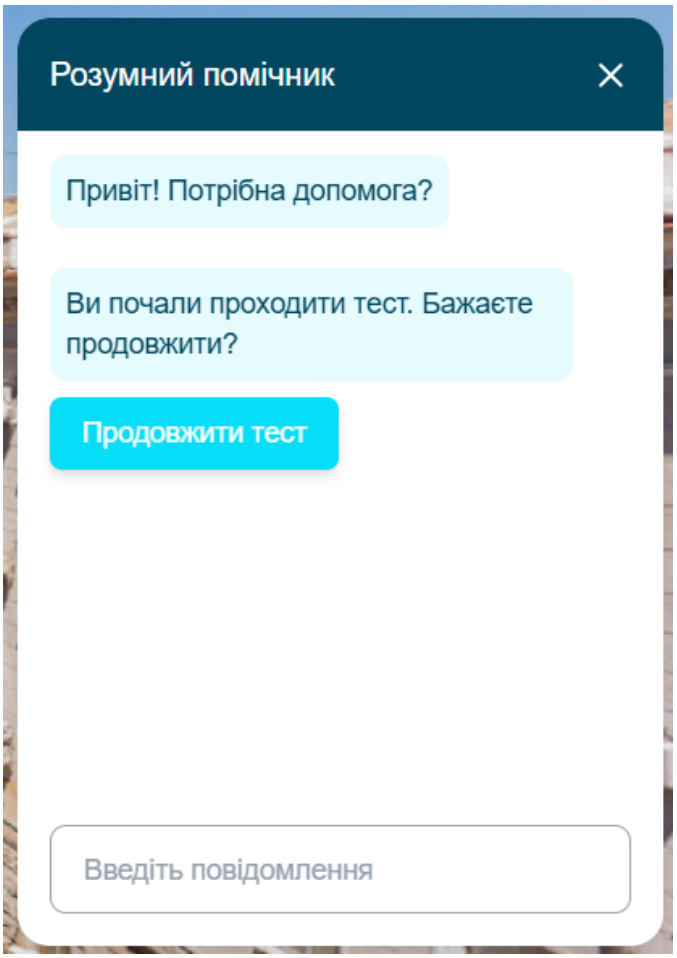


Рисунок 4.4 – Кнопка продовжити тест

Окрему увагу приділено візуальній частині: QuizChoices з’являється поступово – варіант за варіантом, створюючи ефект, ніби інтерфейс «оживає». Усі елементи з'являються анімовано знизу, що значно підвищує привабливість тесту для користувача.

Загалом, таке інтегроване рішення дозволяє абітурієнту без перемикань між сторінками пройти професійний тест у знайомому середовищі спілкування з ботом. UI-рішення роблять інтерфейс не тільки функціональним, але й емоційно приємним у використанні.

4.2 Динамічне формування структури тесту

Ключова задача адаптивного профорієнтаційного тесту – не просто задати набір універсальних питань, а створити динамічну структуру, яка змінюється відповідно до відповідей користувача. Тест повинен адаптуватися до його інтересів і поступово звужувати тематику, формуючи персоналізовану освітню траєкторію.

Для реалізації адаптивної логіки в тесті була запроваджена система інтересів QUIZ_TOPIC. Це перелік потенційних напрямів, до яких може проявити схильність користувач. Наприклад:

```
export enum QUIZ_TOPIC {
  PROGRAMMING = 'PROGRAMMING',
  MATH = 'MATH',
  PHYSICS = 'PHYSICS',
…
```

Інтереси виступають своєрідними "тегами", які накопичуються у користувача у відповідь на його дії. В залежності від наявних інтересів, тест обирає релевантні наступні питання.

Кожне питання у тесті може мати поле requiredInterests. Це означає, що питання буде показане тільки тоді, коли у користувача вже є хоча б один з необхідних інтересів. Наприклад, питання про створення комп’ютерних ігор не буде задане абітурієнту, який ще не виявив інтерес до програмування.

Для старту тесту був розроблений спеціальний вступний блок питань – UNSURE_QUIZ_QUESTIONS. Ці запитання мають нейтральний характер і не потребують жодного requiredInterest. Їхня мета – м’яко зібрати перші ознаки інтересу. Наприклад «Які предмети вам подобались у школі?» або «Що з цього вас цікавить найбільше: створення чогось руками, розрахунки, комунікація?»

Якщо користувач обирає "Інформатика", то до його профілю додається інтерес PROGRAMMING. Тоді тест "розблоковує" запитання з цим requiredInterest.

Алгоритм підбору наступного запитання враховує переважаючий інтерес. Наприклад, якщо у користувача вже 4 очки по PROGRAMMING і лише 2 по MATH, наступне релевантне запитання буде саме з програмування. Такий механізм забезпечує відчуття логічного прогресу в тесті.

Усі питання згруповано по QUIZ_TOPIC, з поступовим ускладненням. Це дає змогу уникнути різких переходів – усі питання мають логічну прогресію. Наприклад, блок PROGRAMMING містить такі питання:

– чи подобається вам програмування?
– які мови програмування ви хотіли б вивчити?
– які програми ви хотіли б створювати?
– вам цікаво створення мобільних додатків, вебсайтів чи щось інше?

Для виключення зайвих запитань була реалізована система відмови від інтересу. Якщо користувач прямо вказує, що певний напрямок йому не цікавий, наприклад, відповідає «Ні» на «Чи цікаве вам програмування?», то до масиву dislikeInterests додається PROGRAMMING.

Подальші питання з requiredInterests у яких є PROGRAMMING ігноруються. Це забезпечується на етапі генерації наступного питання. Але навіть у такому разі dislikeInterests зберігається, щоб відобразити це в підсумковій аналітиці – негативна інформація теж важлива для точних рекомендацій.

Якщо у користувача виявлено інтерес до загальної теми, наприклад PROGRAMMING, то лише після цього починають з’являтися запитання по дочірнім напрямам, як-от WEB, MOBILE, AI. Таким чином зберігається поступовість: Загальний інтерес → Конкретизація → Уточнення профілю (приклад: Frontend, Backend, GameDev, AI). Це додає глибини і дозволяє будувати персоналізовану траєкторію без надлишку зайвих питань.

Початкове сортування користувачів у тесті реалізоване через вступне запитання: «Чи маєте ви вже приблизне уявлення, що вам цікаво?». Це дозволяє одразу визначити, чи має абітурієнт чітке уявлення про власні інтереси (рис. 4.5).

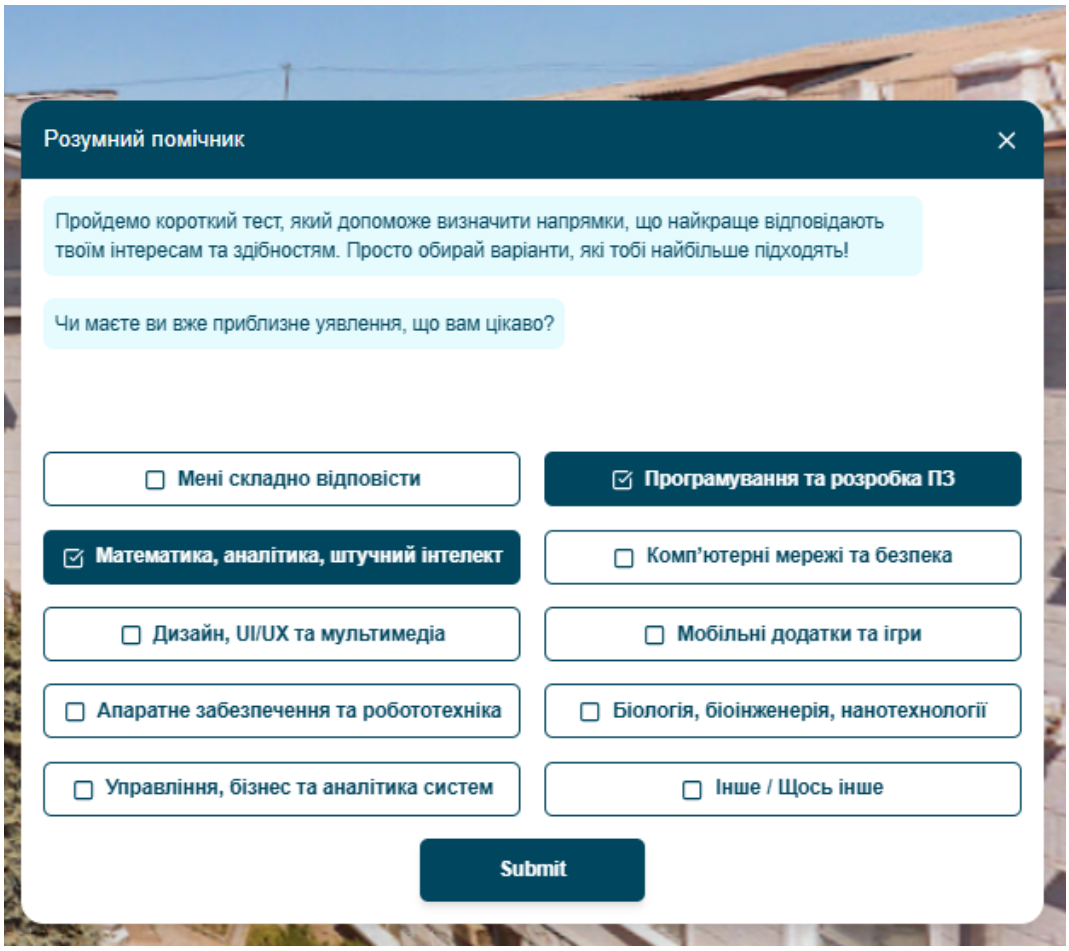


Рисунок 4.5 – Вступне запитання

Якщо користувач обирає конкретний напрямок, наприклад, «Інженерія програмного забезпечення» чи «Штучний інтелект», до його профілю одразу додається відповідний інтерес QUIZ_TOPIC. Після цього система починає формувати питання, які безпосередньо стосуються обраної теми. Це робить тест більш персоналізованим і допомагає користувачу отримати змістовні відповіді, що відповідають його реальним інтересам.

Якщо ж користувач не впевнений або не може чітко визначитися з напрямком, активується набір питань UNSURE_QUIZ_QUESTIONS. Вони допомагають м'яко виявити потенційні інтереси без зайвого тиску. Такий підхід робить тест гнучким і зручним для різних користувачів – від тих, хто вже має певні уподобання, до тих, хто лише шукає свій шлях.

4.3 Динамічне формування структури тесту

У процесі формування результатів адаптивного тесту ключову роль відіграє накопичення інтересів користувача протягом усього проходження квізу. Для цього у контексті ChatWidgetContext були додані спеціальні змінні – quizInterests та dislikeInterests. Вони зберігають не просто перелік тем, які сподобались або не сподобались абітурієнту, а ще й частотність згадування кожного інтересу. Таким чином, при кожному виборі користувачем тієї чи іншої відповіді, яка має асоціацію з певним інтересом, у відповідному об’єкті або створюється новий запис, або інкрементується вже існуючий. Це дає змогу не просто зафіксувати факт, що людині подобається, наприклад, програмування, а й зрозуміти, наскільки цей інтерес домінує над іншими. Поступове накопичення балів дозволяє побудувати багатогранну модель профілю користувача, яка враховує не лише кількість, а й послідовність проявів певного інтересу.

На базі цих даних виконується формування кінцевого результату – списку спеціальностей, які можуть бути найбільш релевантними для конкретного абітурієнта. Для цього був заздалегідь створений окремий масив з описами всіх можливих освітніх програм, які є у ВНЗ. Кожна з них містить назву спеціальності, факультет, до якого вона належить, а також список інтересів, які повинні бути у користувача, аби така спеціальність вважалась для нього потенційно придатною. Наприклад, для того щоб рекомендувати спеціальність на кшталт «Системна інженерія», у профілі користувача повинні бути виявлені інтереси до автоматизації, системного аналізу та розробки програмного забезпечення.

Завдяки наявним даним про інтереси абітурієнта система спочатку виконує порівняння кожної програми навчання з поточним станом quizInterests, фільтруючи лише ті, які не містять жодного інтересу зі списку dislikeInterests. Це дозволяє сформувати перший основний список результатів – саме ті освітні напрямки, які повністю узгоджуються з вподобаннями користувача, без конфліктів або сумнівів (рис. 4.6). Цей список є пріоритетним для демонстрації,

адже саме він створює враження, що система "розуміє" інтереси користувача та пропонує йому саме те, чого він хоче.

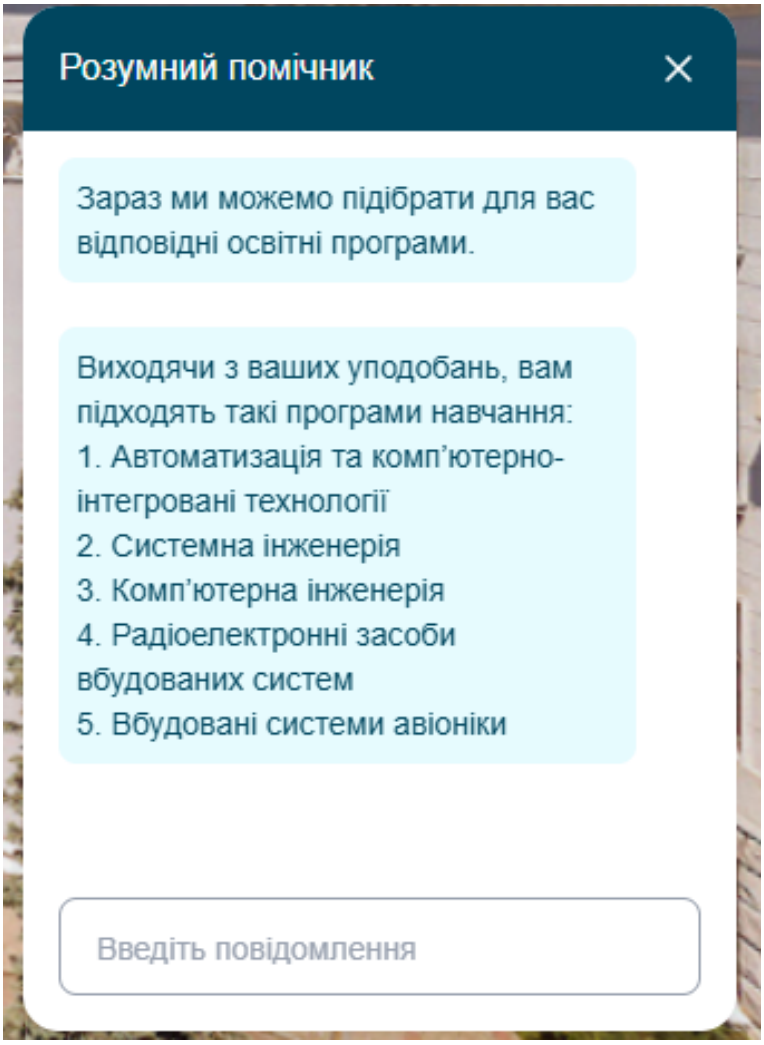


Рисунок 4.6 – Результати тесту відносно уподобань абітурієнта

Паралельно формується ще один список – альтернативних можливостей. До нього включаються всі ті освітні програми, які могли б бути потенційно релевантними, якби користувач не вказав, що йому щось не подобається. Цей список будується на основі тих самих інтересів, але вже без виключення dislikeInterests. Це дозволяє виявити програми, які могли бути несправедливо виключені, або ж навпаки – допомагає користувачу побачити ширшу картину можливостей. Якщо в альтернативному списку з'являються спеціальності, яких немає в основному, вони демонструються окремим блоком з поясненням, що це програми, які теж можуть бути йому цікаві, незважаючи на попередні відповіді

(рис. 4.7). Це підсилює відчуття довіри до системи, яка не просто діє за жорсткою логікою, а пропонує розширений контекст для роздумів.

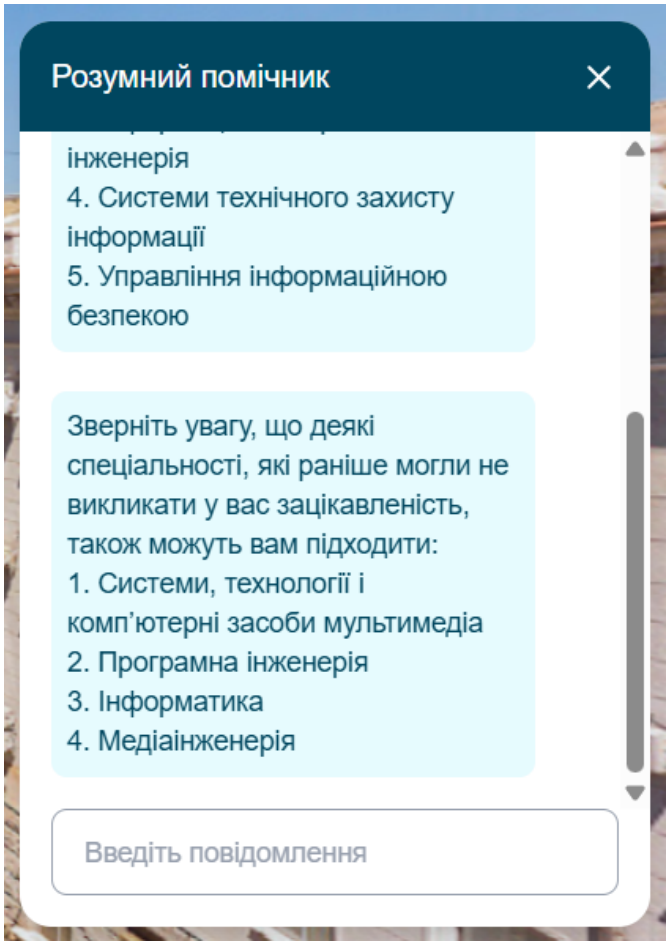


Рисунок 4.7 – Рекомендації відносно здібностей абітурієнта

Обидва списки піддаються сортуванню за ступенем відповідності. Для цього виконується підрахунок кількості інтересів, які збігаються з requiredInterests кожної спеціальності. Чим більше таких збігів, тим вище розміщується програма у фінальному результаті. Це дозволяє автоматично висунути на перші позиції ті напрями, які мають найбільшу кількість дотичних точок з профілем користувача. Наприклад, якщо в системі видно, що користувач отримав найбільше балів по темам «робототехніка» та «автоматизація», то програми, які поєднують обидва ці інтереси, опиняються у самому верху списку, навіть якщо вони мають складну структуру або не є масовими. У результаті користувач бачить у топі саме ті напрямки, які не просто відповідають його

поточному запиту, а й найкраще співпадають з його глибинними інтересами, навіть якщо він сам до кінця цього не усвідомлює.

Цей підхід дозволяє створити унікальний користувацький досвід – коли чат-виджет не просто ставить питання, а проводить абітурієнта через персоналізовану подорож з урахуванням його минулого досвіду, інтересів, сумнівів і рішень. Завдяки двом окремим списками рекомендацій і гнучкому механізму формування результатів, адаптивний тест виконує не лише профорієнтаційну, а й мотиваційну функцію, допомагаючи користувачу не просто вибрати спеціальність, а й побачити власний потенціал у новому світлі.

Після того як користувач відповідає на певну кількість запитань, наприклад, дванадцять, профорієнтаційний тест автоматично завершується. Це рішення базується на принципі досягнення достатньої кількості відповідей, щоб сформувати обґрунтовані рекомендації щодо вибору освітніх програм. У цей момент значення змінної quizOpen у контексті додатку змінюється на false, що автоматично призводить до зміни стану інтерфейсу: чат-виджет повертається до свого звичного компактного вигляду, адаптованого для вільного спілкування з чат-ботом без обмежень, пов’язаних з тестуванням.

Паралельно із завершенням тесту в чаті з’являється автоматичне системне повідомлення, яке містить результати проходження – список рекомендованих освітніх програм або напрямів підготовки, що найкраще відповідають інтересам і відповідям користувача. Це повідомлення виконує функцію підсумку, даючи змогу побачити власну освітню траєкторію в концентрованому вигляді. Водночас воно є своєрідним запрошенням до подальшої розмови та поглибленого вивчення запропонованих варіантів.

Оскільки після завершення тесту чат повертається у звичний режим, користувач може одразу поставити додаткові запитання, наприклад: «Чим відрізняється комп’ютерна інженерія від прикладної інформатики?» або «Що вивчають на спеціальності з робототехніки?». Завдяки цьому результат тесту стає відправною точкою для подальшого діалогу та самостійного вивчення варіантів.

4.4 Застосування елементів теорії автоматичного управління у побудові адаптивної логіки тесту

Під час створення адаптивного профорієнтаційного тесту було використано принципи, характерні для теорії автоматичного управління (ТАУ). Ця дисципліна розглядає способи побудови систем, які здатні автоматично реагувати на зміну зовнішніх умов або стану об’єкта з метою досягнення бажаного результату. У контексті даного проєкту це означає побудову логіки тесту як замкненої системи з зворотним зв’язком, яка в режимі реального часу адаптує структуру та зміст подальших питань, залежно від відповідей користувача.

Адаптивність тесту реалізовано на основі так званої логіки переходів між станами, де кожен стан – це питання або блок запитань, а перехід до наступного стану визначається вихідним сигналом (тобто вибраною відповіддю). Таким чином, тест поводиться як система зі змінними коефіцієнтами передачі, які залежать від поточного контексту (вибрані відповіді, рівень зацікавленості в темі, накопичені «теги інтересів»).

Ключові аспекти, у яких проявляються ідеї ТАУ:

– зворотний зв’язок – після кожної відповіді система оновлює свій внутрішній стан (наприклад, додає або прибирає тег інтересу) і приймає рішення щодо подальшої траєкторії запитань; які мови програмування ви хотіли б вивчити?

– регулятор – умовно виконує роль логіка прийняття рішень, яка обирає найдоцільніше наступне питання з урахуванням заданих правил та вхідних сигналів; вам цікаво створення мобільних додатків, вебсайтів чи щось інше?

– цільова функція – наближення користувача до найбільш релевантної освітньої спеціальності на основі сукупності відповідей;

– стабілізація процесу – виключення повторів, нескінченних циклів або зайвих гілок тестування через обмеження по кількості питань та перевірку умов зупинки.

Застосування логіки, подібної до систем автоматичного регулювання, дозволило реалізувати гнучкий, але структурований підхід до формування тесту, що значно покращило його персоналізацію, навігацію та ефективність. Кожен користувач проходить унікальний маршрут, який залежить від його вибору, але в той же час залишається в межах контрольованої системи – це дозволяє гарантувати завершення тесту з максимально точним результатом.

Таким чином, навіть у межах гуманітарно орієнтованого застосування (профорієнтація), елементи теорії автоматичного управління знайшли практичне використання як основа для розробки адаптивного тестування нового покоління.

4.5 Охорона праці при розробці програмного забезпечення

Організація умов праці в галузі інформаційних технологій відіграє важливу роль не лише з точки зору ефективності виконання завдань, а й з огляду на збереження здоров’я та працездатності розробника. Одним із ключових факторів, що впливає на самопочуття та продуктивність при тривалій роботі за комп’ютером, є якість освітлення робочого простору.

Під час реалізації дипломного проєкту розробка вебсайту здійснювалася в умовах, що задовольняють гігієнічні вимоги до освітлення приміщень згідно з положеннями, викладеними у підручнику Основи охорони праці [29]. У посібнику визначено основні вимоги до параметрів освітленості для різних видів зорових робіт, зокрема – в офісних і навчальних приміщеннях, де виконуються завдання середньої точності, що забезпечує належний рівень безпеки та зниження втомлюваності користувача під час тривалої роботи за комп’ютером.

Згідно з вищезазначених норм, для приміщень, де відбувається робота з відеотерміналами та дисплеями, мінімальна освітленість робочої поверхні має становити не менше 300 лк при загальному освітленні, а оптимальні значення

для робіт середньої точності – до 500 лк залежно від розміру елементів зорової задачі [29].

У процесі реалізації цього проєкту програмна частина розроблялася в освітленому приміщенні з доступом до природного світла, доповненого системою штучного освітлення. Основне джерело – стельовий світильник із лампами потужністю 12 Вт (LED, колірна температура 4200–4500 K), що забезпечували рівномірне розсіювання світлового потоку без утворення тіней. Для додаткового комфорту використовувалася регульована настільна лампа зі спрямованим потоком світла, що дозволяло уникати контрастів між яскравістю екрану та навколишнім середовищем.

За допомогою цифрового люксметра було зафіксовано середній рівень освітленості на робочій поверхні – 430 люкс, що перебуває у межах рекомендованих норм для зорових завдань середньої точності. Зазначене значення забезпечує достатній візуальний комфорт під час тривалого читання та написання коду, знижує ризики перенапруження очей і пов’язаних з цим проблем зі здоров’ям.

З урахуванням нормативів та умов, у яких проводилася розробка, можна зробити висновок, що організація робочого місця була виконана відповідально та відповідала базовим вимогам охорони праці у сфері ІТ. Дотримання санітарно-гігієнічних стандартів сприяло збереженню концентрації, стабільному виконанню завдань і мінімізації негативного впливу на фізичний стан розробника.

Таким чином, середовище, в якому здійснювалася робота над проєктом, можна вважати безпечним, комфортним і таким, що забезпечувало належні умови для виконання інтелектуальної праці відповідно до чинного законодавства та галузевих норм.

ВИСНОВКИ

У межах дипломної роботи було повністю опрацьовано концепцію, структуру та функціональну реалізацію вебсайту-путівника для абітурієнтів Харківського національного університету радіоелектроніки. Основною метою стало створення зручного та сучасного інструменту, який би не тільки знайомив потенційних студентів з університетом, а й допомагав орієнтуватися у виборі майбутньої спеціальності відповідно до власних інтересів, схильностей та особистісних якостей.

Важливою особливістю сайту стала інтеграція інтелектуального чат-бота, що забезпечує взаємодію у формі звичного діалогу. Це значно полегшує навігацію по інформації, особливо для користувачів, які не мають досвіду роботи з великими освітніми порталами. В основі логіки чат-бота лежить векторний пошук, реалізований за допомогою Supabase і PgVector – сучасного розширення для PostgreSQL, що дозволяє зберігати та обробляти embedding-вектори безпосередньо в базі даних. Такий підхід виключає необхідність використання зовнішніх сервісів для обробки запитів, зменшує затримки й забезпечує високу масштабованість.

З метою підвищення користувацької цінності проєкту було реалізовано адаптивний профорієнтаційний тест. Він не є стандартним набором запитань – навпаки, структура тесту динамічна: нові запитання підбираються залежно від попередніх відповідей, що дозволяє враховувати індивідуальну логіку кожного користувача. Завдяки цьому рекомендації щодо спеціальностей стають більш точними та персоналізованими. У деяких запитаннях у чаті метадані містять спеціальне поле, яке активує кнопку для проходження тесту – наприклад, коли користувач ставить запитання на кшталт «як вибрати спеціальність», «яка професія мені підходить» тощо. Це дозволяє надавати рекомендації лише тоді, коли вони справді релевантні.

Після завершення тесту користувач отримує список відповідних освітніх напрямів та пряме посилання на офіційний сайт університету для детального

ознайомлення. Такий підхід забезпечує як практичну користь, так і заклик до подальшої взаємодії з університетом.

У процесі реалізації дипломного проєкту було не лише враховано технічні вимоги, а й глибоко проаналізовано поведінку та потреби цільової аудиторії – майбутніх абітурієнтів. Створений вебсайт поєднує в собі сучасні технології, функціональність, адаптивність і простоту використання. Він забезпечує ефективну інформаційну підтримку, допомагає уникнути дезорієнтації під час вступної кампанії та сприяє формуванню виваженого професійного вибору.

У результаті, розроблене рішення може бути не тільки інструментом залучення нових студентів до ХНУРЕ, а й прикладом того, як сучасні технології можуть бути інтегровані в освітній процес для підтримки користувача на кожному етапі прийняття важливих рішень.

Крім того, дана кваліфікаційна робота сприяє досягненню Цілі сталого розвитку №4 Якісна освіта, а саме завдання 4.4 – шляхом створення інноваційного цифрового інструменту для профорієнтації та підтримки молоді у виборі освітнього шляху відповідно до її здібностей, інтересів та потенціалу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-2015. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. – 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми Системна інженерія / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В.Токарєва, А.І. Бронніков. Харків: ХНУРЕ, 2022. 66 с.

3. Лукієнко Д. В. Аналіз технологій для вебсайту-помічника абітурієнта: чому Next.JS та Google таблиці – оптимальне рішення / Д. В. Лукієнко // Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки («Computer-integrated technologies, automation and robotics» CITAR-2025): збірник студенських наукових статей. – Харків : ХНУРЕ, 2025. – С.73-75.

4. Харківський національний університет радіоелектроніки – Офіційний сайт. URL: https://nure.ua/ (дата звернення: 18.03.2025).

5. Official MIT Admissions Website – Usability Analysis, 2023. URL: https://mitsloan.mit.edu/ (дата звернення: 18.03.2025).

6. Stanford University Website, 2022. URL: https://www.stanfordtram.com/ (дата звернення: 18.03.2025).

7. Petrov, I. Chatbots in Education: Trends and Prospects, 2021. Journal of Educational Technology.

8. Nielsen Norman Group. UX Design for Educational Platforms, 2020. URL: https://www.nngroup.com/articles/ux-for-education (дата звернення: 18.03.2025).

9. Journal of Human-Computer Interaction. User Experience and Chatbots, 2019. Vol. 35, Issue 4.

10. UX Research Report. Chatbot Effectiveness in Information Retrieval, 2021. URL: https://uxresearch.com/chatbot-effectiveness (дата звернення: 18.03.2025).

11. Educational Testing Service. Adaptive Testing Systems, 2018. URL: https://www.ets.org/research/adaptive-testing (дата звернення: 18.03.2025).

12. OECD Reports. Innovations in Career Guidance, 2021. URL: https://www.oecd.org/education/career-guidance-innovations (дата звернення: 18.03.2025).

13. Smith, J. Adaptive Learning and Personalization, 2020. Educational Technology Review, Vol. 12, No. 2.

14. W3C. Multilingual Web Design Principles, 2022. URL: https://www.w3.org/International/articles/multilingual-web (дата звернення: 18.03.2025).

15. CMS Journal. Content Management in Education, 2020. Vol. 8, Issue 1.

16. Google Developers. Responsive Web Design Basics, 2023. URL: https://developers.google.com/web/fundamentals/design-and-ux/responsive (дата звернення: 18.03.2025).

17. Документація Next.js – Створення Next-додатку (Create Next App). URL: https://nextjs.org/docs/app/create-next-app (дата звернення: 19.03.2025).

18. Документація Tailwind CSS – Директива @apply. URL: https://tailwindcss.com/docs/functions-and-directives#apply (дата звернення: 2.04.2025).

19. Документація Tailwind CSS – Темна тема (Dark Mode). URL: https://tailwindcss.com/docs/dark-mode (дата звернення: 2.04.2025).

20. Документація next-intl – Підключення мультимовності та URL-локалізація. URL: https://next-intl.dev/docs (дата звернення: 12.04.2025).

21. Офіційна документація next-intl – Налаштування маршрутизації з підтримкою i18n у App Router. URL: https://next-intl.dev/docs/getting-started/app-router/with-i18n-routing (дата звернення: 13.04.2025).

22. Офіційна документація Supabase – Початок роботи з аутентифікацією та клієнтом. URL: https://supabase.com/docs/guides/auth (дата звернення: 4.05.2025).

23. Офіційна документація Supabase – Використання клієнта у Next.js та захищені маршрути. URL: https://supabase.com/docs/guides/with-nextjs (дата звернення: 4.05.2025).

24. OpenAI – Embeddings API Documentation URL: https://platform.openai.com/docs/guides/embeddings (дата звернення: 4.05.2025).

25. OpenAI – Модель text-embedding-3-large URL: https://platform.openai.com/docs/models/embedding-models (дата звернення: 4.05.2025).

26. Supabase – Документація PgVector (векторний пошук) URL: https://supabase.com/docs/guides/database/pgvector (дата звернення: 4.05.2025).

27. PostgreSQL Extension PgVector – GitHub репозиторій URL: https://github.com/pgvector/pgvector (дата звернення: 5.05.2025).

28. Supabase – Документація по інтеграції бази даних із векторним пошуком URL: https://supabase.com/docs/guides/ai/examples/headless-vector-search?utm_source=chatgpt.com (дата звернення: 6.05.2025).

29. Власюк А. І., Ткаченко С. В. Основи охорони праці: підручник. – Київ: Центр учбової літератури, 2021.