

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»

УДК 004.43

До захисту допущено:

Завідувач кафедри

_____Олександр РОЛІК

«____» _____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою

«Інформаційні управляючі системи та технології»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Веб-орієнтована система обміну навичками та знаннями»

Виконав:

Студент 2 курсу, групи ІС-32 мп

Тарасенко Роман Юрійович _____

Керівник:

доцент кафедри , к.т.н., доцент,

Цьопа Наталія Володимирівна _____

Рецензент:

Завідувач кафедри інформаційних систем та

технологій ФІТ КНУ ім. Тараса Шевченка, д.т.н.

професор,

Дружинін Володимир Анатолійович _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___» _____ 2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Тарасенку Роману

1. Тема дисертації «Веб-орієнтована система обміну навичками та знаннями», науковий керівник дисертації Цьопа Наталія Володимирівна, к.т.н., доцент, затверджені наказом по університету від «08» 11 2024 р. № 5016-с
2. Термін подання студентом проєкту «09» 12 2024 р.
3. Об'єкт дослідження: процес обміну навичками та знаннями за допомогою веб-орієнтованих систем.
4. Предмет дослідження: методи та засоби реалізації веб-орієнтованої системи, яка забезпечує ефективний обмін навичками та знаннями між користувачами.
5. Вихідні дані до проєкту: Веб-орієнтована система, яка має забезпечити легкий доступ користувачів до вивчення нових навичок; інтуїтивно зрозумілий інтерфейс для навігації користувачів з різними технологічними навичками; впровадження заходів безпеки для захисту даних користувачів та запобігання несанкціонованому доступу; можливість підбору навичок, який заснований на рейтингу користувачів; особистий кабінет користувача

з доступом до вподобаних навичок, надісланих та отриманих запитів на обмін; простий зворотній зв'язок з адміністрацією.

6. Перелік завдань, які потрібно розробити: провести огляд існуючих програмних рішень, визначити їх переваги та недоліки; розробка сценаріїв використання системи; розробка структурної схеми, обґрунтування вибору елементів та технологій; розробка математичної складової для роботи системи рейтингу; проєктування бази даних; розробка серверної та клієнтської частин; аналіз спроектованого інтерфейсу; реалізація адаптивності веб-системи; проведення тестування на критичні помилки роботи; розроблення стартап-проєкту.

7. Перелік обов'язкового графічного матеріалу: Додаток А – Діаграма варіантів використання (А3); Додатки Б, В – Діаграми послідовностей (А3); Додаток Г – Діаграма активності (А3); Додаток Д – Діаграма станів (А3); Додаток Е – Діаграма компонентів (А3); Додаток Ж – Діаграма розгортання (А3); Додаток И – Діаграма бізнес-процесів. Користувач (А3); Додаток К – Діаграма бізнес-процесів. Робота в кабінеті (А3); Додаток Л – Діаграма бізнес-процесів. Перегляд навиків (А3); Додаток М – Діаграма бізнес-процесів. Додавання нового навика (А3).

8. Орієнтовний перелік публікацій: не заплановано.

9. Дата видачі завдання 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.09.2024	
2.	Розроблення та узгодження технічного завдання	27.01.2024	
3.	Розроблення структури веб-орієнтованої системи, дизайну сторінок	01.10.2024	
4.	Підготовка матеріалів першого, другого та третього розділів дипломного проєкту	11.10.2024	
5.	Програмна реалізація веб-орієнтованої системи. Тестування системи	18.10.2024	
6.	Підготовка матеріалів четвертого, п'ятого та шостого розділів дипломного проєкту	02.11.2024	
7.	Підготовка матеріалів сьомого та восьмого розділу. Підготовка графічної частини дипломного проєкту	17.11.2024	

Студент

Науковий керівник

Роман ТАРАСЕНКО

Наталія ЦЬОПА

РЕФЕРАТ

Веб-орієнтована система обміну навичками та знаннями: 122 с., 75 табл., 51 рис., 11 дод., 24 джерела.

ER-ДІАГРАМА, NODE.JS, БАЗА ДАНИХ, ВЕБ-ОРІЄНТОВАНА СИСТЕМА, JAVASCRIPT, VUE.JS, MYSQL.

У даній магістерській дисертації представлена розробка веб-додатку, який забезпечує новий сучасний підхід до навчання та дозволяє користувачам з різних куточків світу обмінюватись своїм досвідом в режимі онлайн. Веб-орієнтована система обміну навичками та знаннями дозволяє користувачам, які вже попередньо зареєструвались, а після – авторизувались, створити свою навичку та використати її для обміну з іншими зацікавленими учасниками. Такий підхід дозволить користувачам не лише ділитись з іншими своїм досвідом та вмінням, але й забезпечить отримання актуальних знань від справжніх фахівців. Такі знання не можна одержати з навчальної літератури та курсів, оскільки вони засновані на практичному досвіді.

Розроблена система дозволить обирати індивідуальний час та темп навчання. Додаток стане в пригоді не тільки користувачам, які вже працевлаштовані та хочуть вивчити щось нове, але й буде особливо корисний для студентів, які паралельно з формальною освітою можуть здобувати додаткові знання та розширювати свої навички. Основною перевагою запропонованого рішення є – його доступність, оскільки система не потребує фінансових витрат і її робота залежить лише від бажання користувачів вчитись новому та обмінюватись знаннями.

Метою цієї магістерської дисертації є підвищення ефективності процесу самоосвіти, за рахунок розробки системи, яка забезпечить якісний обмін навичками та знаннями між користувачами в режимі онлайн в будь-якому куточку світу.

Представлена магістерська дисертація не лише демонструє технічну реалізацію у сфері додаткової освіти, але й має практичне значення для користувачів, які шукають нові шляхи для саморозвитку та удосконалення власних умінь та знань як для професійного зростання, так і у сфері особистого розвитку.

ABSTRACT

Web-oriented system for exchanging skills and knowledge: 122 p., 75 tables, 51 figures, 11 appendices, 24 sources.

ER-DIAGRAM, NODE.JS, DATABASE, WEB-ORIENTED SYSTEM, JAVASCRIPT, VUE.JS, MYSQL.

This master's thesis presents the development of a web application that provides a new modern approach to learning and allows users from different parts of the world to share their experience online. A web-based system for exchanging skills and knowledge allows users who have previously registered and then logged in to create their own skill and use it to exchange with other interested participants. This approach will allow users not only to share their experience and skills with others, but also to obtain up-to-date knowledge from real experts. Such knowledge cannot be obtained from educational literature and courses, since it is based on practical experience. The developed system will allow you to choose your own time and pace of learning. The application will be useful not only to users who are already employed and want to learn something new, but will also be especially useful for students who, in parallel with formal education, can gain additional knowledge and expand their skills. The main advantage of the proposed solution is its accessibility, since the system does not require financial costs and its operation depends only on the desire of users to learn new things and exchange knowledge.

The purpose of this master's thesis is to increase the efficiency of the self-education process by developing a system that will ensure high-quality exchange of skills and knowledge between users online in any corner of the world.

The presented master's thesis not only demonstrates technical implementation in the field of additional education, but also has practical significance for users who are looking for new ways for self-development and improvement of their own skills and knowledge both for professional growth and in the field of personal development.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ.....	5
ВСТУП.....	6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1 Опис предметного середовища та процесу діяльності	8
1.2 Аналіз існуючих рішень та результатів дослідження.	9
1.2.1 SkillShare.....	10
1.2.2 skillharbour.....	12
1.2.3 Результати проведеного аналізу.....	14
1.3 Постановка задачі	16
Висновки до розділу 1	17
2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ	18
2.1 Функціональні вимоги.....	18
2.2 Нефункціональні вимоги	23
Висновки до розділу 2	25
3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ.....	26
Висновки до розділу 3	32
4 ПРОЄКТУВАННЯ БАЗИ ДАНИХ	33
4.3 Етапи проєктування бази даних	33
4.4 Структура бази даних.....	35
4.5 Опис таблиць бази даних	36
Висновки до розділу 4	40
5 ПРОЄКТУВАННЯ СИСТЕМИ	41
5.1 Сценарії використання системи	41
5.2 Архітектура веб-орієнтованої системи.....	49
5.2.1 Діаграма варіантів використання	49
5.2.2 Діаграми послідовності.....	50
5.2.3 Діаграма станів	50
5.2.4 Діаграма активності.....	51

5.2.5 Діаграма компонентів.....	51
5.2.6 Діаграма розгортання	52
5.3 Вибір технологій.....	52
5.3.1 Вибір мови програмування.....	52
5.3.2 Платформа для проєктування серверної частини.....	58
5.3.3 Вибір веб-фреймворків	60
5.3.4 Середовище розробки	64
5.4 Розробка серверної частини веб-системи.....	65
5.5 Розробка клієнтської частини веб-системи.....	69
Висновки до розділу 5	72
6 ІНТЕРФЕЙС КОРИСТУВАЧА ТА UX/UI ДИЗАЙН.....	74
6.1 Спроектований інтерфейс	74
6.2 Адаптивність та кросплатформенність.....	86
Висновки до розділу 6	92
7 ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМИ.....	93
7.1 Тестування функціональності	93
Висновки до розділу 7	97
8 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ	98
8.1 Опис ідеї проєкту	98
8.2. Технологічний аудит ідеї проєкту.....	100
8.3. Аналіз ринкових можливостей запуску стартап-проєкту	101
8.4. Розроблення ринкової стратегії проєкту	109
8.5. Розроблення маркетингової програми стартап-проєкту	112
Висновки до розділу 8	117
ВИСНОВКИ	119
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	121
ДОДАТКИ	123

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД – база даних;

СКБД – система керування базами даних;

ПЗ – програмне забезпечення;

CMS – система керування вмістом;

CRM – система управління взаємовідносинами з клієнтами;

ERP – система для будь-яких бізнес-потреб;

DOM – це програмний інтерфейс для веб-документів;

MVC - модель-вид-контролер (архітектурний шаблон);

SPA – веб-додаток, розміщений на одній веб-сторінці;

API – інтерфейс програмування застосунків;

MVVM – шаблон проектування, що застосовується під час проектування архітектури застосунків (додатків).

ВСТУП

У сучасному світі концепція «одна спеціальність на життя» уже не працює, оскільки професії змінюються значно швидше, ніж програма навчальних закладів. Можна значно сповільнити свій розвиток, якщо не йти шляхом прогресу, не здобувати новий досвід, навички та не вдаватись до неформальних форм освіти.

Існує думка, що мережа Інтернет – невичерпне джерело інформації, і це так. Величезні обсяги інформації не завжди дають результат у вивченні специфічних навиків, для яких важливою є практична складова. Тому розроблена веб-орієнтована система обміну навичками та знаннями спрямована на те, щоб допомогти тим користувачам, які прагнуть удосконалити не лише свої теоретичні знання, а й практичні уміння.

Актуальність обраної теми полягає в тому, що попит на самоосвіту зростає, а отже подібні системи існують, однак більшість з них платні та не надають бажаного рівня знань. Системи обміну навичками дозволяють людям вчитись будь-де та будь-коли, також такі системи значно ефективніші за платформи, які надають послуги з онлайн-курсів, оскільки орієнтовані тільки на взаємодію двох користувачів, що в подальшому прискорює та покращує навчальний процес. Окрім того, така система може забезпечити користувачам конкурентну перевагу, бо дозволяє в короткий строк освоїти нову технологію та використати її при зміні роботи.

Мета роботи – підвищення ефективності процесу самоосвіти, за рахунок розробки веб-орієнтованої системи обміну навичками та знаннями, яка забезпечить якісне та зручне навчання в режимі онлайн і дозволить не лише скоротити час навчання, але й допоможе здійснити підбір навичок відповідно до рейтингу користувачів.

Об'єктом дослідження у даній роботі виступає процес обміну навичками та знаннями за допомогою веб-орієнтованих систем.

Предметом дослідження є методи та засоби реалізації веб-орієнтованої системи, яка забезпечує ефективний обмін навичками та знаннями між

користувачами.

Для досягнення поставленої мети у процесі розробки веб-орієнтованої системи обміну навичками та знаннями, будуть використані наступні методи досліджень:

1. аналітичний (включає пошук та вивчення існуючих системи обміну навичками, їх функціональних можливостей, а також розгляд переваг та недоліків);
2. математичне моделювання;
3. програмування (розробка системи);
4. тестування (перевірка виконаних технічних умов та надійності системи).

Практична цінність розроблюваної системи полягає в наступному:

- дозволить розширити доступ до навичок та знань не залежно від географічного розташування користувача;
- прискорить процес одержання практичних навичок, за рахунок адаптивного навчання та взаємного обміну навичок;
- підвищить конкурентоспроможність на ринку праці;
- дозволить адаптуватись до непрогнозованих змін на ринку праці;
- надасть індивідуальний підхід до користувача;
- прийтиме постійному навчанню та самовдосконаленню.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметного середовища та процесу діяльності

Перед початком створення будь-якого рішення чи продукту – необхідно визначити, чи він потрібен світу та хто ним буде користуватись. Цільовою аудиторією веб-орієнтованої системи обміну навичками та знаннями будуть не тільки студенти, яким важко оплатити дорогі курси чи вебінари, але й вже повністю реалізовані працівники різноманітних компаній, які прагнуть саморозвитку та вдосконалення. Варто зазначити, що сьогодні вже існує значна кількість систем, які допомагають користувачам у самонавчанні, однак всі вони з часом перетворюються на магазин з продажу курсів, де не існує індивідуального підходу до користувача [1]. Для вирішення такої проблеми існують різного роду репетиторства та майстер-класи, однак всі вони мають спільні недоліки:

- дороговартісні;
- орієнтування на теоретичні питання проблеми і ніяк не на практичні;
- залежні від географії;
- викладачі не завжди зацікавлені в навчанні.

Перелічені проблеми сильно пригнічують мотивацію людей у навчанні, тому вирішенням цього слугуватиме розробка веб-орієнтованої системи обміну навичками та знаннями. Дана система усуватиме перелічені недоліки, оскільки ґрунтуватиметься на ентузіазмі людей. Користувачі будуть прагнути поділитись своїми здобутками у сфері розробки програмного забезпечення, вивченні іноземних мов чи навіть садоводстві, оскільки це те, що вони вже знають і вміють і при цьому вони бажають обмінюватись своїми навичками.

Таким чином, метою даної магістерської дисертації є забезпечити швидке, доступне та індивідуальне засвоєння нових навичок та знань, за рахунок розробки ефективної веб-орієнтованої системи.

Для досягнення поставленої мети необхідно виконати наступний перелік задач:

1. проаналізувати існуючі рішення;

2. провести порівняльний аналіз можливостей систем-конкурентів, дослідити їх переваги та недоліки;
3. визначити функціональні потреби користувачів такої системи;
4. створити простий та сучасний дизайн веб-системи;
5. обґрунтувати доцільність вибору стеку технологій, який буде використаний у процесі розробки веб-орієнтованої системи;
6. розробити веб-орієнтовану систему обміну навичками та знаннями;
7. провести аналіз та тестування розробленої системи.

1.2 Аналіз існуючих рішень та результатів дослідження.

Аналіз існуючих програмних рішень досить важливий підрозділ, оскільки дозволяє глибше зрозуміти проблеми, дослідити переваги та недоліки продукту конкурентів у процесі користування. Також такий аналіз допомагає розробити план, який спростить процес створення своєї програмної системи.

Аналіз програмних рішень включатиме в себе:

1. пошук подібний систем в мережі Інтернет з такими ж функціональними можливостями;
2. огляд реалізації наявних функціональних можливостей (дослідження наскільки вони ефективно забезпечують необхідні потреби користувачів);
3. визначення недоліків та переваг досліджуваних систем;
4. виявлення відсутніх функціональних можливостей, але затребуваних ринком.

Пошук подібний систем в мережі Інтернет висвітлює в більшості платформи для надання курсів чи форуми для спілкування та обміном досвіду. Поясненням цього слугує те, що якщо такі системи є, то вони непопулярні, бо їх витісняють платформи з надання курсів.

Оскільки систем з потрібними функціональними можливостями мало, то для проведення аналізу було обрано дві найбільш схожі системи: SkillShare [2] і Skills Harbour [3].

Проведення аналізу має орієнтуватись на найбільш затребуваних функціональних можливостях, до яких належить:

- можливість переглядати доступні навички ще до реєстрації та авторизації;
- можливість реєстрації та авторизації;
- наявність зміни персональних даних в особистому кабінеті;
- захищеність персональних даних;
- наявність форми для зворотного зв'язку з адміністрацією;
- можливість створювати свої навички та пропонувати їх на обмін;
- наявність різноманітної категорії навичок обміну досвідом;
- зручний та лаконічний дизайн особистого кабінету та всієї системи;
- можливість залишати коментарі;
- можливість переглядати рейтинг користувачів та на основі нього здійснювати підбір навичок.

1.2.1 SkillShare

SkillShare – це платформа для онлайн навчання та освіти. Дана система допомагає студентам та всім охочим вивчити різноманітні навички, якими можна буде скористатись в подальшій кар'єрі чи навіть особистому житті. SkillShare фокусується на справжньому різноманітті, не зациклюючись тільки на деяких темах. На рис. 1.1 продемонстровано фрагмент головної сторінки платформа SkillShare.

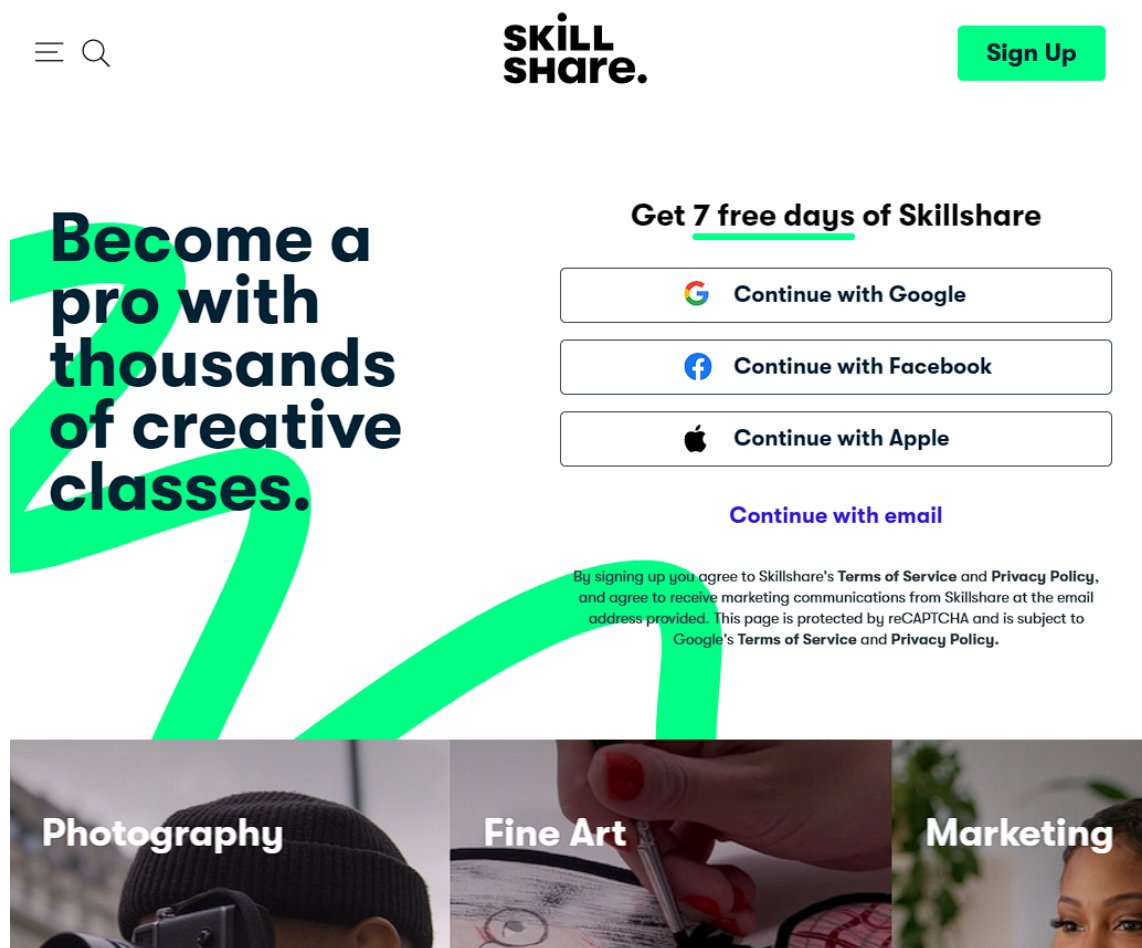


Рисунок 1.1 – Фрагмент головної сторінки SkillShare

Прогорнувши сторінку вниз, можна побачити категорії доступних курсів для навчання. Головна сторінка виглядає гарно та багатообіцяюче, зразу розкриває всю інформаційну складову платформи. Курси представляються за допомогою відеоматеріалів. Частина курсів виконана професійно, але до іншої частини є претензії в користувачів. Основними недоліками називають те, що більшість курсів орієнтовані на початківців, хоч і складаються не тільки з теоретичних лекційних матеріалів. Варто зазначити, що важливою особливістю даної платформи є робота над проєктом у процесі вивчення курсу. Отже значною мірою приділяється увага закріпленню теорії на практиці.

Головним недоліком, який найбільше дратує користувачів SkillShare, – є цінова політика платформи. На цій платформі можна купити підписку і вона дає доступ до величезної колекції уроків, однак така підписка невинуватна, оскільки користувач хоче зосередитись на одному напрямі навчання для вивчення

конкретної навички. І знову ж таки дана платформа працює на продажі курсів, а у такій системі не завжди отримується якісне навчання, бо не всі зацікавлені в цьому.

Отже, до основних переваг системи онлайн навчання SkillShare належить:

- велика кількість різноманітних категорій, де можна знайти необхідний навик;
- реєстрація та авторизація;
- зручний та лаконічний дизайн;
- багатофункціональний особистий кабінет;
- у процесі навчання наявне засвоєння практичних навичок (залежить від курсу);
- різні формати навчання, під які можна з легкістю підлаштовуватись.

До недоліків даної системи можна віднести:

- безплатні курси присутні, але їх дуже мало і вони неефективні;
- не всі курси надають практичні навички;
- більшість курсів корисні тільки для новачків;
- платформа надає тільки обмежену інформацію про курси та отримані навички у процесі навчання;
- цінова політика;
- дана система знову ж таки працює на продаж курсів та надає обмежені можливості для обміну навичками та досвідом;

Аналіз показав, що платформа для онлайн навчання має свої переваги тільки для новачків, які хочуть спробувати себе в новій сфері та отримати базові навички. Важливим недоліком цієї системи є цінова політика та орієнтованість на продаж курсів, такий напрям роботи платформи шкодить якісному навчальному процесу.

1.2.2 Skillharbour

Skillharbour – це система, за допомогою якої можна здійснювати обмін навичками чи талантами без фінансових витрат. Фрагмент головної сторінки платформи Skillharbour представлено на рис. 1.2.

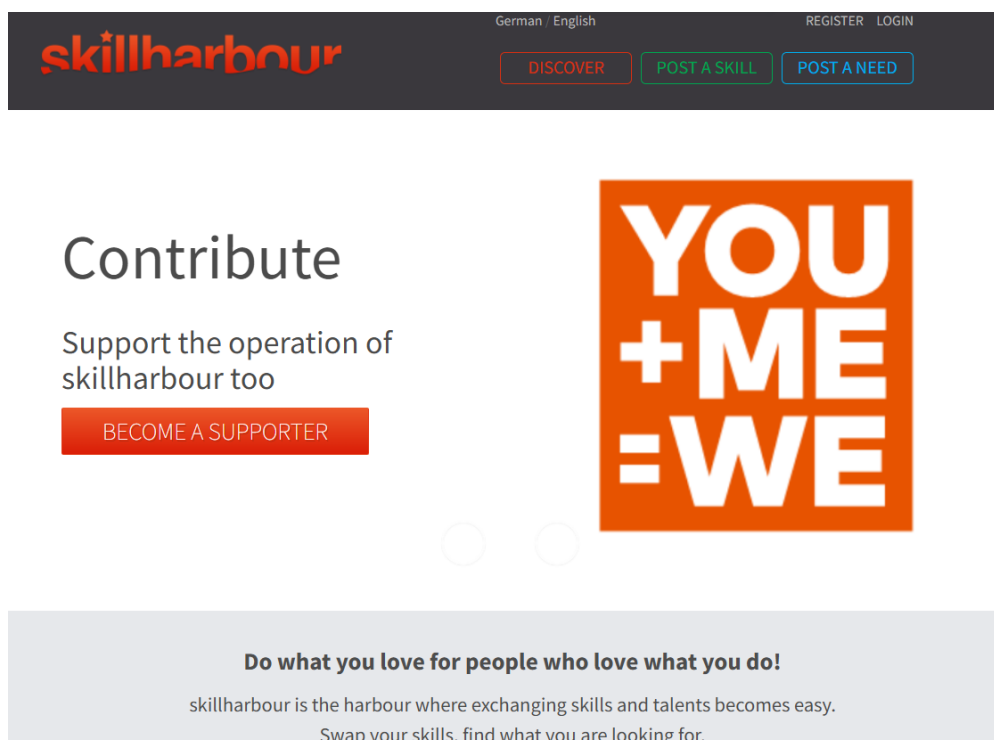


Рисунок 1.2 – Фрагмент головної сторінки Skillharbour

На цій платформі учасники можуть обмінювати свій час на здобуття потрібних знань. Робота системи виглядає наступним чином: користувач витрачає годину часу, щоб допомогти іншому користувачу та в заміन отримує кредитну годину. Далі цей кредит використовується для отримання послуг від інших учасників, які пропонують свої навички. Щоб мати змогу створювати навички та шукати необхідні – потрібно зареєструватись та заповнити профіль своєю інформацією. Незважаючи на мінімально реалізовані функціональні можливості, система має важливі недоліки – немає можливості оцінити користувачів чи переглянути сторінки з відгуками про обмін навичками, який вже відбувався. Також великою проблемою є слабка безпека сайту, оскільки він не має чинного SSL-сертифікату. Відсутність такого сертифікату створює ризики для користувачів не тільки в процесі реєстрації та заповненні особистого профілю, а й процесі обміну навичками та навчанні.

Отже, до переваг платформи Skillharbour можна віднести:

- реалізовані мінімальні функціональні можливості платформи;

- можливість реєстрації та авторизації;
- наявність зміни персональних даних в особистому кабінеті;
- можливість створювати свої навички та пропонувати їх на обмін;
- наявність різноманітної категорії навичок обміну досвідом.

З першого погляду проаналізована платформа забезпечує мінімально необхідні функціональні можливості, однак існують важливі недоліки, через які не тільки незручно використовувати систему, а й небезпечно:

- відсутній захист персональних даних;
- застарілий дизайн платформи;
- відсутня можливість залишати коментарі;
- відсутня можливість переглядати рейтинг користувачів;
- немає можливості для забезпечення зворотного зв'язку з адміністрацією.

1.2.3 Результати проведеного аналізу

У ході аналізу існуючих рішень було розглянуто дві платформи SkillShare та Skillharbour, проаналізовано їх функціональні можливості, розкрито переваги та недоліки.

Варто зазначити, що система SkillShare хоч і відноситься до системи обміну знаннями, але вона більше орієнтована на продаж курсів. Система Skillharbour має функції для обміну навичками, які певно мірою працюють. Ці дві системи об'єднують те, що вони допомагають користувачам отримувати нові навички та досвід, вони виконують мінімальні функціональні можливості, а саме:

- можна зареєструватись та авторизуватись;
- можливість керувати обліковим записом то його інформацією;
- можна вибрати курс/запропонувати навичку на обмін.

Але варто сфокусуватись на їх недоліках: наприклад, SkillShare – це платформа, яка надає платні курси, вона не забезпечить належний рівень отримання нових навичок, оскільки не всі зацікавлені в цьому процесі. Також недоліком є те, що платформа хоч і має безплатні курси, але їх дуже мало та

ефективність наближається до нуля. Skillharbour хоч і безплатна система, однак такі недоліки як: відсутність зворотного зв'язку, прогалини в захисті системи, застарілий та незручний дизайн, відсутня можливість переглядання рейтингу, коментарів – є критичним. Для підведення підсумку у табл. 1.1 наведено необхідні функціональні можливості та наявність їх у проаналізованих існуючих рішеннях.

Таблиця 1.1 – Аналіз функціональних можливостей систем

Затребувані функціональні можливості	Платформу обміну навичками та знаннями	
	SkillShare	Skillharbour
Реєстрація та авторизація	+	+
Наявність особистого кабінету	+	+
Керування персональною інформацією в особистому кабінеті	+	+
Сучасний дизайн	+	+/-
Лаконічне наповнення платформи	+	-
Швидкодія	+/-	-
Захищеність персональних даних	+/-	-
Наявність форми для зворотного зв'язку з адміністрацією	+/-	-
Наявність різноманітної категорії навичок обміну досвідом	+/-	-
Можливість залишати коментарі	+/-	-
Можливість переглядати рейтинг користувачів	+/-	-
На безкоштовній основі	-	+
Наявність різноманітної категорії навичок обміну досвідом	+/-	-

Проаналізувавши табл. 1.1 можна дійти висновку, що розглянуті системи не забезпечують всіх необхідних функціональних можливостей, тому прийнято рішення розробити нову веб-орієнтовану систему обміну навичками та знаннями, яка повноцінно реалізує найбільш затребувані функціональні можливості та надасть змогу швидко та зручно обмінюватись навичками та досвідом.

1.3 Постановка задачі

Метою даної магістерської дисертації є підвищення ефективності процесу самоосвіти, за рахунок розробки веб-орієнтованої системи обміну навичками та знаннями. Така система допоможе студентам та людям, які прагнуть самовдосконалення, спробувати себе в найрізноманітніших галузях та здобути нових вершин. Створена система має врахувати всі сучасні тенденції, які застосовують при проектуванні веб-додатків, та включатиме виконання таких задач:

- створення функції реєстрації;
- створення функції авторизації;
- створення особистого кабінету
- наявність функції управління даними в особистому кабінеті;
- створення функції пошуку та фільтрів;
- наявність функції створення навиків;
- наявність функції, яка забезпечить обмін навиком;
- наявність функції вибору категорій;
- створення полів для коментарів та форми для зворотного зв'язку;
- наявність можливості переглядати рейтинг та брати участь у його формуванні.

Висновки до розділу 1

У процесі написання першого розділу магістерської дисертації проведено аналіз предметного середовища. Процес аналізу визначив цільову аудиторію веб-орієнтованої системи, типи навичок та знань, які найбільш затребувані, та формат взаємодії між користувачами.

Також в даному розділі було розглянуто існуючі програмні рішення. Даний етап дозволяє глибше зрозуміти проблеми, дослідити переваги та недоліки продукту конкурентів у процесі користування. Для аналізу було обрано дві найбільш схожі системи: SkillShare і skillharbour. Проведення аналізу базувалось на найбільш затребуваних функціональних можливостях, огляді переваг та недоліків. Результатом такого аналізу послугувало розроблення порівняльної таблиці, за допомогою якої було оцінено рівень реалізації всіх затребуваних функцій. У процесі аналізу поставлено задачі, які мають бути вирішені при проєктуванні веб-орієнтованої системи обміну навичками та знаннями.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Функціональні вимоги

Функціональні вимоги використовують для пояснення того, як саме система повинна функціонувати. Ці вимоги визначають, що саме повинна робити система, щоб забезпечити потреби користувача. Функціональні вимоги – це свого роду функції, які помічає користувач в ході використання системи. Зазвичай функціональні вимоги складаються з таких частин як функції та поведінки. Функція визначає те, що виконує система, а поведінка – яким чином це виконує система [4].

Функціональні вимоги до веб-орієнтованої системи обміну навичками та знаннями представлено в табл. 2.1-2.19.

Таблиця 2.1 – Характеристика функціональної вимоги FR001

Код	FR001
Назва	Можливість здійснити реєстрацію в системі
Характеристика	Користувач має змогу здійснити реєстрацію за допомогою введення таких даних як: електронна пошта, пароль та ПІБ.

Таблиця 2.2 – Характеристика функціональної вимоги FR002

Код	FR002
Назва	Можливість здійснити авторизацію в системі
Характеристика	Користувач, який вже зареєструвався, має змогу здійснити автентифікацію за допомогою введення таких даних як: електронна пошта (або логін) та пароль.

Таблиця 2.3 – Характеристика функціональної вимоги FR003

Код	FR003
Назва	Можливість здійснити вихід з системи

Код	FR003
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу здійснити вихід з системи за допомогою натискання відповідної кнопки.

Таблиця 2.4 – Характеристика функціональної вимоги FR004

Код	FR004
Назва	Можливість редагувати особисті дані у системі
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу здійснити додавання, видалення та редагування персональних даних.

Таблиця 2.5 – Характеристика функціональної вимоги FR005

Код	FR005
Назва	Можливість створювати список улюблених навичок у системі
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу здійснити додавання вподобаної навички у свій список.

Таблиця 2.6 – Характеристика функціональної вимоги FR006

Код	FR006
Назва	Можливість встановити фото профілю у системі
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу здійснити завантаження фотографії для свого профілю.

Таблиця 2.7 – Характеристика функціональної вимоги FR007

Код	FR007
Назва	Можливість переглянути сторінку із запропонованими навичками в системі
Характеристика	Користувач (навіть неавторизований) має змогу переглянути сторінку із запропонованими на обмін навичками.

Таблиця 2.8 – Характеристика функціональної вимоги FR008

Код	FR008
Назва	Можливість переглянути конкретну виставлену навичку
Характеристика	Користувач (навіть неавторизований) має змогу переглянути детальний опис виставленої навички на обмін.

Таблиця 2.9 – Характеристика функціональної вимоги FR009

Код	FR009
Назва	Можливість створювати власні навички
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу створити власні навички.

Таблиця 2.10 – Характеристика функціональної вимоги FR010

Код	FR010
Назва	Можливість редагувати власні навички
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу редагувати свої навички. Присутня можливість змінювати опис, назву навички та здібності.

Таблиця 2.11 – Характеристика функціональної вимоги FR011

Код	FR011
Назва	Можливість видаляти власні навички

Код	FR011
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу видалити свої навички.

Таблиця 2.12 – Характеристика функціональної вимоги FR012

Код	FR012
Назва	Можливість пропонувати власну навичку на обмін
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу вибрати вподобану навичку та запропонувати власну на обмін.

Таблиця 2.13 – Характеристика функціональної вимоги FR013

Код	FR013
Назва	Можливість переглядати запропоновані навички на обмін
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу переглядати надіслані заявки на обмін навичками.

Таблиця 2.14 – Характеристика функціональної вимоги FR014

Код	FR014
Назва	Можливість переглядати одержані запити на обмін навичок
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу переглядати одержані заявки на обмін навичками.

Таблиця 2.15 – Характеристика функціональної вимоги FR015

Код	FR015
Назва	Можливість переглядати залишені відгуки інших користувачів

Код	FR015
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу переглядати відгуки, які залишили інші користувачів після обміну навичками.

Таблиця 2.16 – Характеристика функціональної вимоги FR016

Код	FR017
Назва	Можливість переглядати залишені відгуки інших користувачів
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу переглядати відгуки, які залишили інші користувачів після обміну навичками.

Таблиця 2.17 – Характеристика функціональної вимоги FR017

Код	FR017
Назва	Можливість залишати відгуки після обміну навичками
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу залишити відгук після завершення процесу обміну навичками та знаннями.

Таблиця 2.18 – Характеристика функціональної вимоги FR018

Код	FR018
Назва	Можливість редагувати свої відгуки
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу редагувати відгук після його написання.

Таблиця 2.19 – Характеристика функціональної вимоги FR019

Код	FR019
Назва	Можливість видаляти свої відгуки

Код	FR019
Характеристика	Користувач, який вже зареєструвався та увійшов в обліковий запис, має змогу видаляти власні відгуки після їх написання.

2.2 Нефункціональні вимоги

Нефункціональні вимоги – це набір стандартів, які застосовують для оцінки роботи певної системи. Дані вимоги є теж достатньо важливими, оскільки забезпечують високу ефективність роботи створеної програмної системи [5].

До переваг нефункціональних вимог належить:

- забезпечують надійність та продуктивність системи;
- покращують взаємодію користувача зі створеною системою;
- допомагають сформуванню політики безпеки системи.

Нефункціональні вимоги до веб-орієнтованої системи обміну навичками та знаннями представлено в табл. 2.20-2.26.

Таблиця 2.20 – Характеристика нефункціональної вимоги NFR001

Код	NFR001
Назва	Підтримка великої кількості користувачів
Характеристика	Створена система має підтримувати до 4 тис. користувачів на годину.

Таблиця 2.21 – Характеристика нефункціональної вимоги NFR002

Код	NFR002
Назва	Час відповіді цільової сторінки системи
Характеристика	Створена система має відображати дані не пізніше 5 секунд після завантаження (в тому числі через мобільний інтернет 4G).

Таблиця 2.22 – Характеристика нефункціональної вимоги NFR003

Код	NFR003
Назва	Сумісність браузерів
Характеристика	Створена система має працювати у всіх основних сучасних веб-браузерах (Google Chrome, Firefox, Opera, Edge та Safari).

Таблиця 2.23 – Характеристика нефункціональної вимоги NFR004

№	NFR004
Назва	Сумісність операційних систем
Характеристика	Створена система має справно працювати на всіх основних сучасних версіях операційних систем (Windows 10/11, macOS 14+, останні дистрибутиви Linux, Android 10+ та IOS 12+).

Таблиця 2.24 – Характеристика нефункціональної вимоги NFR005

№	NFR005
Назва	Вимоги до безпеки
Характеристика	Всі паролі мають бути у захешованому вигляді md5.

Таблиця 2.25 – Характеристика нефункціональної вимоги NFR006

№	NFR006
Назва	Вимоги до доступності
Характеристика	Система має швидко завантажувати веб-сторінки навіть при нестабільному Інтернет-підключенні.

Таблиця 2.26 – Характеристика нефункціональної вимоги NFR007

№	NFR007
Назва	Вимоги до дизайну
Характеристика	Система повинна мати стандартизований дизайн та бути зрозумілою у керуванні.

Висновки до розділу 2

У процесі написання цього розділу було сформовано вимоги до розробленої системи. Функціональні вимоги складають базу у визначенні функціональних можливостей веб-орієнтованої системи та включають усі дії з системою, які користувач може виконати. Такі вимоги було структуровано та наведено у ході написання даного розділу.

Також не менше важливими є нефункціональні вимоги, оскільки вони великою мірою відповідають за забезпечення якості та продуктивності системи. Розроблені функціональні вимоги врахували те, що система має витримувати значне навантаження, має забезпечувати швидкий відгук на будь-які дії клієнта, працювати при нестабільному Інтернет-з'єднанні та коректно відображатись на основних сучасний операційний системах чи веб-браузерах.

Отже, наведені функціональні та нефункціональні вимоги дозволили задати орієнтири, за якими має слідувати та на які має орієнтуватись розробник веб-орієнтованої системи обміну навичками та знаннями.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Концепція рейтингових систем у веб-орієнтованих платформах

Рейтингові системи стали важливим компонентом веб-орієнтованих платформ, оскільки забезпечують прозору взаємодію між користувачами, які можуть не мати попереднього досвіду спілкування один із одним. Потреба в рейтингових системах виникла з розвитком перших платформ електронної комерції, таких як eBay у 1995 році. Саме eBay вперше реалізував просту систему рейтингів, яка дозволяла користувачам залишати позитивні, нейтральні чи негативні відгуки після завершення угоди. Такий підхід заклав основу для створення довіри між покупцями і продавцями.

З часом рейтингові системи знайшли застосування на платформах для обміну знаннями та навичками (Coursera, Udemy), соціальних мережах (Reddit, YouTube) та навіть у транспортних послугах (Uber, BlaBlaCar). Їх головне завдання — створення механізму зворотного зв'язку, який одночасно стимулює якісну поведінку та дозволяє іншим користувачам оцінювати надійність і професійність учасників. Однією з перших вдосконалених моделей для обчислення рейтингу стала система зваженого середнього. Ця модель дозволяє враховувати важливість і складність внеску користувача, забезпечуючи більш справедливий підхід. Наприклад, на платформах, де відгуки чи оцінки можуть значно варіюватися за значенням і значимістю (навчальні курси, проекти з фрілансу), зважене середнє допомагає уникнути впливу занадто високих чи низьких оцінок, які можуть викривити загальний результат. Розробка таких моделей стала результатом досліджень у галузях статистики, поведінкової економіки та користувацького досвіду (UX). Зокрема, використання вагових коефіцієнтів почалося з оцінювання студентів у навчальних системах, а потім адаптувалося для потреб комерційних платформ. Ідея вагового підходу була заснована на принципах, запропонованих у роботах з аналізу статистичних даних XX століття.

Рейтингові системи є ключовим елементом багатьох сучасних веб-орієнтованих платформ, які дозволяють забезпечити ефективну взаємодію між

користувачами. Вони забезпечують прозорість, об'єктивність та довіру, що є важливим фактором для онлайн-платформ, таких як сервіси обміну послугами, навчальні платформи чи транспортні системи. Застосування рейтингових систем почалося з перших онлайн-платформ, зокрема таких як eBay, де рейтинг дозволяв оцінювати надійність продавців та покупців. Це стало базовим механізмом для створення репутації та впровадження зворотного зв'язку, який мотивує учасників взаємодії підтримувати високий рівень обслуговування чи надання послуг.

Ефективна рейтингова система повинна дотримуватися певних принципів, які забезпечують її якісне функціонування. Одним із ключових принципів є прозорість, коли користувачі мають чітке розуміння того, як формується їхній рейтинг та які дії можуть вплинути на його покращення. Це дозволяє мінімізувати недовіру до системи та виключає ризик маніпуляцій. Іншим важливим аспектом є справедливість, яка забезпечує об'єктивне відображення внеску кожного учасника. Наприклад, використання зваженого середнього дозволяє врахувати складність чи важливість виконаних завдань, що надає більш справедливу оцінку результатів. Також важливо, щоб рейтингові системи стимулювали активність учасників, пропонуючи їм додаткові можливості чи переваги за високі оцінки, що в свою чергу мотивує їх брати участь у складніших та значущих взаємодіях. Загалом, такі підходи сприяють не лише підвищенню ефективності платформи, але й формуванню довготривалих зв'язків між її користувачами.

3.2 Математичні основи моделі зваженого рейтингу

Математичні основи моделі зваженого рейтингу ґрунтуються на принципах статистики, зокрема на обчисленні зваженого середнього. Ця методика дозволяє забезпечити справедливість та об'єктивність у рейтингових системах, враховуючи різну важливість або складність оцінюваних взаємодій. У контексті веб-орієнтованих платформ модель зваженого рейтингу дозволяє визначати не лише загальний показник активності чи якості користувача, але й інтегрувати в оцінку додаткові фактори, такі як складність завдань або рівень компетенцій. Зважене

середнє є фундаментом цієї моделі. Воно розраховується за формулою:

$$R = \frac{\sum_{i=1}^n w_i \cdot r_i}{\sum_{i=1}^n w_i}, \quad (3.1)$$

де R — це загальний рейтинг користувача, r_i — окремі оцінки за взаємодії, w_i — вагові коефіцієнти, які визначають важливість кожної оцінки, а n — загальна кількість взаємодій. Цей підхід дозволяє відобразити вклад кожного результату у підсумкову оцінку залежно від його значущості.

Роль вагових коефіцієнтів у моделюванні рейтингу є надзвичайно важливою. У стандартних умовах усі оцінки мають однакову вагу, однак у системах, що враховують складність або важливість завдання, вагові коефіцієнти дозволяють змістити акцент на значущі взаємодії. Наприклад, виконання складного завдання з високою оцінкою матиме більший вплив на підсумковий рейтинг, ніж виконання простого завдання. Таким чином, ваги слугують інструментом диференціації внеску користувача, що стимулює його до участі у складніших або важливіших проєктах. Ключовими змінними моделі є ваги, оцінки та кількість взаємодій. Ваги визначають значущість кожного обміну чи завдання; оцінки відображають якість виконання; кількість взаємодій дозволяє оцінити активність користувача. Співвідношення цих параметрів визначає баланс між якістю та кількістю виконаних завдань. Наприклад, користувач із великою кількістю середніх оцінок може мати нижчий рейтинг, ніж користувач із меншою кількістю, але вищими оцінками, якщо модель надає більшу вагу якості. Ця методика має широкий спектр застосування, зокрема на платформах обміну знаннями, у системах відгуків та рейтингах спеціалістів. Обчислення зваженого середнього дозволяє уникнути викривлення даних через вплив поодиноких оцінок або надмірну активність у малозначущих взаємодіях, що робить модель особливо ефективною для аналізу довготривалої активності користувачів.

3.3 Застосування вагових коефіцієнтів у рейтингових системах

Вагові коефіцієнти відіграють ключову роль у створенні об'єктивних і справедливих рейтингових систем. Їхнє використання дає змогу враховувати різну важливість або складність завдань чи взаємодій, забезпечуючи точніше оцінювання внеску кожного користувача. Важливість або складність взаємодії визначається за кількома критеріями. Зокрема, більш складні завдання, що потребують значних зусиль чи високого рівня компетенції, мають отримувати вищі ваги, ніж прості та менш значущі. Вплив ваг також залежить від таких параметрів, як час, витрачений на виконання завдання, рівень знань або навичок, потрібних для його реалізації, а також від масштабності результату для всієї спільноти. Використання ваг дозволяє уникнути викривлень у системі рейтингу. Наприклад, користувач, який виконав декілька складних завдань, може мати вищий рейтинг, ніж той, хто виконав велику кількість простих. Це стимулює учасників платформи до виконання важливих завдань, які мають більший вплив на розвиток спільноти чи проєкту, забезпечуючи справедливий розподіл оцінок і визнання їхнього внеску.

Налаштування ваг у рейтингових системах може здійснюватися різними способами. Один із методів передбачає використання експертної оцінки, коли ваги визначаються фахівцями у відповідній галузі. Інший спосіб базується на аналізі даних, що дає змогу використовувати статистичні моделі для визначення значущості взаємодій. Також популярним є залучення самих користувачів через опитування чи зворотний зв'язок для оцінки складності й важливості завдань. Сучасні платформи дедалі частіше застосовують адаптивні алгоритми, які динамічно змінюють ваги на основі поведінки користувачів та результатів їхньої діяльності. Вагові коефіцієнти у рейтингових системах є необхідним інструментом для забезпечення збалансованого й прозорого оцінювання, а також для мотивації користувачів до якісної участі. Їхнє правильне застосування гарантує ефективність платформи та формує довіру серед її учасників.

3.4 Стимулюючий ефект рейтингових систем для учасників обміну

Рейтингові системи є потужним інструментом стимулювання активності та побудови довіри серед учасників веб-орієнтованих платформ. Їх вплив базується на психологічних та соціальних факторах, які спонукають користувачів до більш активної участі у взаємодіях, зокрема у складних і значущих обмінах. Один із ключових механізмів впливу рейтингових систем — мотивація користувачів. Високий рейтинг слугує ознакою компетентності та надійності, що, своєю чергою, підвищує довіру до користувача з боку інших учасників платформи. Для багатьох це стає стимулом брати участь у більш складних обмінах, які, попри більші витрати часу та зусиль, сприяють підвищенню їхнього рейтингу. Такий підхід забезпечує природне прагнення до розвитку навичок, виконання якісніших завдань і, як наслідок, зміцнення репутації.

Важливим аспектом є роль рейтингових систем у формуванні довіри між користувачами. У платформах, де взаємодія відбувається між незнайомими людьми, репутація відіграє вирішальну роль. Рейтинг, заснований на попередньому досвіді взаємодій, виступає індикатором надійності, допомагаючи учасникам приймати обґрунтовані рішення. Наприклад, у системах обміну послугами або знаннями високий рейтинг виконавця збільшує ймовірність його вибору для виконання завдань, формуючи довготривалі відносини довіри. Попри очевидні переваги рейтингових систем, вони не позбавлені ризиків зловживання. Серед поширених викликів — необ'єктивні оцінки, маніпуляції з рейтингами або спроби накручування показників. Для уникнення таких проблем платформи впроваджують різні механізми захисту, зокрема автоматичний аналіз аномальної активності, перевірку даних вручну, а також введення чітких правил взаємодії. Крім того, використання багатофакторних моделей рейтингу, де враховуються як кількісні, так і якісні аспекти, допомагає зробити систему стійкішою до маніпуляцій. Загалом, рейтингові системи виконують дві важливі функції: стимулювання користувачів до активної та якісної участі, а також забезпечення довіри та прозорості у взаємодії. Їх ефективне налаштування дозволяє знизити ризики

зловживань і підтримувати здоровий баланс між інтересами учасників платформи.

3.5 Перспективи розвитку моделі рейтингу для веб-орієнтованих систем

Необхідно зазначити, що модель рейтингу є динамічним інструментом, який постійно вдосконалюється відповідно до потреб і викликів сучасних веб-орієнтованих систем. Перспективи її розвитку охоплюють інтеграцію нових змінних, адаптацію до потреб різних груп користувачів і вдосконалення механізмів зворотного зв'язку для оцінки ефективності. Одним із напрямків розвитку є інтеграція додаткових змінних у модель рейтингу. Традиційні рейтингові системи зазвичай базуються на кількісних оцінках, однак врахування таких параметрів, як час виконання завдання, якість відгуків або рівень складності взаємодії, може значно підвищити точність оцінювання. Наприклад, завдання, виконане вчасно, або позитивний відгук, що підтверджує високий рівень професійності, можуть суттєво впливати на загальну оцінку. Такий підхід дозволяє враховувати більш широкий спектр факторів, які впливають на довіру та репутацію користувачів.

Адаптація рейтингової системи до нових користувачів є ще однією перспективою. Нові користувачі часто стикаються з проблемою недостатньої кількості оцінок, що може негативно впливати на їхню видимість або довіру з боку інших учасників. Для вирішення цієї проблеми можуть використовуватися адаптивні алгоритми, які базуються на аналізі перших взаємодій користувача. Наприклад, системи можуть тимчасово надавати новим учасникам базовий рейтинг або використовувати демографічні дані й початкові показники активності для побудови їхньої репутації. Оцінка ефективності моделі через зворотний зв'язок від користувачів також є важливим аспектом розвитку рейтингових систем. Регулярне отримання відгуків щодо функціонування системи дозволяє виявляти її сильні та слабкі сторони, а також забезпечувати своєчасне оновлення. Використання опитувань, аналізу поведінкових даних і фокус-груп дозволяє зрозуміти, наскільки система відповідає потребам користувачів і сприяє побудові довіри та стимулюванню активності. Зокрема, впровадження елементів гейміфікації або

візуалізації досягнень може покращити користувацький досвід і підвищити залученість.

Розвиток рейтингових систем у веб-орієнтованих платформах спрямований на їхню більшу точність, гнучкість та адаптивність. Інтеграція нових змінних, створення механізмів підтримки новачків та активне використання зворотного зв'язку дозволять забезпечити довгострокову ефективність цих систем і їхню релевантність у динамічному цифровому середовищі.

Висновки до розділу 3

Рейтингові системи є важливим елементом веб-орієнтованих платформ, забезпечуючи прозору взаємодію, довіру та стимулюючи користувачів до активної та якісної участі. Їх розвиток пройшов довгий шлях від простих моделей, заснованих на кількісних оцінках, до складних алгоритмів, що враховують вагові коефіцієнти, складність завдань і інші фактори. Використання зваженого середнього як основи для моделювання рейтингу дозволяє забезпечити справедливість оцінювання, стимулюючи користувачів брати участь у складніших взаємодіях і сприяючи їхньому професійному розвитку. Інтеграція нових змінних у рейтинг, адаптація системи до нових користувачів та використання зворотного зв'язку є ключовими перспективами, які дозволяють підвищити ефективність цих систем. Такі підходи сприяють довгостроковому успіху платформ, забезпечують мотивацію для активності користувачів та створюють комфортне середовище для взаємодії. Разом із механізмами запобігання зловживанням ці кроки дозволяють удосконалити моделі рейтингу, зробивши їх універсальним інструментом для підтримки якості та продуктивності у сучасному цифровому середовищі.

Таким чином, рейтингові системи не лише відображають поточний стан діяльності користувачів, але й формують їхній майбутній розвиток, адаптуючись до викликів і потреб цифрового світу.

4 ПРОЄКТУВАННЯ БАЗИ ДАНИХ

4.1 Етапи проєктування бази даних

Проектування бази даних – це достатньо довгий процес. Він включає наступні етапи:

- концептуального проєктування;
- логічного проєктування;
- фізичного проєктування.

Етап концептуального проєктування – це процес представлення БД, який включає визначення найважливіших сутностей та зв'язків, які між ними існують [6]. На етапі концептуального проєктування – розроблено концептуальну діаграму, яка представлена на рис. 4.1.

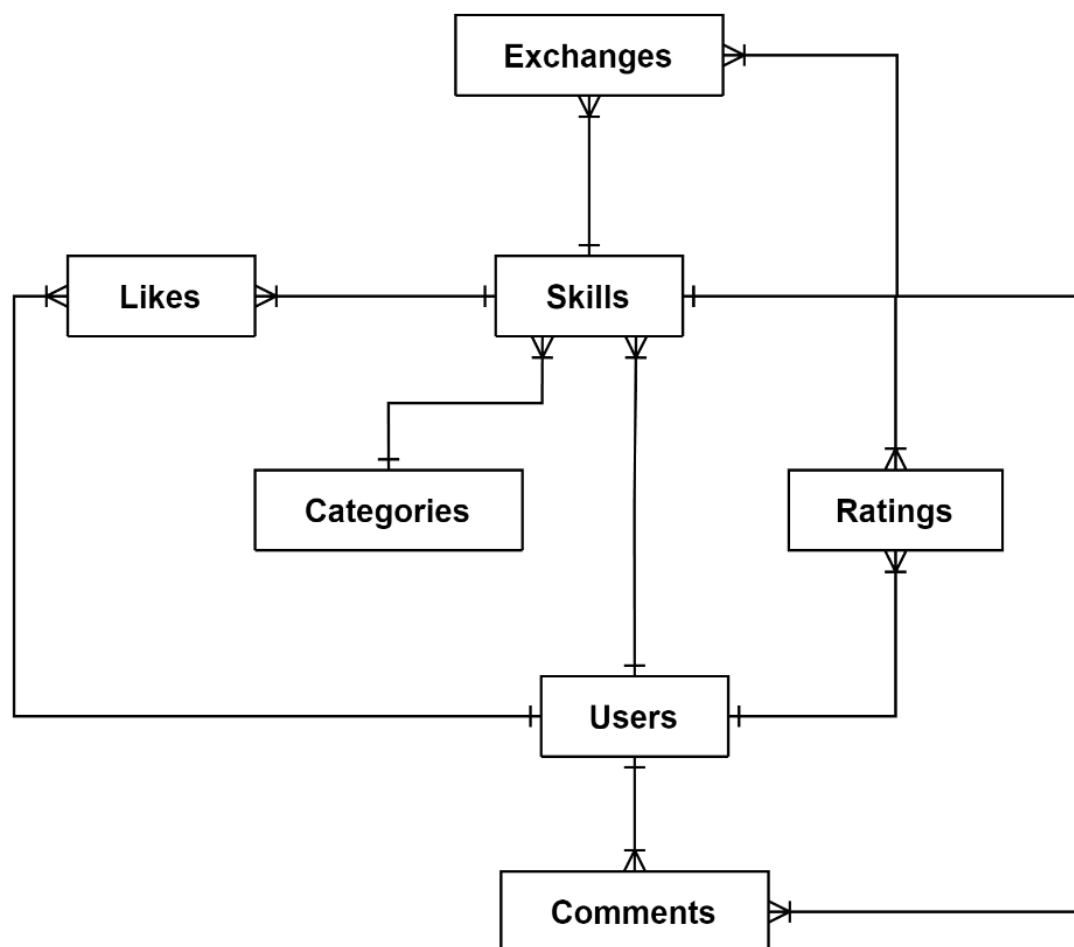


Рисунок 4.1 – Концептуальна діаграма БД

Це перший етап у проєктуванні, який передбачає створення концептуальної моделі даних для потрібної предметної області. Така модель заснована на інформації, яка походить від вимог користувача.

Даний етап проєктування ніяким чином не залежить від системи керування базами даних (СКБД) чи обраної мови програмування.

Процес логічного проєктування застосовується до конкретної моделі даних. За певними правилами логічна модель перетворюється з концептуальної. Однак, така логічна модель має бути коректною. Для того, щоб переконатись в структурній узгодженості та мінімальній збитковості – використовують правила нормалізації даних [6].

На етапі логічного проєктування – розроблено логічну діаграму, яка представлена на рис. 4.2.

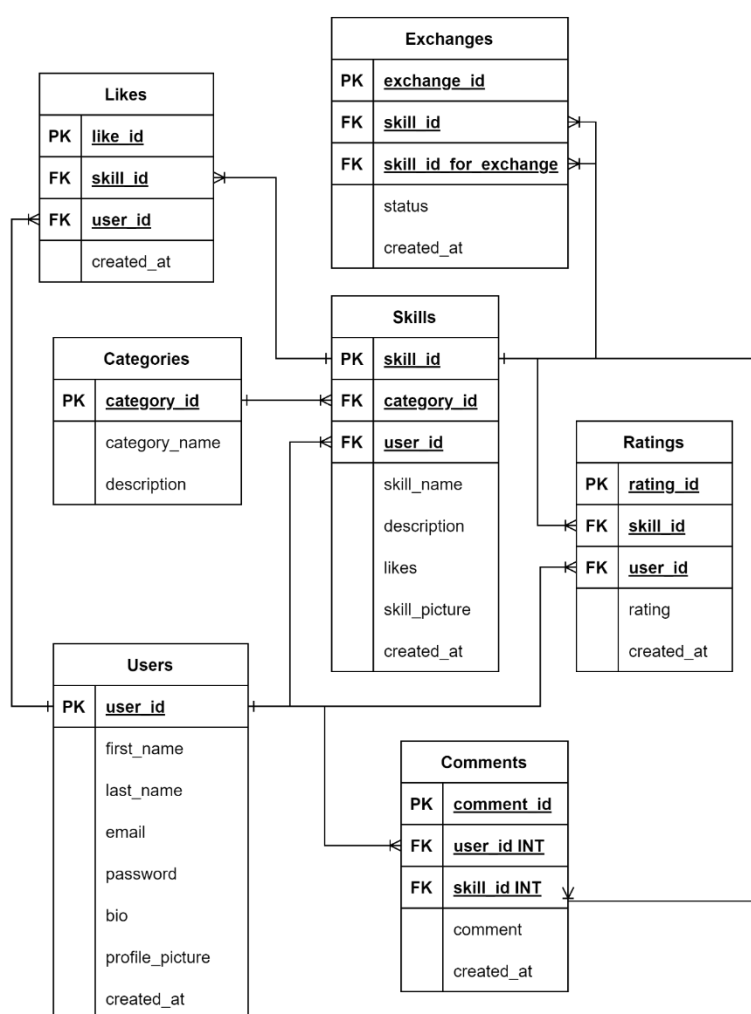


Рисунок 4.2 – Логічна діаграма БД

Фізичне проєктування бази даних полягає у фізичній реалізації логічної моделі [7]. Для реалізації фізичної моделі було проведено основні етапи нормалізації, спроектовано БД та побудовано діаграму за допомогою СКБД MySQL (рис. 4.3).

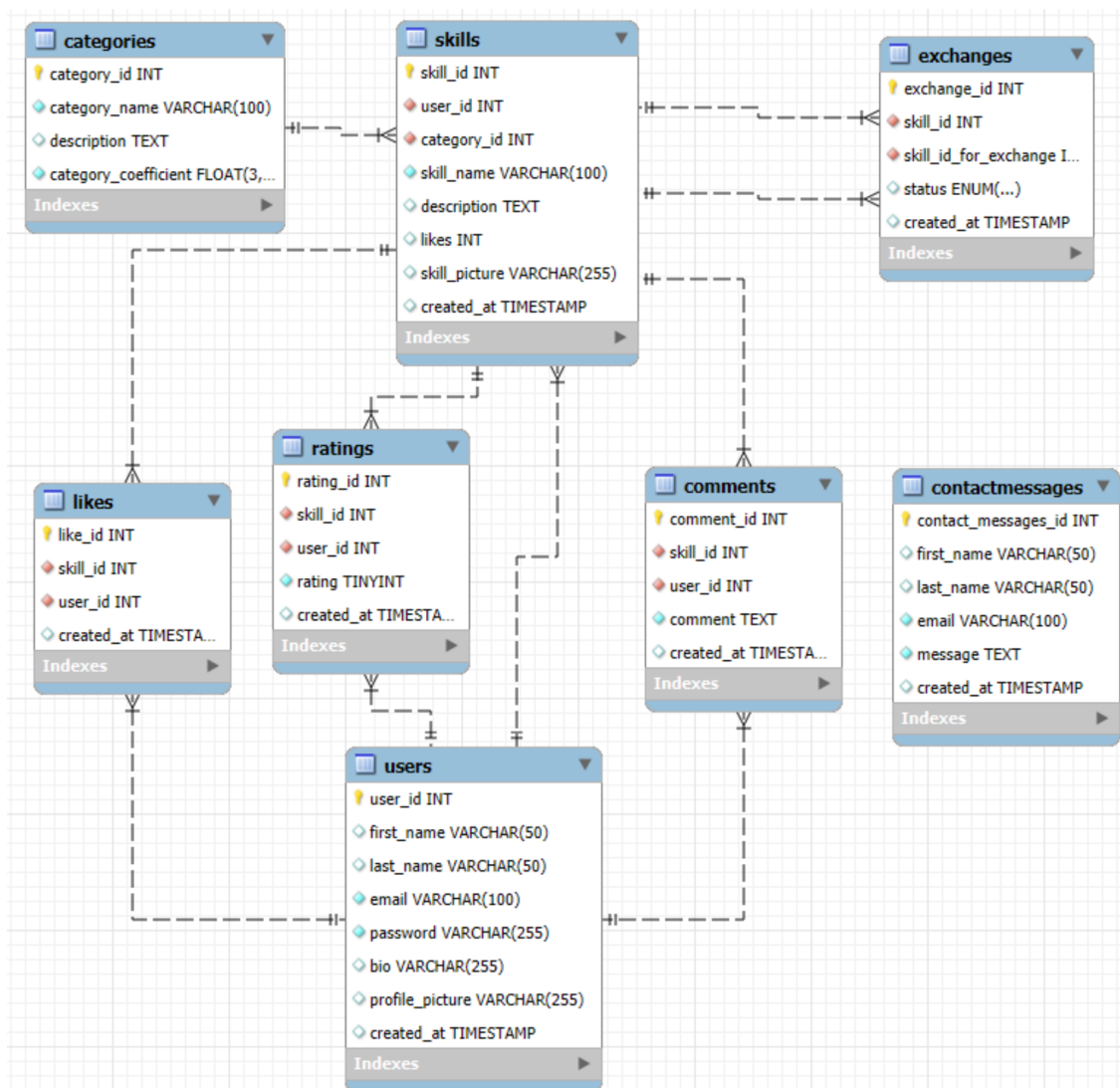


Рисунок 4.3 – Діаграма створеної БД в системі керування базами даних MySQL

4.2 Структура бази даних

Зберігання даних виконується за допомогою реляційної системи керування базами даних MySQL. У процесі проєктування бази даних виділено сутності, опис яких наведено в таблиці 4.1.

Таблиця 4.1 – Характеристика таблиць створеної бази даних

Назва таблиці	Характеристика
users	Міститься вся інформація про зареєстрованих користувачів
skills	Міститься вся інформація про створені навички
exchanges	Містить в собі інформацію про здійснені обміни навичками
categories	Містить в собі інформацію про існуючі категорії, використовується для фільтрації навичок
likes	Міститься в собі інформацію про вподобані навички
comments	Міститься в собі інформація про залишені коментарі
contactmessages	Містить в собі інформацію про будь-якого користувача, який заповнив форму для зв'язку з адміністрацією
rating	Містить в собі дані для обчислення рейтингової складової кожної навички

4.3 Опис таблиць бази даних

Всю необхідну інформацію про створену БД (назва стовпців, тип даних та інші характеристики) представлено у вигляді табл. 4.2 – 4.8. Аббревіатура РК представляє первинні ключі, а FK – зовнішні.

Таблиця 4.2 – Опис таблиці users

<i>users</i>		
Назва поля	Тип даних	Характеристики
PK user_id	INT	Первинний ключ
first_name	VARCHAR(50)	Обов'язкове поле

<i>users</i>		
Назва поля	Тип даних	Характеристики
last_name	VARCHAR (50)	Обов'язкове поле
email	VARCHAR (100)	Обов'язкове поле
password	VARCHAR (255)	Обов'язкове поле
bio	TEXT	Обов'язкове поле
profile_pictures	VARCHAR (255)	Обов'язкове поле
created_at	TIMESTAMP	Обов'язкове поле

Таблиця 4.3 – Опис таблиці skills

<i>skills</i>		
Назва поля	Тип даних	Характеристики
PK skill_id	INT	Первинний ключ
skill_name	VARCHAR(100)	Обов'язкове поле
description	TEXT	Обов'язкове поле
likes	INT	Обов'язкове поле
skill_picture	VARCHAR (255)	Обов'язкове поле
created_at	TIMESTAMP	Обов'язкове поле
FK user_id	INT	Зовнішній ключ
FK category_id	INT	Зовнішній ключ

Таблиця 4.4 – Опис таблиці exchanges

<i>exchanges</i>		
Назва поля	Тип даних	Характеристики
PK exchanges_id	INT	Первинний ключ
status	ENUM(...)	Обов'язкове поле

<i>exchanges</i>		
Назва поля	Тип даних	Характеристики
created_at	TIMESTAMP	Обов'язкове поле
FK skill_id	INT	Зовнішній ключ
FK skill_id_for_exchange	INT	Зовнішній ключ

Таблиця 4.5 – Опис таблиці comments

<i>comments</i>		
Назва поля	Тип даних	Характеристики
PK comment_id	INT	Первинний ключ
comment	TEXT	Обов'язкове поле
created_at	TIMESTAMP	Обов'язкове поле
FK skill_id	INT	Зовнішній ключ
FK user_id	INT	Зовнішній ключ

Таблиця 4.6 – Опис таблиці likes

<i>likes</i>		
Назва поля	Тип даних	Характеристики
PK like_id	INT	Первинний ключ
created_at	TIMESTAMP	Обов'язкове поле
FK skill_id	INT	Зовнішній ключ
FK user_id	INT	Зовнішній ключ

Таблиця 4.7 – Опис таблиці *categories*

<i>categories</i>		
Назва поля	Тип даних	Характеристики
PK category_id	INT	Первинний ключ
category_name	VARCHAR(100)	Обов'язкове поле
description	TEXT	Обов'язкове поле

Таблиця 4.8 – Опис таблиці *contactmessages*

<i>contactmessages</i>		
Назва поля	Тип даних	Характеристики
PK contact_messages_id	INT	Первинний ключ
first_name	VARCHAR(50)	Обов'язкове поле
last_name	VARCHAR(50)	Обов'язкове поле
email	VARCHAR(100)	Обов'язкове поле
message	TEXT	Обов'язкове поле
created_at	TIMESTAMP	Обов'язкове поле

Таблиця 4.9 – Опис таблиці *rating*

<i>rating</i>		
Назва поля	Тип даних	Характеристики
PK rating_id	INT	Первинний ключ
rating	TINYINT	Обов'язкове поле
created_at	TIMESTAMP	Обов'язкове поле
FK skill_id	INT	Зовнішній ключ
FK user_id	INT	Зовнішній ключ

Висновки до розділу 4

У процесі написання цього розділу магістерської дисертації було спроектовано базу даних. Це один із ключових етапів розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки всі дані мають десь зберігатись та буди доступними за запитом. У процесі розробки були проаналізовані етапи проєктування бази даних. Дані етапи дозволили розробити структуру бази даних та визначити логіку взаємодії з нею. Також в ході аналізу – спроектовані концептуальна, логічна та фізична діаграми БД. На етап фізичного проєктування було обрано СКБД MySQL, яка відповідала вимогам до продуктивності та масштабованості. Варто зазначити, що для розробки веб-орієнтованої системи обміну навичками та знаннями було обрано реляційну базу даних через її ефективність при роботі з великими обсягами інформації.

Заключний етап цього розділу – опис таблиць бази даних, який включає назву полів, тип даних та характеристику. Отже, проєктування БД продемонструвало всі основні етапи, які починаються від аналізу вимог та закінчуються створенням фізичної моделі. Результатом цього розділу є створення надійною, швидкої та ефективною бази даних.

5 ПРОЄКТУВАННЯ СИСТЕМИ

5.1 Сценарії використання системи

Веб-орієнтована система обміну навичками та знаннями в основному залежатиме від авторизації, однак частина функціональних можливостей таких як: переглянути доступні навички чи рейтинг користувачів, який сформувався у процесі обмінів, – доступні будь-кому. Але варто зазначити, що реєстрація та авторизація є надзвичайно важливою, оскільки відкриває більшу частину функціоналу.

Опис наявних сценаріїв користувача представлено в табл. 5.1-5.14.

Таблиця 5.1 – Варіант використання UC001

Назва	Перегляд сторінки з доступними навичками
Характеристика	Неавторизований користувач має можливість переглянути список навичок, а також побачити їх деталі
Учасник	Неавторизований користувач
Передумова	Наявність виставлених навичок іншими учасниками
Успішний сценарій	1. Перехід на сторінку з опублікованим списком навичок 2. Перегляд деталей потрібної навички
Результат	Неавторизований користувач побачив опубліковані навички та переглянув їх деталі
Виключення	Якщо список навичок порожній, то відобразиться відповідне повідомлення

Таблиця 5.2 – Варіант використання UC002

Назва	Перегляд відфільтрованих навичок
Характеристика	Неавторизований користувач має можливість відфільтрувати навички по категоріях

Назва	Перегляд відфільтрованих навичок
Учасник	Неавторизований користувач
Передумова	Наявність виставлених навичок іншими учасниками
Успішний сценарій	1. Перехід на сторінку з опублікованим списком навичок 2. Встановлення необхідних критеріїв
Результат	Неавторизований користувач перейшов на сторінку з опублікованими навичками, застосував фільтр та отримав результат його роботи
Виключення	Якщо список навичок порожній, то відобразиться відповідне повідомлення

Таблиця 5.3 – Варіант використання UC003

Назва	Перегляд деталей конкретної навички
Характеристика	Неавторизований користувач має можливість переглянути деталі конкретної навички
Учасник	Неавторизований користувач
Передумова	Наявність виставлених навичок іншими учасниками
Успішний сценарій	1. Перехід на сторінку з опублікованим списком навичок. 2. Вибір конкретної навички та перегляд її деталей
Результат	Неавторизований користувач перейшов на сторінку з опублікованими навичками, вибрав бажану та переглянув її деталі.
Виключення	Якщо список навичок порожній, то відобразиться відповідне повідомлення

Таблиця 5.4 – Варіант використання UC004

Назва	Зв'язок з адміністрацією
Характеристика	Неавторизований користувач має можливість заповнити форму для зв'язку з адміністрацією
Учасник	Неавторизований користувач
Передумова	Наявність створеної форми
Успішний сценарій	1. Перехід на сторінку «Контакти» 2. Заповнення форми 3. Натиснення кнопки «Надіслати» 4. Отримання повідомлення про успіх
Результат	Неавторизований користувач перейшов на сторінку з контактною інформацією, заповнив форму та надіслав
Виключення	Якщо під час набирання форми пропустити поле чи ввести некоректні дані, то відобразиться відповідне повідомлення

Таблиця 5.5 – Варіант використання UC005

Назва	Реєстрація в системі
Характеристика	Неавторизований користувач має можливість зареєструватись у системі
Учасник	Неавторизований користувач
Передумова	Наявність сторінки для реєстрації
Успішний сценарій	1. Перехід на сторінку реєстрації 2. Введення реєстраційних даних 3. Отримання повідомлення про успішну реєстрацію
Результат	Неавторизований користувач перейшов на сторінку з реєстрацією, ввів реєстраційні дані та одержав свій обліковий запис

Назва	Реєстрація в системі
Виключення	Якщо такий користувач існує, то відобразиться відповідне повідомлення

Таблиця 5.6 – Варіант використання UC006

Назва	Авторизація користувача
Характеристика	Неавторизований користувач має можливість здійснити вхід в обліковий запис
Учасник	Неавторизований користувач
Передумова	Наявність сторінки для входу в профіль
Успішний сценарій	1. Перехід на сторінку входу 2. Введення даних, які використовувались при реєстрації 3. Отримання повідомлення про успішний вхід
Результат	Неавторизований користувач перейшов на сторінку з входом в обліковий запис, ввів свої дані та здійснив вхід у власний профіль
Виключення	Якщо були помилково введені дані, то відобразиться відповідне повідомлення

Таблиця 5.7 – Варіант використання UC007

Назва	Редагування особистого профілю
Характеристика	Користувач має можливість змінити особисту інформацію у своєму обліковому записі
Учасник	Користувач
Передумова	Наявність входу в обліковий запис та перехід у власний кабінет

Назва	Редагування особистого профілю
Успішний сценарій	1. Перехід на вкладку «Інформація про користувача» 2. Зміна потрібних даних 3. Отримання повідомлення про успішне збереження нових даних у профілі
Результат	Користувач перейшов на сторінку зі своїми особистими даними, виконав їх редагування та отримав повідомлення про успішне збереження
Виключення	Не існує

Таблиця 5.8 – Варіант використання UC008

Назва	Створення власної навички
Характеристика	Користувач має можливість створювати власні навички
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Додати навичку» 2. Введення необхідної інформації 3. Отримання повідомлення про успішне створення навички
Результат	Користувач перейшов на сторінку для створення навичок, ввів необхідну інформацію про навичку та отримав повідомлення про успішне створення навички
Виключення	Не існує

Таблиця 5.9 – Варіант використання UC009

Назва	Редагування власної навички
Характеристика	Користувач має можливість редагувати власні навички
Передумова	Наявність входу в обліковий запис

Назва	Редагування власної навички
Успішний сценарій	1. Перехід на вкладку «Кабінет» > «Мої навички» 2. Вибір потрібної навички та натискання кнопки «Редагувати» 3. Зміна необхідної інформації стосовно навички 4. Отримання повідомлення про успішне оновлення навички
Результат	Користувач перейшов на сторінку зі своїми навиками, натиснув кнопку «Редагувати», змінив необхідні дані та отримав повідомлення про успішне оновлення навички
Виключення	Не існує

Таблиця 5.10 – Варіант використання UC010

Назва	Видалення власної навички
Характеристика	Користувач має можливість видаляти власні навички
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Кабінет» > «Мої навички» 2. Вибір потрібної навички та натискання кнопки «Видалити» 3. Підтвердження операції у спливаючому вікні 4. Отримання повідомлення про успішне видалення навички
Результат	Користувач перейшов на сторінку зі своїми навиками та вибрав потрібну, натиснув кнопку «Видалити», підтвердив своє рішення та отримав повідомлення про успішне видалення навички
Виключення	Не існує

Таблиця 5.11 – Варіант використання UC011

Назва	Процес обміну навичкою
Характеристика	Користувач має можливість здійснити обмін навичкою
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Переглянути навички» 2. Вибрати бажану навичку 3. Натиснути кнопку «Запропонувати обмін навичками» 4. Вибрати зі списку одну із своїх навичок 5. Отримання повідомлення про успішне надсилання запиту
Результат	Користувач перейшов на сторінку з доступними навичками, вибрав бажану та свою навичку, яку пропонує на обмін, та отримав повідомлення про успішне надсилання запиту
Виключення	Якщо користувач уже проводить обмін з такою навичкою та цим учасником, то відобразиться відповідне повідомлення

Таблиця 5.12 – Варіант використання UC012

Назва	Вподобання навички
Характеристика	Користувач має можливість вподобати навичку (додати до списку уподобань)
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Переглянути навички» 2. Натиснути кнопку «Вподобати» 3. Переглянути вподобану навичку на вкладці «Вподобані навички» в особистому кабінеті

Назва	Вподобання навички
Результат	Користувач перейшов на сторінку з доступними навичками, вподобав деяку навичку та переглянув список із вподобаними навичками у своєму кабінеті
Виключення	Якщо користувач уже вподобав цю навичку в того самого учасника, то навичка візуально це продемонструє

Таблиця 5.13 – Варіант використання UC013

Назва	Перегляд надісланих заявок на обмін навичкою
Характеристика	Користувач має можливість переглянути список надісланих заявок
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Мої заявки» > «Надіслані заявки» 2. Натиснути кнопку «Переглянути інформацію» на прийнятій заявці 3. Отримання опису навички та контактної інформації відправника
Результат	Користувач перейшов на вкладку з надісланими заявками, перевірів статус заявок та переглянув інформацію стосовно навички
Виключення	Якщо користувачу відхили заявку, то кнопки «Переглянути інформацію» не існуватиме. Також буде вписано статус «Відхилено»

Таблиця 5.14 – Варіант використання UC014

Назва	Перегляд отриманих заявок на обмін навичкою
Характеристика	Користувач має можливість переглянути список отриманих заявок
Учасник	Користувач
Передумова	Наявність входу в обліковий запис
Успішний сценарій	1. Перехід на вкладку «Мої заявки» > «Отримані заявки» 2. Натискання кнопки «Переглянути інформацію» на прийнятій заявці 3. Отримання опису навички та контактної інформації відправника
Результат	Користувач перейшов на вкладку з отриманими заявками, перевірів статус заявок та переглянув інформацію стосовно навички
Виключення	Якщо користувач відхилив заявку, то кнопки «Переглянути інформацію» не існуватиме. Також буде вписано відповідний статус, який вибрав користувач

5.2 Архітектура веб-орієнтованої системи

5.2.1 Діаграма варіантів використання

Діаграма варіантів використання застосовується для представлення сценаріїв використання системи та користувачів, які користуються функціями цієї системи. На такій діаграмі в основному використовують позначення акторів (символом людини) та варіантів використання (еліпсом). Актори з'єднуються з варіантами використання стрілкою, яка спрямована в бік варіанту використання. Таку стрілку називають напрямленою асоціацією. Також існують зв'язки узагальнення [8].

Діаграма варіантів використання представляє функціональні вимоги, а саме показує те, що система здатна виконувати з точки зору користувача. Діаграма варіантів використання, яка побудована для веб-орієнтованої системи обміну

навичками та знаннями, представлена в Додатку А.

5.2.2 Діаграми послідовності

Діаграма послідовності показує взаємодію об'єктів, які упорядковані за часом. За допомогою такої діаграми можна представити всю взаємодію моделей як часовий графік життя всіх об'єктів. На діаграмі послідовності розміщують об'єкти, які беруть участь при взаємодії, однак ніякі зв'язки з розміщеними там статичними об'єктами не демонструються та не візуалізуються. Весь процес взаємодії розміщених об'єктів здійснюється за допомогою повідомлень, ці повідомлення посилаються від одних об'єктів до інших. Таке повідомлення – це сигнал для виконання операції. Цю операцію повинен виконати об'єкт, який прийняв надіслане повідомлення [9]. Часто на діаграмах послідовності використовують три різні типи повідомлень, кожне з них має своє графічне призначення:

- виклик процедури (найпоширеніший тип повідомлень, який застосовується для виклику процедур та виконання операцій);
- асинхронний виклик (застосовують для простого асинхронного повідомлення);
- повернення з процедури (застосовують для демонстрації повернення з виклику процедури).

Діаграма послідовності (для контактів), яка побудована для веб-орієнтованої системи обміну навичками та знаннями, представлена на в Додатку Б. Діаграма послідовності функціонування представлена в Додатку В.

5.2.3 Діаграма станів

Діаграма станів призначена описати послідовності станів та переходів, які визначають поведінку модельованої системи протягом її життєвого циклу. Такі діаграми найчастіше застосовують для того, щоб описати поведінку деяких системи чи підсистем [8].

Діаграма станів, яка побудована для веб-орієнтованої системи обміну навичками та знаннями, представлена в Додатку Г.

5.2.4 Діаграма активності

Діаграма активності – це розширений аналог блок-схеми будь-якого алгоритму роботи системи. Цей тип діаграми доцільно використовувати для аналізу всіх сценаріїв застосування спроектованої системи. Кожний стан на діаграмі активності демонструє виконання деякої операції. Перехід на наступний стан можливий тільки після завершення поточної операції [10].

Діаграма активності, яка побудована для веб-орієнтованої системи обміну навичками та знаннями, представлена в Додатку Д.

5.2.5 Діаграма компонентів

Діаграми компонентів використовують для моделювання компонентів, які допомагають зробити заявлені функціональні можливості. Цей тип діаграм використовується для демонстрації організації компонентів системи та зв'язку між ними [8]. Компоненти можуть бути наступними:

- програмні компоненти (бази даних або користувацький інтерфейс);
- апаратні компоненти (схема приладу).

Компонентні діаграми повинні допомогти інформаційно пояснити функціональні можливості спроектованої системи зацікавленій стороні. Діаграма компонентів, яка побудована для веб-орієнтованої системи обміну навичками та знаннями, представлена в Додатку Е.

5.2.6 Діаграма розгортання

Діаграма розгортання демонструє архітектуру програмного продукту. Цей тип діаграми призначений для демонстрації елементів та компонентів, які існують тільки на етапі виконання. Інші компоненти, які ніяким чином не використовуються на етапі виконання – на цьому типі діаграма не показуються [8].

Діаграма розгортання, яка побудована для веб-орієнтованої системи обміну навичками та знаннями, представлена в Додатку Ж.

Також були розроблені діаграми бізнес-процесів: користувач, робота в кабінеті, перегляд навиків, додавання нового навичку. Дані діаграми наведено в Додатках И, К, Л та М.

5.3 Вибір технологій

5.3.1 Вибір мови програмування

Мова програмування відіграє важливу роль у процесі розробки веб-системи, оскільки окреслює як саме веб-додаток буде функціонувати. Існує велика кількість різноманітних мов програмування та всі вони мають певне призначення. Оскільки потрібно вибрати саме ту мову програмування, яка ідеально підійде для веб-орієнтованої системи обміну навичками та знаннями, то необхідні вибрати перелік з трьох найпопулярніших мов та проаналізувати їх сильні та слабкі сторони [11]. До списку найпопулярніших мов програмування, які використовують у розробці веб-систем належать:

- PHP;
- Python;
- JavaScript;
- Java;
- Go;
- Rust;
- Ruby.

Для здійснення вибору буде проведено аналіз мов PHP, Python та JavaScript. PHP – мова програмування, яка призначена для розробки веб-систем та веб-інтерфейсів. Ця мова належить до списку найпопулярніших мов для веб-проектування та дозволяє створювати додатки будь-якої складності. Часто розробники надають перевагу мові програмування PHP, оскільки на ній розроблені різноманітні готові рушії та фреймворки. Варто зазначити, що для створення простих веб-додатків на PHP використовують системи керування вмістом (CMS). WordPress, Joomla і Drupal – це одні з найпопулярніших систем керування вмістом, які написати на PHP. Мова програмування PHP була створена спеціально для розробки веб-орієнтованих систем, тому це її основна галузь, для якої її застосовують до тепер [12]. Також у наш час на даній мові програмування створюють:

- веб-додатки (малі та середні);
- CMS (системи керування вмістом – готові рушії, які дозволяють з величезною швидкістю зібрати веб-додаток;
- CRM (системи для керування відносинами з клієнтами – готові системи, які використовують для збору, зберігання та аналізу інформації про користувачів);
- ERP (автоматизована система планування ресурсів на підприємствах, призначена допомогти підприємствам оптимізувати основні бізнес-процеси).

Для здійснення вибору на користь певної мови програмування необхідно розглянути переваги та недоліки. До переваг скриптової мови програмування PHP належать:

- проста у процесі вивчення;
- добре працює з іншими технологіями;
- велика користувацька спільнота;
- добре працює з найпопулярнішими базами даних;
- має велику кількість безкоштовних бібліотек та фреймворків;
- продуктивна;

- сумісна з основними операційними системами;
- широко застосовується на ринку;
- дозволяє швидко масштабувати систему.

До недоліків даної мови програмування належать:

- наявні вразливості в системі безпеки;
- наявні протиріччя синтаксису;
- проблеми з назвами вбудованих функцій;
- змішаний код.

Розглянувши наявні переваги та недоліки можна зробити висновок, що дана мова програмування PHP не підходить для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки містить вагомий критичний недолік – наявні вразливості в системі безпеки.

Python – це надзвичайно популярна об'єктно-орієнтована мова програмування, яка містить велику бібліотеку модулів та відкритий програмний код. Ця мова програмування має достатньо зрозумілий синтаксис, тому повною мірою підходить для початківців в галузі програмування. Також мову програмування Python широко застосовують в таких областях як: веб-розробка, аналіз даних, розроблення ігор, машинне навчання, автоматизація та ін [12]. (рис. 5.8).

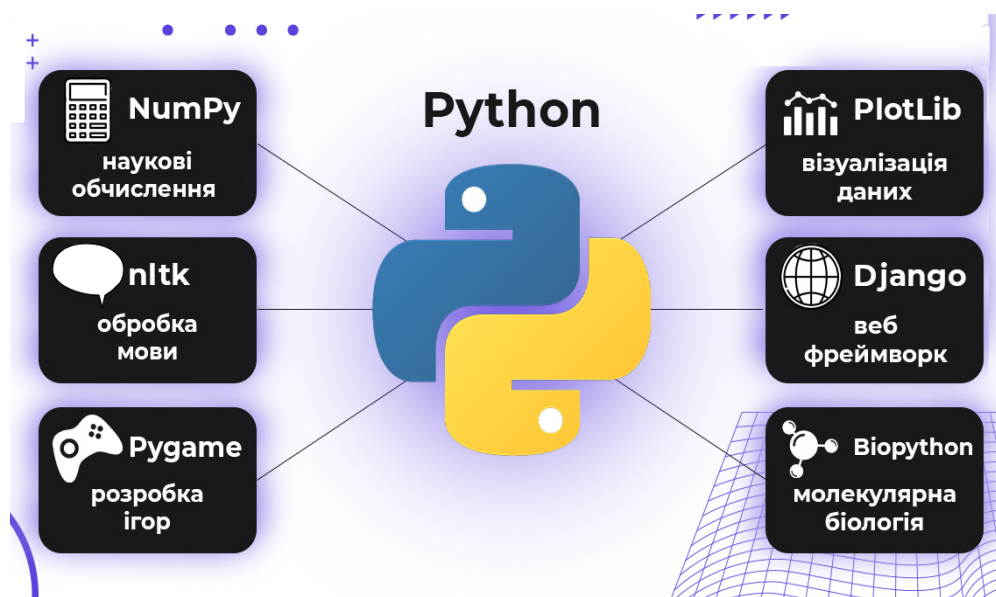


Рисунок 5.8 – Сфери застосування мови програмування Python

Окрім того, що мова Python містить велику бібліотеку сторонніх інструментів та модулів, вона також чудово підтримується активною спільнотою розробників, які часто випускають оновлення та надають величезні обсяги навчальних матеріалів [13].

Для здійснення вибору на користь певної мови програмування необхідно розглянути їх переваги та недоліки. До переваг високорівневої мови програмування Python належать:

- простий та доступний програмний код;
- велика кількість корисних модулів та бібліотек;
- кросплатформенна мова програмування;
- мова з динамічною типізацією;
- хороша підтримка спільноти розробників;
- мова для швидкого створення прототипів;
- єдиний стандарт коду;
- достатня продуктивність.

Недоліками цієї мови програмування є:

- погане керування пам'яттю;
- важко масштабувати для дуже великих проєктів;
- невелика швидкість виконання;
- немає достатньої підтримки для мобільної розробки.
- велика кількість модулів може негативно позначитись на роботі розробленої системи.

Розглянувши наявні переваги та недоліки можна зробити висновок, що високорівнева мова програмування Python не підходить для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки містить такі недоліки як важкість масштабування при збільшенні проєкту та має погане керування пам'яттю.

JavaScript – це скриптова мова програмування, яку застосовують для створення інтерактивних веб-додатків та інтерфейсів. Загальновідомим фактом є те, що кожен веб-додаток розроблений за допомогою трьох технологій – це HTML,

CSS і JavaScript. Остання – відповідає за динамічні та інтерактивні інтерфейси. За допомогою JavaScript можна додавати анімацію чи спливаючі вікна, оновлювати контент та змушувати інтерфейс реагувати на будь-які дії користувача. Головне застосування цієї скриптової мови – веб-розробка [14]. Також JavaScript використовується для:

- створення клієнтської частини (Frontend);
- створення серверних додатків (Backend, Node.js);
- створення програм для ПК;
- розробки додатків для Android та iOS (React Native);
- програмування різного роду техніки;
- створення розширень для браузерів.

Рис. 5.9 демонструє роботу скриптової мови програмування JavaScript у веб-браузері.

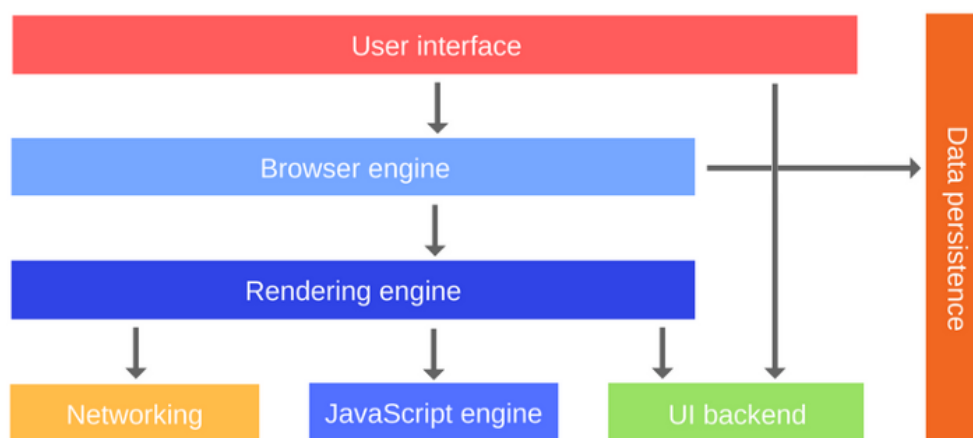


Рисунок 5.9 – Робота скриптової мови програмування JavaScript у веб-браузері

Розглянути роботу JavaScript можна на стандартному прикладі:

1. Користувач веб-додатку натискає кнопку «Реєстрація».
2. Браузер записує дію та запускає JavaScript-код.
3. Мова програмування JavaScript в той час виконує перелік дій: відображає анімацію натискання кнопки та відкриває форму для реєстрації.

4. В залежності від того як створена форма – відбувається оновлення сторінки.

Для здійснення вибору на користь тої чи іншої мови програмування необхідно розглянути переваги та недоліки. До переваг скриптової мови програмування JavaScript належать:

- добре працює на будь-яких пристроях;
- має велику кількість бібліотек та фреймворків;
- проста у вивченні;
- має відкритий вихідний код;
- асинхронна мова програмування;
- підтримує основні парадигми програмування;
- має функцію автоматичного збору сміття.

Дана мова програмування має наступні недоліки:

- може бути не настільки продуктивною, якщо брати до уваги інші мови програмування;
- в окремих випадках динамічна типізація може створювати певні труднощі.

Розглянувши наявні переваги та недоліки можна зробити висновок, що мова програмування JavaScript підійде для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки містить мінімальну кількість недоліків.

Для підтвердження зробленого вибору та підведення підсумків –доцільно зробити порівняльну таблицю. У табл. 5.15 наведено отримані результати у ході проведеного порівняльного аналізу.

Таблиця 5.15 – Проведений порівняльний аналіз вибраних мов програмування

Критерій, за яким було проведено порівняння	Мова програмування		
	PHP	Python	JavaScript
Проста у вивченні	+/-	+/-	+
Містить велику кількість бібліотек та фреймворків	+	+	+

Критерій, за яким було проведено порівняння	Мова програмування		
	PHP	Python	JavaScript
Можна розробляти як клієнтську так і серверну частину	—	—	+
Універсальна	+/-	+	+
Швидкодія	+/-	—	+
Легкість у масштабуванні проєктів	—	—	+
Працює з різноманітним СКБД	+	+	+
Безпека	—	+/-	+/-

З табл. 5.15 можна побачити, що у процесі аналізу популярних мов програмування лідером стала JavaScript. Ця мова програмування універсальна та з використанням додаткових інструментів підходить для розробки клієнтської та серверної частин веб-орієнтованої системи обміну навичками та знаннями.

5.3.2 Платформа для проєктування серверної частини

З проведеного порівняльного аналізу можна помітити, що JavaScript вже дуже давно не зупиняється тільки на розробці клієнтської частини. З розвитком платформи Node.js, мова програмування JavaScript дозволила використовувати себе для розробки не тільки на клієнтській стороні, а й на серверній [15]. Галузі застосування Node.js наведено на рис. 5.10.

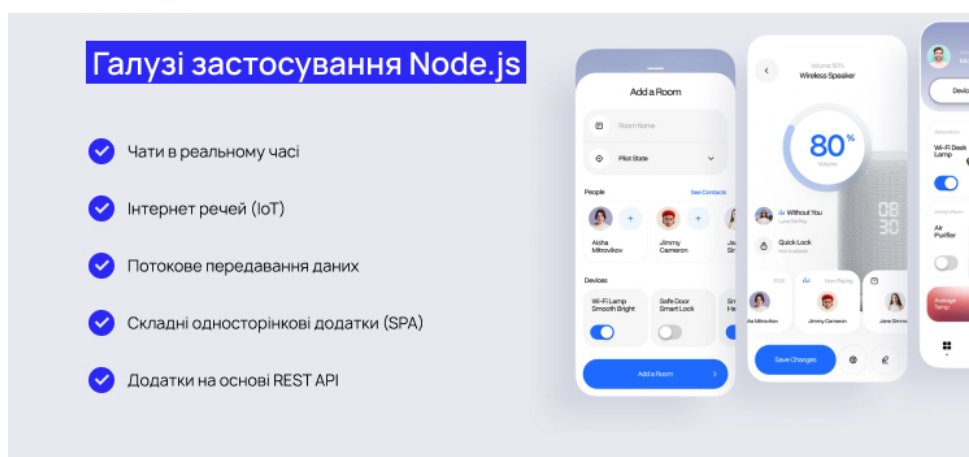


Рисунок 5.10 – Галузі застосування платформи Node.js

Node.js – це платформа для виконання коду JavaScript на стороні сервера. Саме завдяки Node.js – JavaScript отримала друге життя, оскільки розробникам більше не потрібно використовувати кілька мов програмування, переключатись між використанням різних синтаксисів. Це призвело до скорочення часу розробки та глибшого вивчення технологій. Також такий підхід мінімізує проблеми, які можуть виникнути у процесі розробки та спрощують обслуговування розробленої системи. Дана платформа працює асинхронно, тому коли до серверу намагається одночасно підключитись велика кількість людей, то він з легкістю справляється з навантаженням, оскільки не створює окремих потоків на різні підключення [16]. Ефективність в обробці багатьох паралельних запитів допомагає без проблем масштабувати будь-яку веб-систему.

До переваг платформа Node.js належать:

- продуктивна у процесі розробки веб-додатків;
- зручний механізм для обробки помилок;
- продуктивна в повільних мережах;
- дозволяє створювати масштабні проекти та забезпечує просте масштабування;
- можливість застосовувати єдину мову як на клієнті, так і на сервері;
- велика екосистема модулів.

Також Node.js має деякі недоліки:

- однопотоковість;

- недостатня асинхронність.

Платформа Node.js чудово підійде для проєктування серверної частини веб-орієнтованої системи обміну навичками та знаннями, оскільки наявні недоліки критичні тільки для проєктів зі складними обчисленнями, де властиве виконання важким математичних задач.

5.3.3 Вибір веб-фреймворків

Веб-фреймворк – це корисне програмне середовище, яке спрощує розробнику процес створення програмного забезпечення. Дане програмне середовище – це свого роду набір компонентів, бібліотек та інструментів. Також веб-фреймворки надають структуру веб-системі. Вони не тільки спрощують процес розробки, але й покращують якість коду, що в подальшому несе значний вплив на швидкодію всієї системи. Варто зазначити, що фреймворки також полегшують подальшу підтримку розробленої системи, оскільки вони постійно отримують нові оновлення від розробників [17].

Для кожної мови програмування існують свої веб-фреймворки. До списку найбільш популярний фреймворків мови JavaScript належать:

- React;
- Angular;
- Vue.js;
- Ember.js;
- Next.js.

Для здійснення вибору буде проведено аналіз трьох найпопулярніших фреймворків React, Angular та Vue.js.

React – це бібліотека JavaScript, яка має відкритий вихідний код та широко підтримується компанією Мета. Дану бібліотеку використовують для створення швидких та сучасний інтерфейсів. Під час розробки додатків на React можна створювати свої компоненти, які згодом можна повторно використати [18]. Це дуже зручно для розробників, оскільки їм не потрібно буде писати однотипний код

двічі.

До переваг фреймворку React можна віднести:

- швидкодія (використовує віртуальний DOM);
- компонентна архітектура;
- містить велику кількість бібліотек та доповнень;
- дозволяє розробляти швидкі та інтерактивні інтерфейси;
- універсальний (можна використовувати як для веб-додатків, так і для мобільних застосунків);
- можна легко інтегрувати в існуючі проекти.

Недоліками цього фреймворку називають:

- високий поріг входу;
- необхідна велика кількість додаткових бібліотек для складних проєктів;
- можливі проблеми продуктивності у громіздких проєктах;
- занадто швидкі зміни та оновлення (потребує багато зусиль та часу для оновлення знань).

Розглянувши наявні переваги та недоліки можна зробити висновок, що даний фреймворк React не підходить для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки потребує надзвичайно багато зусиль та часу для підтримки працездатності проєкту в майбутньому.

Angular – це популярний веб-фреймворк, який широко підтримується компанією Google. Він має відкритий вихідний код, а його архітектура базується на MVC (Model View Controller) шаблоні. Веб-системи, які створені за допомогою фреймворку Angular просто масштабуються. Дана особливість дозволяє використовувати цей фреймворк для проєктів різної складності.

Angular використовують для розробки SPA (Single Page Application) застосунків. Їх особливість полягає в тому, що вони працюють без перезавантаження веб-сторінки. Усю функціональність веб-системи можна організувати в компонентах, отже цей фреймворк заснований на компонентному підході. Веб-фреймворк Angular використовує TypeScript для написання коду [19]. Це розширення покращує читабельність коду та зменшує кількість помилок при

його написанні.

Angular має наступні переваги:

- розробка чистого коду;
- компонентна організація коду (полегшує підтримку проєкту);
- функції: двонаправлена прив'язка даних, валідація форм та маршрутизація;
- має значну кількість сторонніх інструментів, плагінів та бібліотек;
- довгий термін підтримки командою Google.

До недоліків цього веб-фреймворку варто віднести:

- складний для новачків через велика різноманіття функцій;
- додатки можуть мати значний розмір (даний недолік усувати за допомогою оптимізації);
- надлишкові інструменти для середніх проєктів.

Розглянувши наявні переваги та недоліки можна зробити висновок, що веб-фреймворк Angular не підходить для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки складний для новачків та містить значну кількість надлишкових інструментів, які потрібні тільки для проєктування масштабних веб-систем.

Vue.js – це один із найпопулярніших веб-фреймворків JavaScript, який призначений для розробки користувацьких інтерфейсів. Цей фреймворк полюбили за те, що він дуже швидко освоюється новачками, завдяки його інтуїтивному синтаксису та простому API.

Vue.js – це свого роду бібліотека, яка використовує шаблон MVVM (Model-View-ViewModel). Даний шаблон дозволяє фреймворку легко інтегруватись з іншими проєктами чи бібліотеками. Vue.js добре підходить до невеликих проєктів, в які потрібно додати реактивності. Також Vue.js просто масштабується та чудово підходить для громіздких односторінкових веб-додатків завдяки таким важливим компонентам як Router и Vuex [20].

До переваг фреймворку Vue.js належать:

- низький поріг входу;

- якісно розроблена документація;
- швидкодія додатків, завдяки мізерному розміру фреймворку;
- легко інтегрується;
- можна використовувати як до простих застосунків, так і для великих веб-систем;
- система компонентів полегшує масштабування.

Також слід розглянути недоліки даного веб-фреймворку:

- більше бібліотек та інструментів створюються під більш популярні React або Angular;
- можуть виникнути проблеми при інтеграції в надвеликі проєкти.

Розглянувши наявні переваги та недоліки можна зробити висновок, що веб-фреймворк Vue.js ідеально підійде для розробки веб-орієнтованої системи обміну навичками та знаннями, оскільки має безліч переваг для швидкої та легкої розробки, а його недоліки критичні тільки для надвеликих проєктів.

Для підтвердження зробленого вибору та підведення підсумків доцільно зробити порівняльну таблицю. У табл. 5.16 наведено отримані результати у ході проведеного порівняльного аналізу.

Таблиця 5.16 – Проведений порівняльний аналіз вибраних веб-фреймворків JavaScript

Критерій, за яким було проведено порівняння	Веб-фреймворк JavaScript		
	React	Angular	Vue.js
Простий у вивченні	–	–	+
Легко інтегрується у проєкти	+/-	+/-	+
Легкість у масштабуванні проєктів	+	+/-	+/-
Швидкість розробки	+/-	–	+
Продуктивність	+	+/-	+
Гнучкість та модульність	+/-	+/-	+

З табл. 5.16 можна побачити, що у процесі аналізу популярних веб-фреймворків JavaScript, лідером став Vue.js. Даний фреймворк простий у вивченні та дозволяє легко інтегрувати його у проєкт, також великою перевагою є гнучкість та модульність, яка сильно спрощує масштабування проєкту. Тому Vue.js ідеально підходить для розробки клієнтського інтерфейсу веб-орієнтованої системи обміну навичками та знаннями.

5.3.4 Середовище розробки

Visual Studio Code – це зручний та багатофункціональний редактор коду від компанії Майкрософт. Цей редактор коду призначений для найрізноманітніших мов програмування. За допомогою Visual Studio Code можна не тільки писати програмний код, як в будь-якому текстовому редакторі, а й повноцінно організувати середовище розробки за допомогою величезної кількості плагінів. Це середовище розробки чудово працює на сучасних операційних системах: Windows, MacOS та Linux. Найбільшої популярності та поширення дане середовище набуло через свою безкоштовну основу. Visual Studio Code підійде як новачкам (через свою початкову полегшеність), так і досвідченим розробникам (після підлаштування середовища під себе). Даний редактор підтримує найрізноманітніші мови програмування [21]. Також можна підключити розширення та найбільш популярні фреймворки. Visual Studio Code дозволяє:

- писати та редагувати програмний код;
- надає можливість швидкого створення проєкту зі своєю структурою;
- підсвічує синтаксис коду;
- дозволяє швидко виправити помилки в коді;
- автоматично розставляє відступи;
- можна в кілька разів розширити функції за допомогою плагінів та доповнень із стандартного каталогу.

До переваг Visual Studio Code відносять:

- легко освоєння завдяки інтуїтивно зрозумілому інтерфейсу;

- мультиплатформність;
- за допомогою встановлених плагінів можна додати підтримку будь-якої мови програмування;
- має невеликий розмір.

До недоліків середовища розробки Visual Studio Code відносять тільки те, що він має обмежені можливості у порівнянні Visual Studio, однак наявних можливостей цілком достатньо для створення проєктів середнього розміру.

Отже, для розробки веб-орієнтованої системи обміну навичками та знаннями обрано середовища розробки Visual Studio Code, оскільки воно повністю задовольняє користувацькі потреби.

5.4 Розробка серверної частини веб-системи

Після створення проєкту та установки потрібних модулів, необхідно підключитись до системи керування базою даних MySQL. Фрагмент коду, який демонструє створення підключення до серверу MySQL продемонстровано на рис. 5.11.

```
30  // Підключення до БД
31  const db = mysql.createConnection({
32    host: "127.0.0.1",
33    user: "root",
34    password: "1111",
35    database: "SkillExchange",
36  });
37
38  db.connect((err) => {
39    if (err) {
40      console.error("Database connection failed: " + err.stack);
41      return;
42    }
43  });
```

Рисунок 5.11 – Фрагмент коду: підключення до серверу MySQL

Для роботи з сервером MySQL використовують деякі драйвери. Найпопулярніші – це `mysql2` та `mysql`. Можна використати будь-який з них, однак по тестуванню перевагу надають `mysql2`. Після встановлення пакету створюю підключення. Для цього використовується метод `createConnection()`. У даний метод в якості параметрів вносяться налаштування підключення. Найчастіше використовують такі параметри:

- `host` (хост на якому запущено сервер);
- `user` (назва користувача сервера);
- `password` (пароль користувача)
- `database` (необов'язковий параметр, вказується ім'я бази даних).

З рис. 5.11 також видно, що використано метод `connect()`, його параметри вміщують помилку, яка повернеться при помилці підключення.

Оскільки в проєкті використовується Express та Node.js, то це означає, що можна реалізувати API в стилі REST. Дана архітектура встановлює правила взаємозв'язку з сервером та клієнтом методами HTTP. Ці методи відіграють важливу роль у процесі розробки, оскільки визначають, які операції виконуються на сервері [22]. До основних методів HTTP належать:

- GET;
- POST;
- PUT;
- DELETE.

Метод GET застосовують для отримання даних із сервера. Він корисний тим, що його можна викликає кілька разів підряд, його ще називають «безпечним методом». Даний метод підтримується всіма браузерами та легко застосовується. Відповіді на GET можуть кешуватись, а це покращує продуктивність всієї розробленої системи. Створення одного з багатьох методів GET продемонстровано на рис. 5.12. З рисунку видно, що методом GET запитується інформація про рейтинг, обрахунок якого проводиться в базі даних.

```

225 // рейтинг за вподобаннями
226 app.get("/likes_per_user", (req, res) => {
227   db.query(
228     `SELECT u.user_id, u.first_name, u.last_name, COUNT(l.like_id) AS total_likes
229     FROM Users u
230     LEFT JOIN Likes l ON u.user_id = l.user_id
231     GROUP BY u.user_id
232     ORDER BY total_likes DESC;
233     `,
234     (err, results) => {
235       if (err) return res.status(500).send(err);
236       res.json(results);
237     }
238   );
239 });

```

Рисунок 5.12 – Приклад використання методу GET

Якщо метод GET використовують для отримання даних з сервера, то метод POST навпаки – для відправки даних на сервер. POST може містити велику кількість даних, оскільки не має ніяких обмежень на об'єм. На відміну від GET, POST-запити безпечні для передачі конфіденційних даних. Однак вони менш продуктивні, оскільки потребують величезних об'ємів даних у процесі обробки. Використання одного з багатьох методів POST продемонстровано на рис. 5.13.

```

101 app.post("/register", async (req, res) => {
102   const { last_name, first_name, email, password } = req.body;
103
104   const userExistsQuery = "SELECT * FROM Users WHERE email = ?";
105   db.query(userExistsQuery, [email], async (err, results) => {
106     if (err) return res.status(500).json(err);
107     if (results.length > 0) {
108       return res
109         .status(400)
110         .json({ message: "Користувач з таким email вже існує!" });
111     }
112
113     const hashedPassword = await hashPassword(password);
114     const query =
115       "INSERT INTO Users (last_name, first_name, email, password) VALUES (?, ?, ?, ?)";
116     db.query(
117       query,
118       [last_name, first_name, email, hashedPassword],
119       (err, results) => {
120         if (err) return res.status(500).json(err);
121
122         const token = jwt.sign(
123           { userId: results.insertId, email },
124           SECRET_KEY,
125           {
126             expiresIn: "30d",
127           }
128         );
129         res
130           .status(201)
131           .json({ message: "Користувача успішно зареєстровано!", token });
132       }
133     );
134   });
135 });

```

Рисунок 5.13 – Приклад використання методу POST

З рис. 5.13 видно, що спочатку перевіряється в БД, чи користувач такий існує, якщо користувача не існує, то можна вводити дані і натискати «Зареєструватись». Після цього дані потрапляють в БД і користувач бачить сповіщення про успішну реєстрацію.

Метод PUT застосовують для оновлення наявних ресурсів. Також можна створювати нові ресурси, якщо наявних не достатньо чи їх не існує. Запити PUT теж можуть передавати великі об'єми даних, однак вони менше підтримуються браузерами, якщо брати в порівняння методи GET та POST. Використання одного з багатьох методів PUT продемонстровано на рис. 5.14. З рисунку можна побачити, що даний метод використовується для оновлення даних профілю користувача. Також використовується запит UPDATE до бази даних. По id перевіряється, який користувач виконав запит на оновлення даних, та які дані бажає оновити. Після цього виконується оновлення даних.

```

189 app.put(
190   "/user",
191   authenticateToken,
192   upload.single("profile_picture"),
193   (req, res) => {
194     const { first_name, last_name, email, bio } = req.body;
195     const userId = req.user.userId; // Використовуємо правильну властивість userId
196     const profile_picture = req.file ? req.file.filename : null;
197
198     // Обробка запиту на оновлення профілю
199     const query = `
200     UPDATE Users
201     SET first_name = ?, last_name = ?, email = ?, bio = ?, profile_picture = ?
202     WHERE user_id = ?
203     `;
204
205     const values = [first_name, last_name, email, bio, profile_picture, userId];
206
207     db.query(query, values, (err, result) => {
208       if (err) {
209         console.error("Error updating profile:", err);
210         return res.status(500).json({ message: "Error updating profile" });
211       }
212       res.status(200).json({ message: "Profile updated successfully" });
213     });
214   }
215 );

```

Рисунок 5.14 – Приклад використання методу PUT

Метод DELETE застосовують для видалення непотрібних даних із сервера. Запити типу DELETE є безпечні та прості в реалізації, оскільки дозволяють повторно видаляти дані і отримувати однаковий результат. Дані або будуть видалені, або буде отримане повідомлення, що вони відсутні. Також не слід забувати, що видалення не має відбуватись без належної перевірки. Використання одного з багатьох методів DELETE продемонстровано на рис. 5.15. З рисунку видно, що даний метод застосовується для видалення власних навиків із системи. При успіху чи помилці отримуються відповідні повідомлення.

```

329 app.delete("/skills/:id", authenticateToken, (req, res) => {
330   const { id } = req.params;
331
332   const query = "DELETE FROM skills WHERE skill_id = ?";
333
334   db.query(query, [id], (err, result) => {
335     if (err) {
336       console.error("Error deleting skill:", err);
337       return res.status(500).send({ message: "Server error" });
338     }
339
340     if (result.affectedRows > 0) {
341       return res.status(200).send({ message: "Skill deleted successfully" });
342     } else {
343       return res.status(404).send({ message: "Skill not found" });
344     }
345   });
346 });
347

```

Рисунок 5.15 – Приклад використання методу DELETE

5.5 Розробка клієнтської частини веб-системи

Оскільки проєктування клієнтської частини веб-орієнтованої системи обміну навичками та знаннями реалізується за допомогою веб-фреймворку Vue.js, то написання програмного коду буде задавалось своїм стилем. Застосування фреймворку Vue.js значно спрощує створення візуальної частини системи, оскільки вимагає від розробника, щоб він розбив дану систему на логічні частини та створив до кожної власний компонент. Рис. 5.16 демонструє структуру компонентів інтерфейсу веб-орієнтованої системи обміну навичками та знаннями.

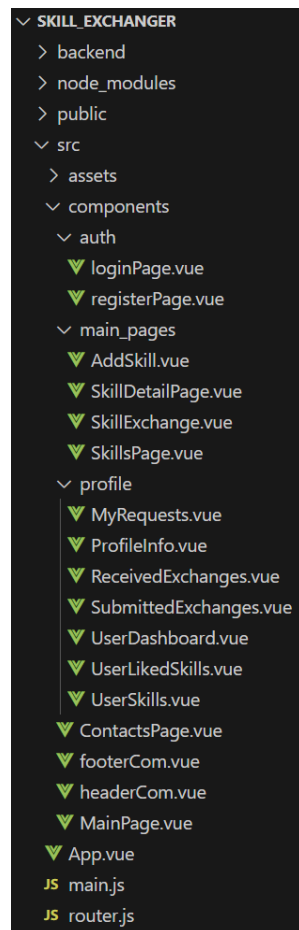


Рисунок 5.16 – Фрагмент структури проекту клієнтської частини

Якщо узагальнити, то структура побудованого проекту за допомогою фреймворку Vue.js складається з наступних елементів:

- components (зберігає компоненти, які можна повторно використати, та сторінки веб-системи);
- App.vue (кореневий компонент);
- main.js (файл для підключень основних бібліотек);
- router.js (файл налаштувань маршрутів, використовується для керування навігацією у веб-системі).

Можна розглянути конфігурацію файлу router.js. Він виконує важливе значення, оскільки саме в ньому застосовується імпортування модуля vue-router та створених компонентів (сторінок веб-системи). Модуль vue-router достатньо важливий, оскільки забезпечує обробку маршрутизації без надсилання нових

запитів до серверу. Фрагмент програмного коду, що міститься в router.js продемонстровано на рис. 5.17.

```

1  import { createRouter, createWebHashHistory } from "vue-router";
2  import MainPage from "../components/MainPage.vue";
3  import ContactsPage from "../components/ContactsPage.vue";
4  import Login from "../components/auth/loginPage.vue";
5  import Register from "../components/auth/registerPage.vue";
6  import UserDashboard from "../components/profile/UserDashboard.vue";
7  import AddSkill from "../components/main_pages/AddSkill.vue";
8  import SkillsPage from "../components/main_pages/SkillsPage.vue";
9  import SkillDetailPage from "../components/main_pages/SkillDetailPage.vue";
10
11  function isAuthenticated() {
12    return !!localStorage.getItem("token");
13  }
14
15  const router = createRouter({
16    history: createWebHashHistory(),
17    routes: [
18      { path: "/", component: MainPage },
19      { path: "/contacts", component: ContactsPage },
20      {
21        path: "/register",
22        component: Register,
23        beforeEnter: (to, from, next) => {
24          if (isAuthenticated()) {
25            next("/user_dashboard");
26          } else {
27            next();
28          }
29        },
30      },
31    ],
32  });

```

Рисунок 5.17 – Фрагмент програмного коду з файлу router.js

Також варто звернути увагу на конфігурацію файлу App.vue, в ньому виконується формування структури будь-якої сторінки системи. Він містить стандартну структуру веб-фреймворку Vue.js, теги: template, script та style. В тезі template наведено інші теги та код HTML, в тезі script – логіка роботи компонента, а в style – застосовані стилі. Фрагмент програмного коду, що міститься в App.vue, продемонстровано на рис. 5.18.

```

1  <template>
2    <headerCom />
3    <div class="margin">
4      <router-view></router-view>
5    </div>
6    <footerCom />
7  </template>
8
9  <script>
10 import headerCom from "../components/headerCom.vue";
11 import footerCom from "../components/footerCom.vue";
12
13 export default {
14   name: "App",
15   components: {
16     headerCom,
17     footerCom,
18   },
19 };
20 </script>
21
22 <style>
23 @font-face {
24   font-family: "Lato";
25   src: url("assets/fonts/Lato-Regular.ttf") format("ttf");
26   font-weight: 400;
27   font-style: normal;
28 }
29
30 @font-face {
31   font-family: "Lato";
32   src: url("assets/fonts/Lato-Bold.ttf") format("ttf");
33   font-weight: 700;
34   font-style: normal;
35 }
36

```

Рисунок 5.18 – Фрагмент програмного коду з файлу App.vue

З рис. 5.18 видно стиль задання компонентів фреймворку Vue.js. Всі інші компоненти, які відповідають сторінці реєстрації, авторизації і тд., містять схожу структуру та використовують аналогічні теги.

Висновки до розділу 5

У цьому розділі магістерської дисертації було розглянуто основні сценарії використання системи, описано детально архітектуру за допомогою розроблених основних UML-діаграм, здійснено вибір технологій, який базувався на аналізі

сучасних вимог до веб-орієнтованих систем та коротко наведено особливості проєктування серверної та клієнтської частин.

Наведені сценарії використання системи продемонстрували широкі функціональні можливості та показали як саме користувачі будуть взаємодіяти з розробленою системою. Планування архітектури та побудова UML-діаграм дозволили візуально продемонструвати взаємодію користувача із системою. UML-діаграми – це «мова», яка дозволяє легко обговорити та продемонструвати роботу системи, як технічним розробникам, так і менеджерам та тестувальникам. Також документація, яку можна створити на основі сценаріїв використання чи наведеної архітектури за допомогою діаграм, у майбутньому спростить додавання нових функцій чи інтеграцію спроектованої системи з іншими платформами.

Здійснений вибір технологій дозволив проаналізувати найпопулярніші мови програмування, розглянути платформи для розробки, доступні фреймворки, побачити їх особливості, переваги та недоліки. У ході такого аналізу було вибрано мову програмування JavaScript, яку можна використати як для розробки серверної, так і клієнтської частин. Для проєктування серверної частини також вибрано середовище Node.js, яке зарекомендувало себе як високопродуктивна платформа з підтримкою асинхронної обробки запитів. Проєктування клієнтської частини виконане за допомогою фреймворк Vue.js, завдяки його простоті при розробці базових можливостей.

6 ІНТЕРФЕЙС КОРИСТУВАЧА ТА UX/UI ДИЗАЙН

6.1 Спроектований інтерфейс

Для входу в спроектовану веб-орієнтовану систему обміну навичками та знаннями необхідно спочатку запустити сервери. Користувач отримує посилання на систему та відкриває його у веб-браузері. Перейшовши за посилання, користувач потрапляє на головну сторінку розробленої веб-системи (рис. 6.1).

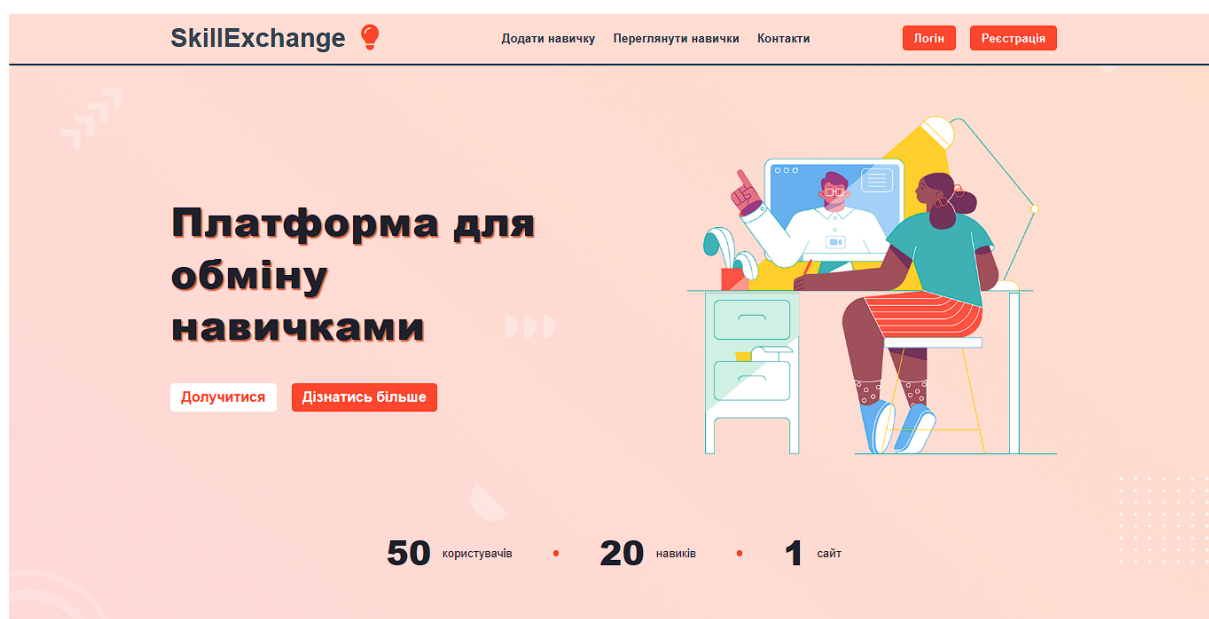


Рисунок 6.1 – Фрагмент головної сторінки веб-системи

На головній сторінці можна побачити наступні розділи:

- Додати навичку
- Переглянути навички;
- Контакти;
- Логін;
- Реєстрація.

Оскільки зацікавлений користувач увійшов в систему вперше та не має облікового запису, то йому поки що не доступні всі можливості платформи. Неавторизований користувач може переглянути головну сторінку, сторінку з виставленими навичками на обмін, сторінку контактів та навіть побачити етап

додавання навички, однак він отримує сповіщення про помилку, оскільки потрібно увійти в обліковий запис.

Для одержання свого облікового запису користувач переходить на сторінку «Реєстрація», вводить свої реєстраційні дані та натискає «Продовжити» (рис. 6.2).

SkillExchange  [Додати навичку](#) [Переглянути навички](#) [Контакти](#) [Логін](#) [Реєстрація](#)

Реєстрація профілю

Прізвище користувача

Ім'я користувача

Email

Пароль

Підтвердіть пароль

[Продовжити](#)

Про нас
 SkillExchange — платформа для обміну навичками, що допомагає користувачам розвивати свої таланти, ділитися досвідом та

Головні посилання
[Головна](#)
[Додати навик](#)
[Переглянути навички](#)

Контакти
 +38 (012) 345-6789
 info@SkillExchange.com
 Київ, Україна

Ми в соцмережах
   

Рисунок 6.2 – Форма реєстрації

Після надсилання форми користувач отримує повідомлення про успіх (рис. 6.3) та має змогу здійснити авторизацію в системі, так як отримав свій обліковий запис. Завершений процес реєстрації перенаправляє користувача на сторінку «Логін». Користувач вводить свої облікові дані та натискає «Продовжити». Якщо дані введені коректно, то можна побачити спливаюче вікно, яке підтверджує успішний вхід (рис. 6.4).

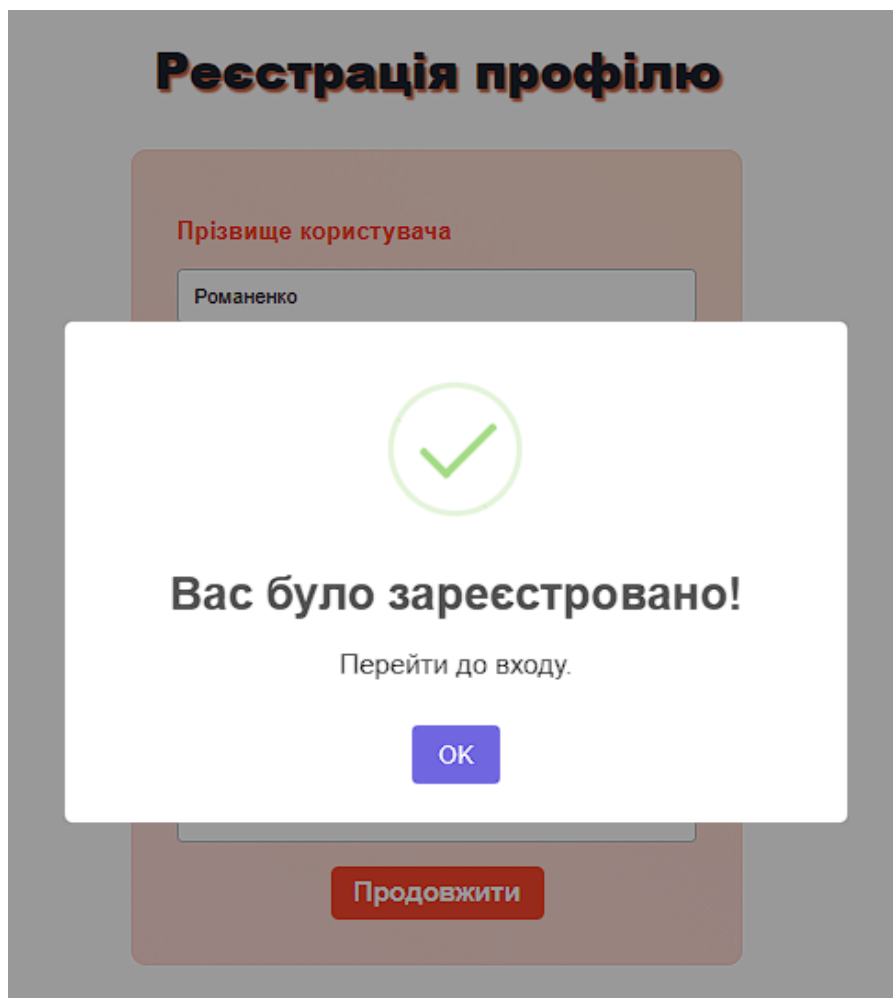


Рисунок 6.3 – Успішний процес реєстрації

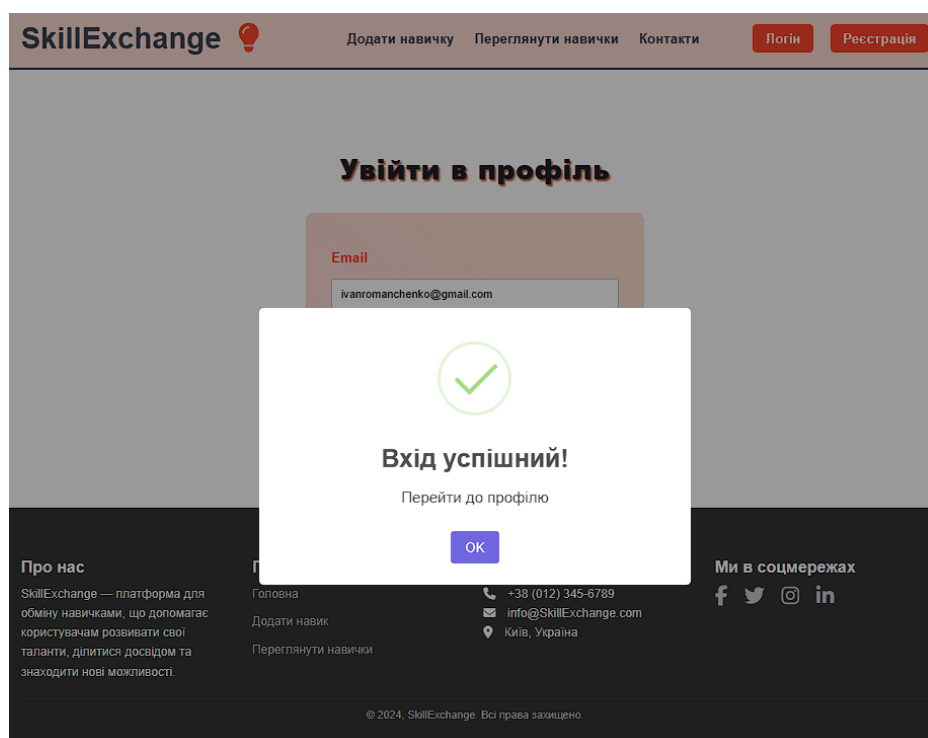


Рисунок 6.4 – Успішний вхід до облікового запису

При успішному вході в особистий кабінет користувач бачить свою особисту інформацію (рис. 6.5).

SkillExchange Додати навичку Переглянути навички Контакти Кабінет

Ваш Кабінет

Інформація про користувача

Мої навички

Мої заявки

Вподобані навички

Вийти

Редагувати профіль

Ім'я
Іван

Прізвище
Романченко

Email
ivanromanchenko@gmail.com

Біографія

Рисунок 6.5 – Вигляд особистого кабінету користувача

Користувач в особистому кабінеті може:

- перейти на вкладку «Мої навички» та переглянути створені;
- перейти на вкладку «Мої заявки» та побачити надіслані/отримані;
- відкрити вкладку «Вподобані навички» та розглянути цікаві пропозиції, які користувач зберіг;
- натиснути на кнопку «Вийти» (вихід з облікового запису).

Також користувач має можливість змінити особисту інформацію (редагувати

профіль). Вигляд особистого кабінету з доданою фотографією та біографією продемонстровано на рис. 6.6.

SkillExchange

Додати навичку Переглянути навички Контакти **Кабінет**

Інформація про користувача

Мої навички

Мої заявки

Вподобані навички

Вийти

Редагувати профіль

Ім'я
Іван

Прізвище
Романченко

Email
ivanromanchenko@gmail.com

Біографія
Програміст з 5-річним досвідом.

Зображення профілю
Choose File | No file chosen

Зберегти

Рисунок 6.6 – Вигляд наповненої сторінки

Щоб приступити до процесу обміну навичкою, користувач повинен перейти на вкладку «Додати навичку» та вписати інформацію. На цій сторінці (рис. 6.7) користувач додає назву навички, її детальний опис, вибирає категорію, до якої вона відноситься (за допомогою випадаючого списку), та має можливість прикріпити зображення. Після внесення всієї інформації натискає «Додати навичку». Переглянути додані навички можна в особистому кабінеті на вкладці «Мої навички» (рис. 6.8).

SkillExchange 

Додати навичкуПереглянути навичкиКонтакти

Кабінет

Додати навичку

Назва навички

Опис

Категорія

Зображення навички

Вибрати файл

Додати навичку

Рисунок 6.7 – Вигляд сторінки «Додати навичку»

SkillExchange 

Додати навичкуПереглянути навичкиКонтакти

Кабінет

Ваш Кабінет

Інформація про користувача

Мої навички

Мої заявки

Вподобані навички

Вийти

JavaScript



JavaScript

Навичка програмування на JavaScript

Кількість вподобань: 0

Створена: 10/1/2024

РедагуватиВидалити

Рисунок 6.8 – Вигляд вкладки «Мої навички» після додавання необхідної навички

Оскільки користувач створив кілька своїх навичок, то після цього він уже може вибрати бажану навичку на вкладці «Переглянути навички» (рис. 6.9). Далі можна перейти на сторінку навички, натиснути кнопку «Запропонувати обмін навичками», вибрати одну зі своїх навичок для обміну та підтвердити обмін натисканням кнопки «Подати запит на обмін» (рис. 6.10).

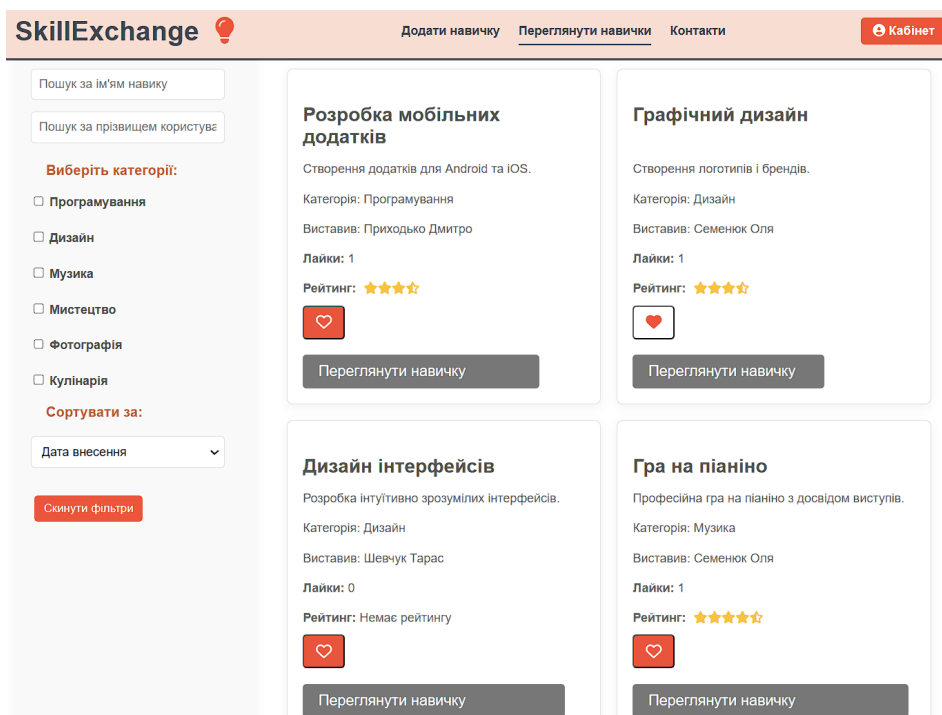


Рисунок 6.9 – Вигляд сторінки з розміщеними навичками

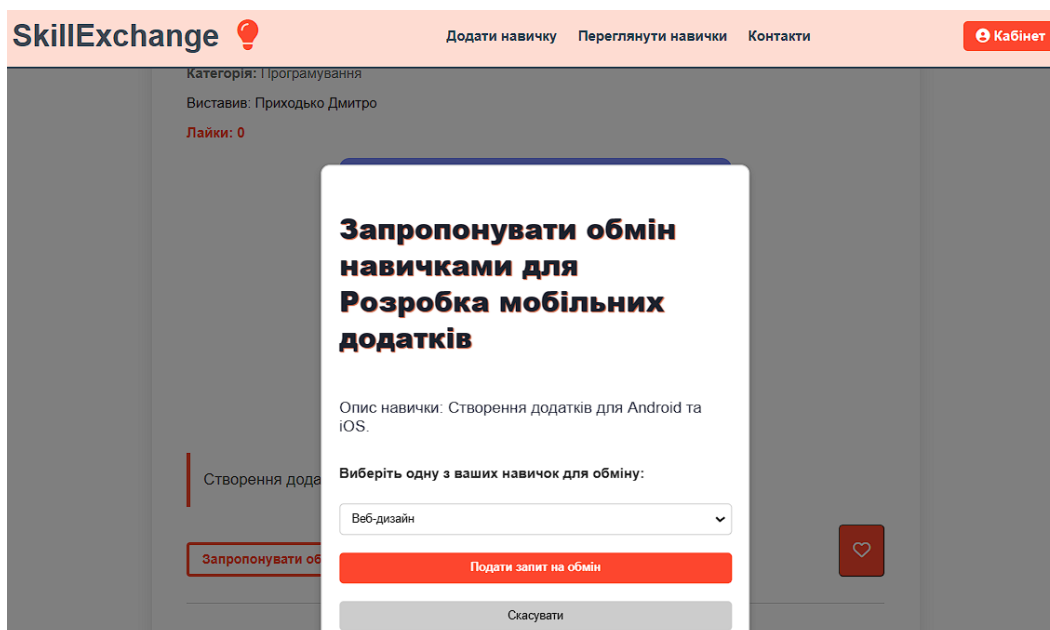


Рисунок 6.10 – Процес подання заявок на обмін навичками

В особистому кабінеті є можливість переглянути надіслані та отримані заявки на обмін (рис. 6.11-6.12).

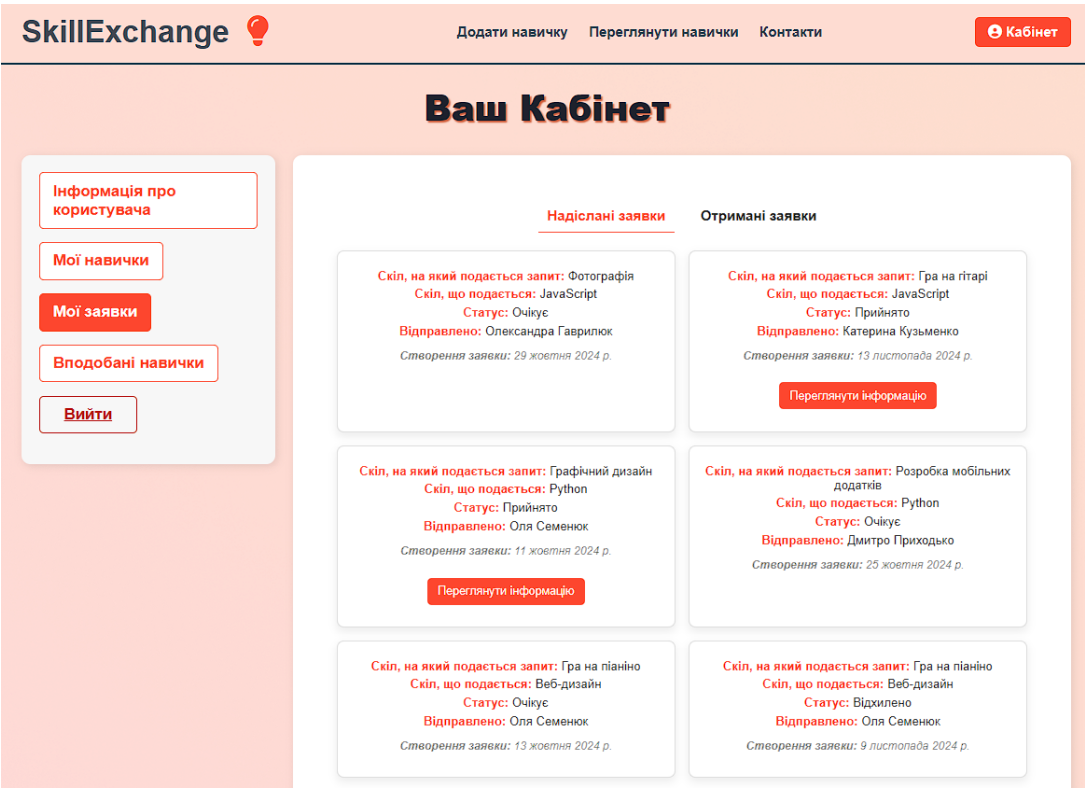


Рисунок 6.11 – Вигляд сторінки з надісланими заявками

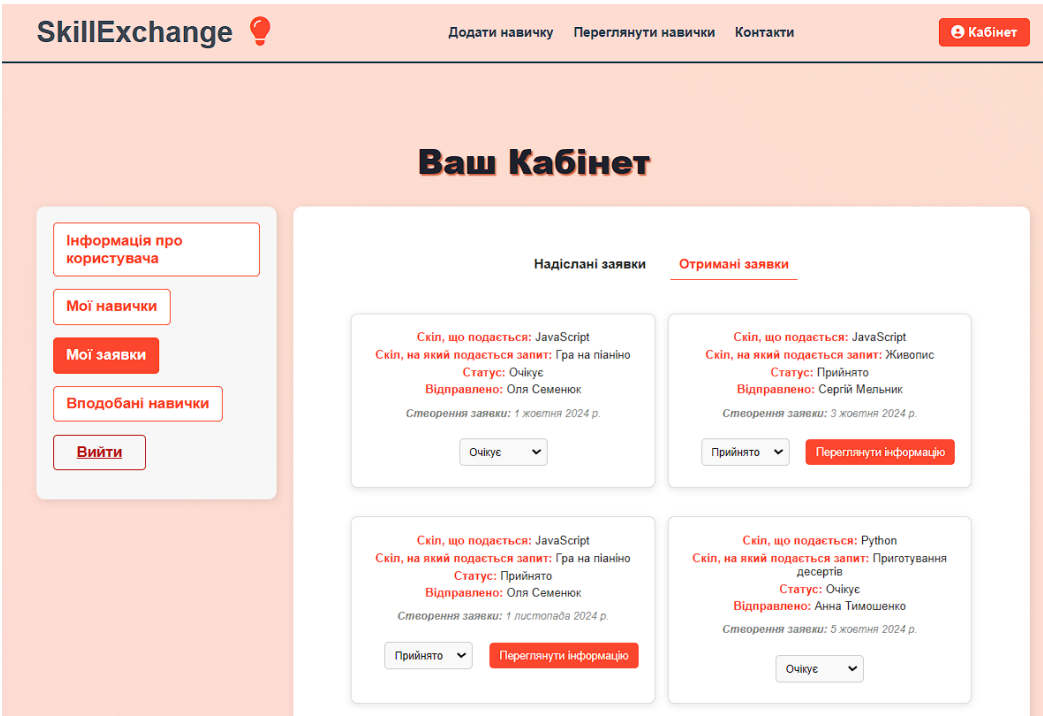


Рисунок 6.12 – Вигляд сторінки з отриманими заявками

На рис. 6.11 можна побачити, що кожна надіслана заявка має свій статус. Користувач, який отримав запит на обмін, може прийняти заявку, а може відхилити. Заявки, по яких ще не прийняті рішення, мають статус «Очікує». З рис. 6.12 видно, що аналогічно працюють отримані заявки. Користувач може прийняти заявку або відхилити. Заявки, які ще не було розглянуті теж мають статус «Очікує». Коли заявка підтверджена, то вона отримує кнопку «Переглянути інформацію».

На сторінці навичок користувач має можливість фільтрувати навички за категоріями, сортувати за датою додавання, рейтингом чи кількістю лайків (рис. 6.13). Також користувач може поставити «Лайк» вподобаній навичці.

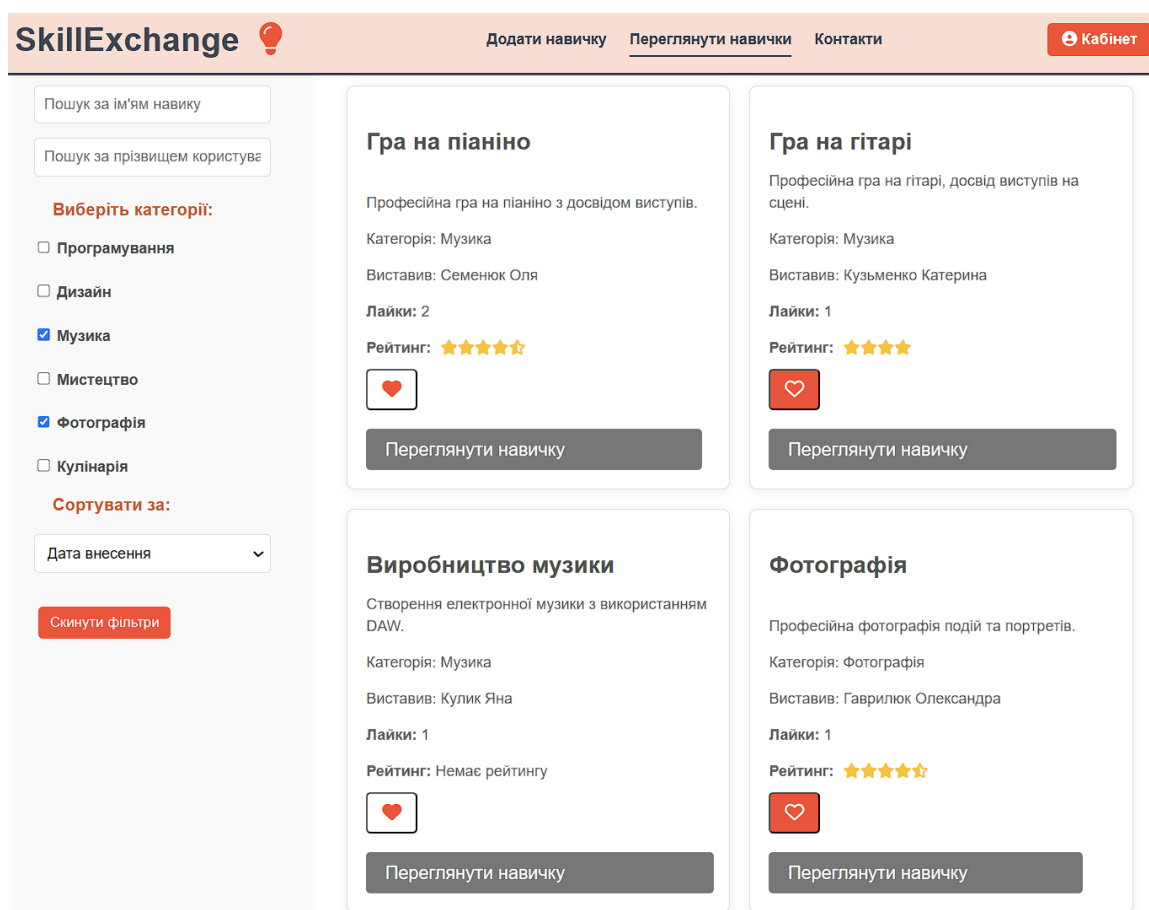


Рисунок 6.13 – Можливість фільтрування та сортування

Вподобані навички можна переглядати у своєму особистому кабінеті, на вкладці вподобані навички (рис. 6.14).

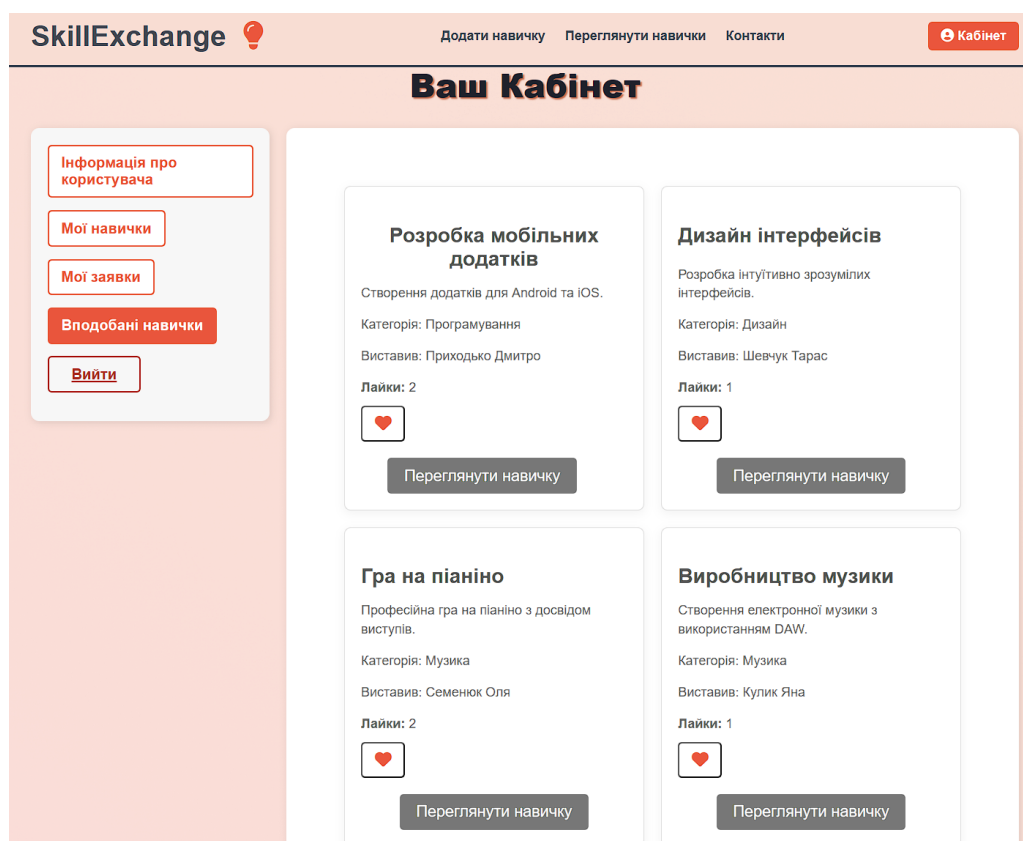


Рисунок 6.14 – Вигляд вкладки «Вподобані навички»

Користувач також може відредагувати власну навичку (рис. 6.15) або зовсім видалити її (рис. 6.16).

Редагувати навик

Назва навика:

JavaScript

Опис:

Навичка програмування на JavaScript

Зображення навички:

Choose File No file chosen

Зберегти

Закрити

Рисунок 6.15 – Реалізована можливість редагування навичку

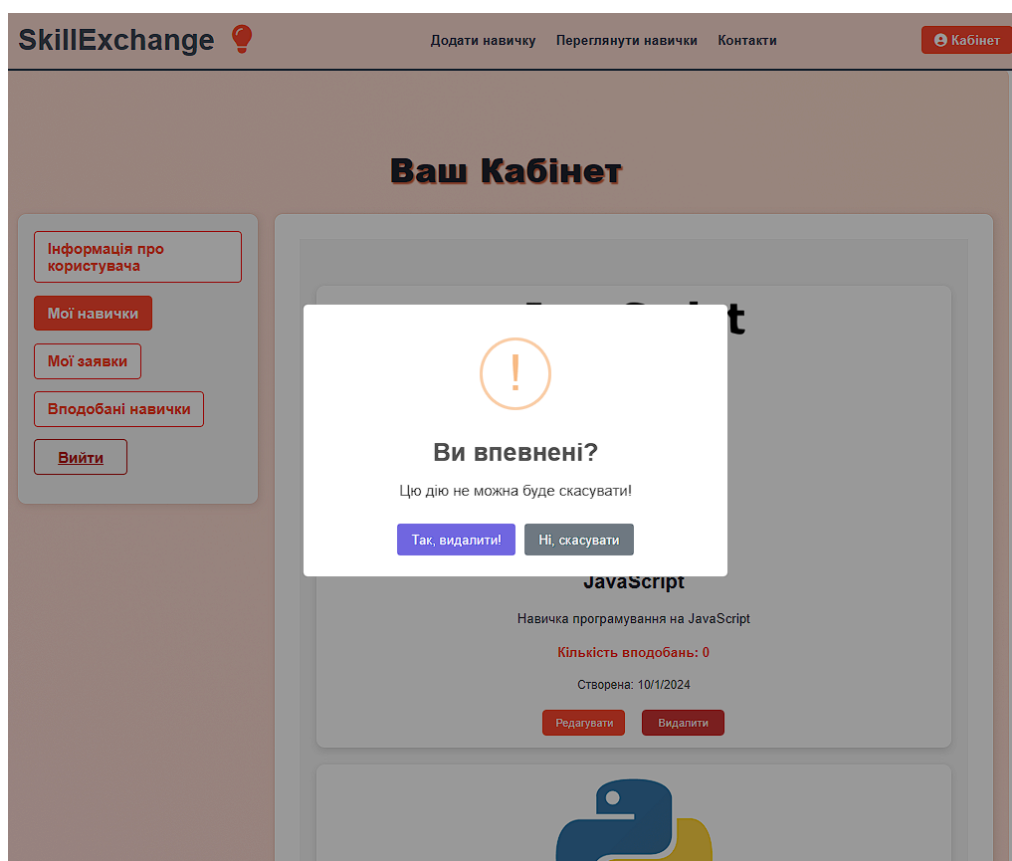


Рисунок 6.16 – Реалізована можливість видалення навичку

Також після завершення обміну навичками, користувач може перейти на сторінку завершеного навичку та залишити відповідний коментар чи виставити оцінку для формування рейтингу (рис. 6.17).

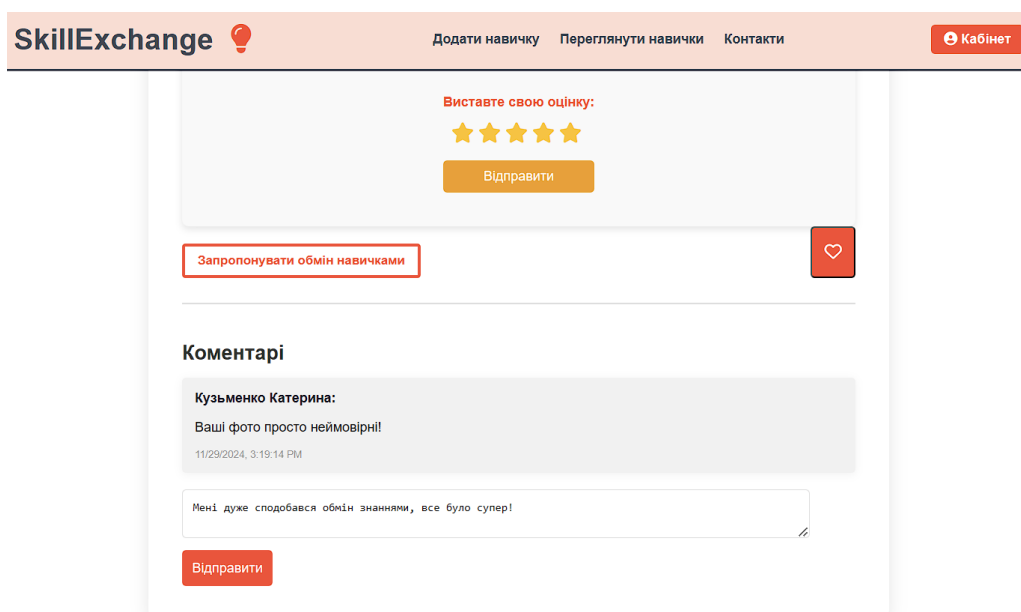


Рисунок 6.17 – Реалізовані можливості оцінки навичку та допису коментарів

Якщо у процесі користування сервісом у користувача виникли питання чи пропозиції, він може зв'язатись з адміністрацією за допомогою відповідної форми на сторінці «Контакти» (рис. 6.18).



SkillExchange 

Додати навичку Переглянути навички Контакти [Кабінет](#)

Є ПИТАННЯ?

Зв'яжіться з нами, ми завжди раді допомогти!

Ваше прізвище:

Ваше ім'я:

Ваш email:

Повідомлення:

Надіслати

Про нас

SkillExchange — платформа для обміну навичками, що допомагає користувачам розвивати свої таланти, ділитися досвідом та знаходити нові можливості.

Головні посилання

Головна

Додати навик

Переглянути навички

Контакти

+38 (012) 345-6789

info@SkillExchange.com

Київ, Україна

Ми в соцмережах

[f](#) [t](#) [i](#) [in](#)

© 2024, SkillExchange. Всі права захищено.

Рисунок 6.18 – Реалізована можливість зв'язку з адміністрацією

Для зручності всі надіслані форми потрапляють не тільки в базу даних, а також приходять в телеграм-бот адміністратора платформи (рис. 6.19).

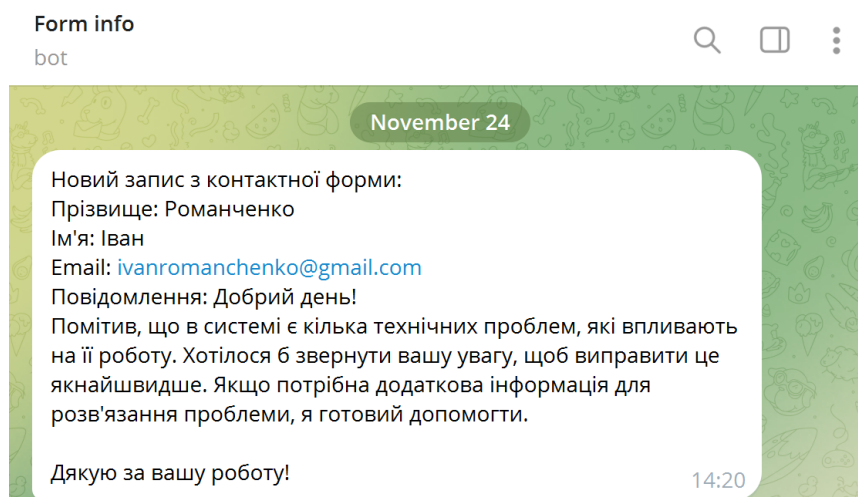


Рисунок 6.19 – Вигляд залишеного повідомлення у зворотній формі

6.2 Адаптивність та кросплатформенність

В основному, користувачі, які використовують веб-додатки та будь-які різноманітні веб-платформи, заходять на них за допомогою мобільного пристрою (з телефону або планшета). І це не пусті слова, дане твердження підтверджене статистичними даними. Оскільки веб-система повинна заохочувати користувача використовувати її знову й знову, то вона має правильно відображатись на будь-якому девайсі. Саме для цього придумали адаптивний дизайн.

Адаптивність веб-додатку – це зручний та коректний для користувача вигляд сторінок, який ніяким чином не спотворює свій вміст, незважаючи на розміри екрану, з якого даний вміст переглядають. Адаптивна розробки веб-системи має багато переваг, оскільки дозволяє заощадити кошти на розробці інших версій та їх просуванні на ринку [23].

Отже, для перевірки адаптивності необхідно запустити створену систему на різних типах екранів. Для демонстрації адаптивності можна порівняти відкриття головної сторінки, сторінки з виставленими навичками та особистий кабінет користувача на мобільному пристрої та планшеті.

Вигляд головної сторінки веб-системи обміну навичками та знаннями, яка відкрита на пристрої iPhone 14 Pro Max, представлено на рис. 6.20.

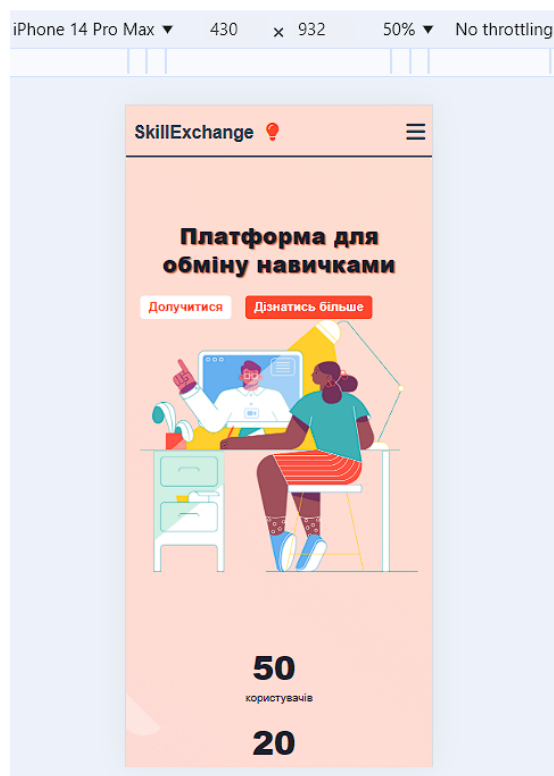


Рисунок 6.20 – Вигляд головної сторінки на пристрої iPhone 14 Pro Max

Вигляд головної сторінки веб-системи обміну навичками та знаннями, яка відкрита на пристрої iPad Pro, представлено на рис. 6.21.

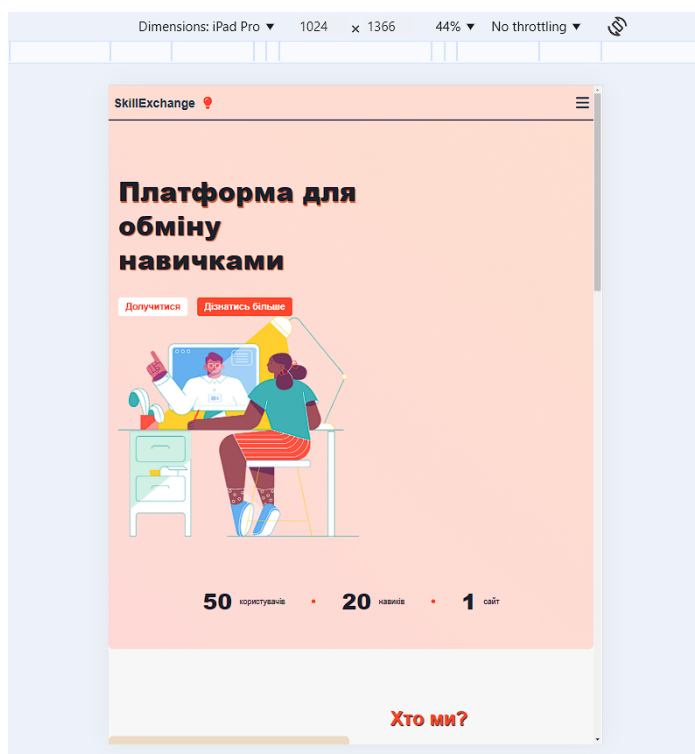


Рисунок 6.21 – Вигляд головної сторінки на пристрої iPad Pro

Вигляд сторінки з навичками веб-системи обміну навичками та знаннями, яка відкрита на пристрої iPhone 14 Pro Max, представлено на рис. 6.22.

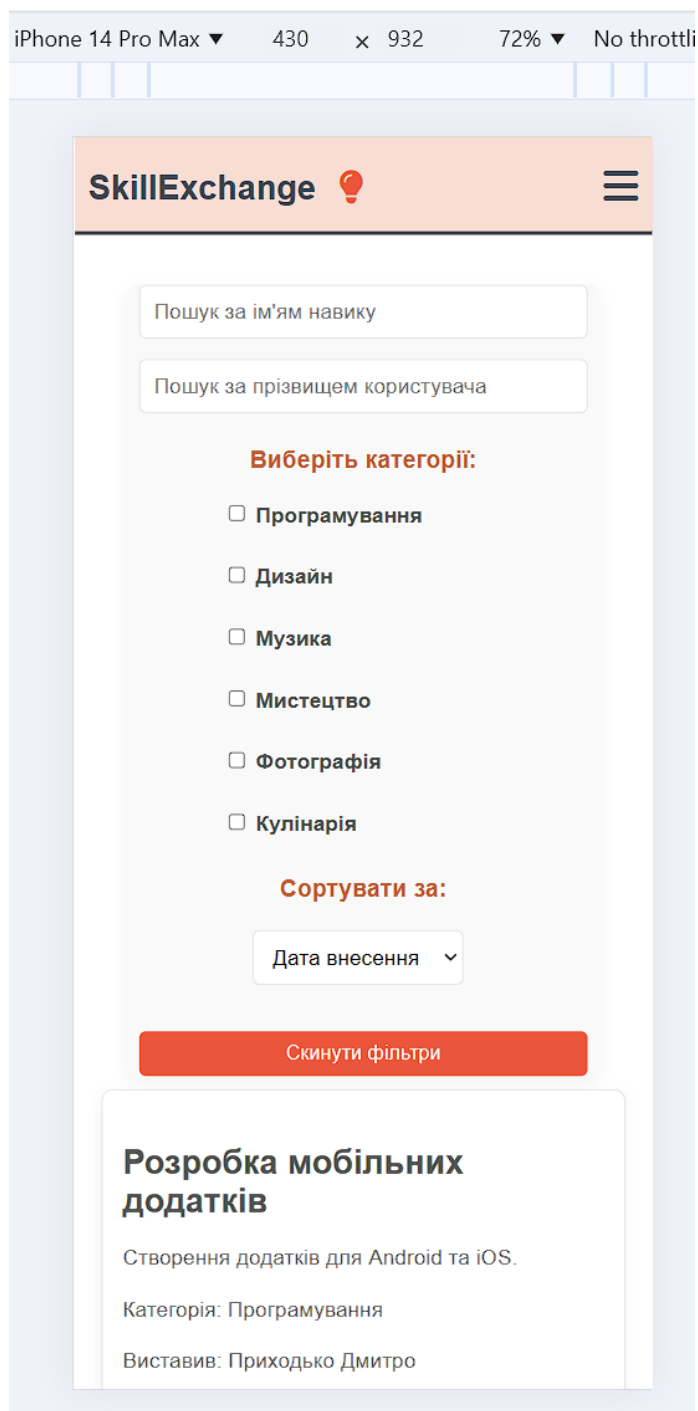


Рисунок 6.22 – Вигляд сторінки з навичками на пристрої iPhone 14 Pro Max

Вигляд сторінки з навичками веб-системи обміну навичками та знаннями, яка відкрита на пристрої iPad Pro, представлено на рис. 6.23.

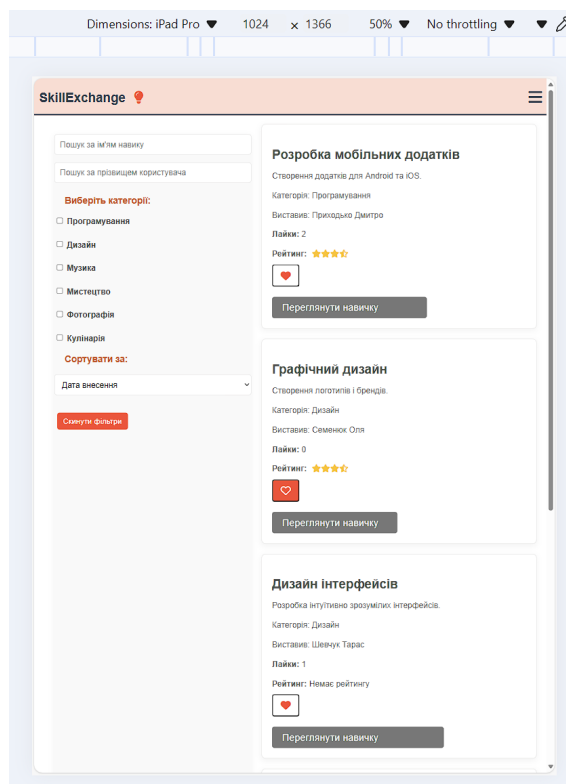


Рисунок 6.23 – Вигляд сторінки з навичками на пристрої iPad Pro

Вигляд особистого кабінету користувача веб-системи обміну навичками та знаннями, яка відкрита на пристрої iPhone 14 Pro Max та iPad Pro, представлено на рис. 6.24-6.25.

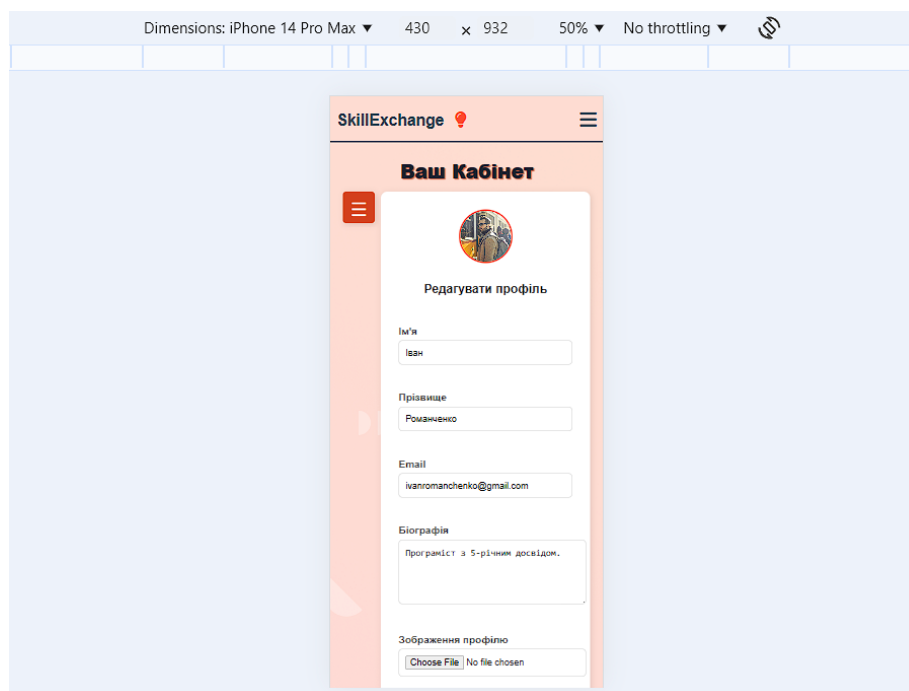


Рисунок 6.24 – Вигляд кабінету користувача на пристрої iPhone 14 Pro Max

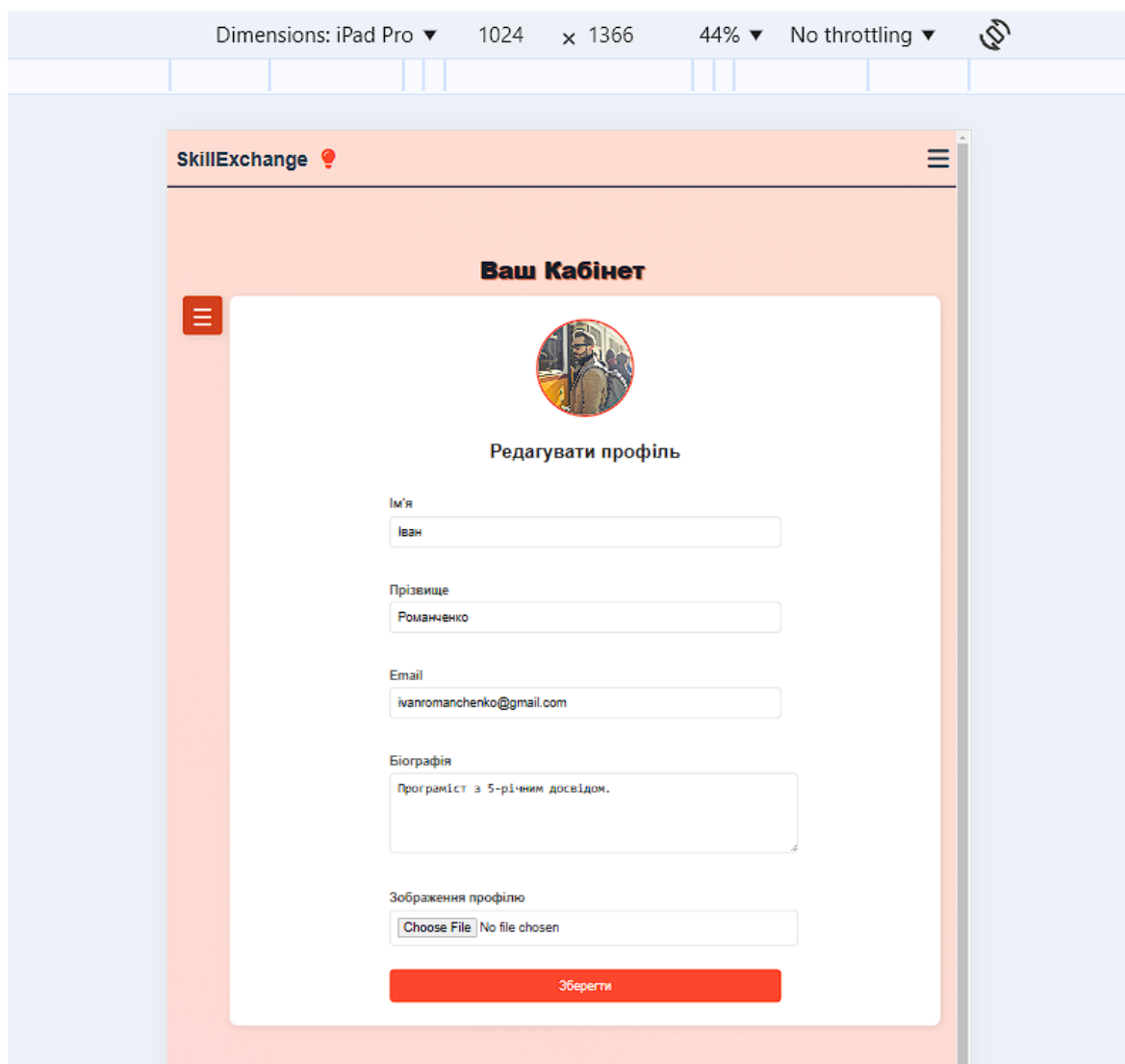


Рисунок 6.25 – Вигляд кабінету користувача на пристрої iPad Pro

Розглянувши рис. 6.20-6.25, можна зробити висновок, що адаптивність працює добре, оскільки всі найбільш навантажені сторінки відкриваються правильно та жодний аномальних зміщень блоків не помічено.

Також варто перевірити роботу веб-системи у трьох різних браузерях, наприклад, Opera (рис. 6.26), Edge (рис. 6.27), та Safari (рис. 6.28). Тестування проводитиметься на вкладці «Контакти».

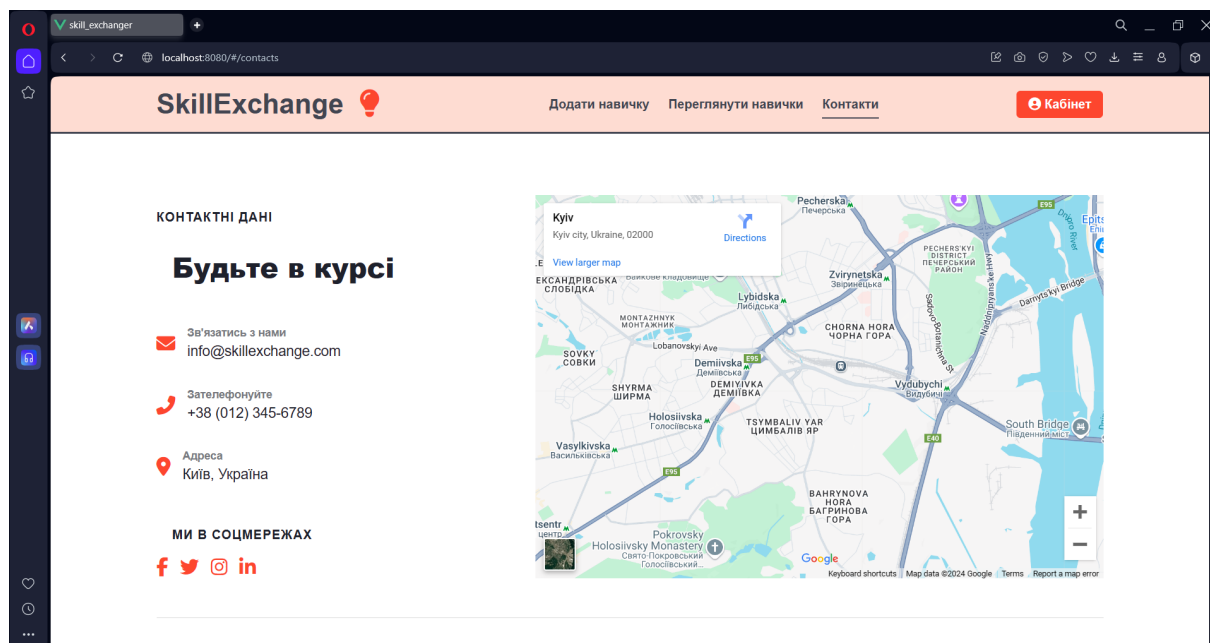


Рисунок 6.26 – Вигляд вкладки «Контакти» відкритою в браузері Опера

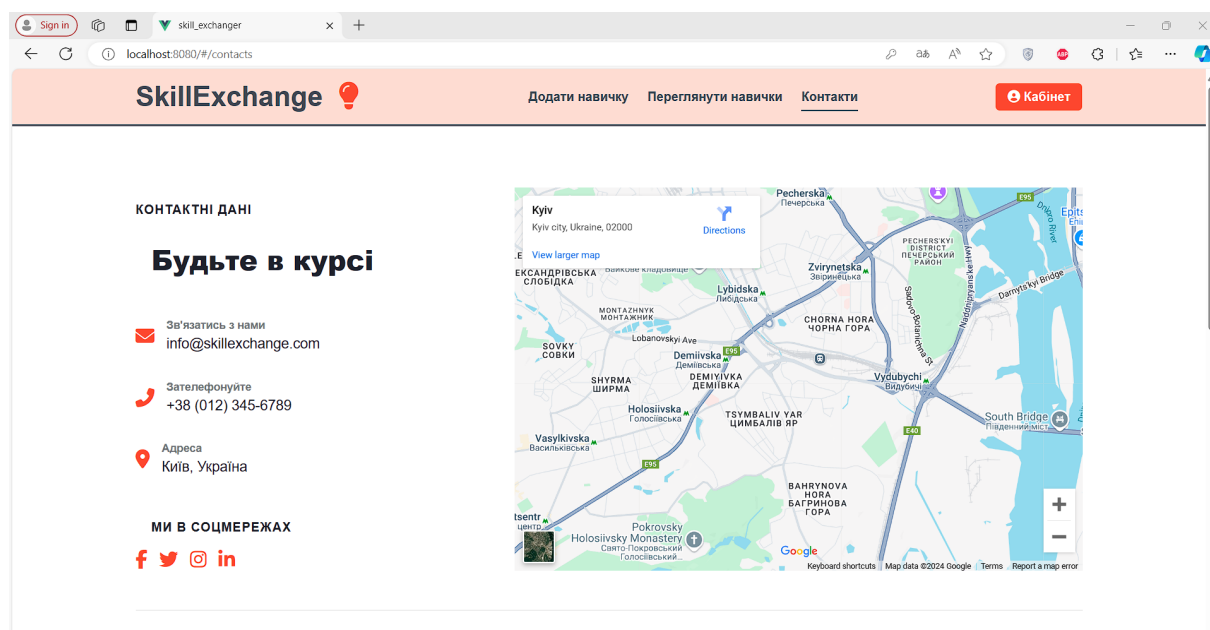


Рисунок 6.27 – Вигляд вкладки «Контакти» відкритою в браузері Edge

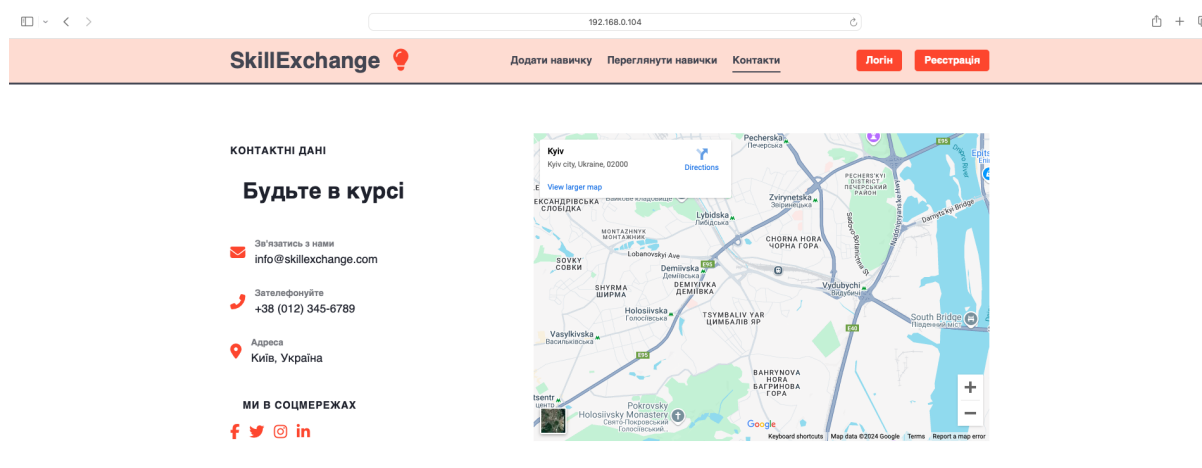


Рисунок 6.28 – Вигляд вкладки «Контакти» відкритої в браузері Safari

Висновки до розділу 6

У даному розділі магістерської дисертації було проаналізовано спроектований інтерфейс та його адаптивність. Також було проведено огляд широкого спектру функціональних можливостей, які надає розроблена веб-орієнтована система. До основних можливостей системи входить: реєстрація, авторизація, заповнення особистого профілю, створення навичок, пошук необхідних навичок, надсилання запитів на обмін навичками, можливість коментування навичок та зворотного зв'язку з адміністрацією.

Важливим завершення цього розділу є аналіз питань адаптивності та кросплатформенності. Адаптивна розробка веб-системи досить важлива та має ряд переваг, оскільки дозволяє заощадити кошти на розробці інших версій та її просувань на ринку.

Результатом роботи є розроблена адаптивна версія веб-орієнтованої системи обміну навичками та знаннями, яка продемонструвала високу гнучкість та швидкодію на різних пристроях. Завдяки адаптивній розробці система забезпечить комфортну роботу, як на персональних комп'ютерах (ПК), ноутбуках, так і на мобільних пристроях, в тому числі планшетах. Дана особливість покращує доступність системи та користувацький досвід.

7 ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМИ

7.1 Тестування функціональності

Для того, щоб перевірити веб-систему на відповідність до всіх вимог, – створено тестові випадки. У веб-орієнтованій системі обміну навичками та знаннями будуть тестуватись критичні недоліки та перевірятись основні функціональні можливості за допомогою димового методу тестування.

Метод димового тестування використовується тоді, коли отримано нову версію системи і її можуть вважати нестабільною. Метою даного типу тестування є здійснення перевірки надійності та стабільності програмної веб-орієнтованої системи [24].

Тестові випадки тестування димовим методом наведено у табл. 7.1.

Таблиця 7.1 – Тестові випадки

№	Назва сценарію	Передумова	Сценарій дії	Результат
1	Можливість реєстрації неавторизованого користувача у веб-системі	Неавторизований користувач без облікового запису	1. Неавторизований користувач натискає на кнопку «Реєстрація». 2. Неавторизований користувач заповнює поля. 3. Неавторизований користувач натискає на кнопку «Продовжити».	Повідомлення про успіх
2	Можливість авторизації у веб-системі у ролі типового користувача	У неавторизованого користувача існує обліковий запис	1. Неавторизований користувач натискає на кнопку «Логін». 2. Неавторизований користувач заповнює необхідні поля і натискає на кнопку «Продовжити»	Повідомлення про успіх

№	Назва сценарію	Передумова	Сценарій дії	Результат
3	Можливість переглянути виставлені навички	Неавторизований/авторизований користувач	1. Неавторизований/авторизований користувач переходить на сторінку «Переглянути навички»	Відкривається сторінка з виставленими навичками
4	Можливість додавати навички	Авторизований користувач (надалі просто користувач)	1. Користувач переходить на сторінку «Додати навичку» 2. Користувач заповнює форму та натискає кнопку «Додати навичку»	Повідомлення про успіх
5	Можливість переглядати додані навички	Користувач існує	1. Користувач переходить на вкладку «Кабінет» 2. Користувач натискає на кнопку «Мої навички»	Відкривається сторінка з опублікованими навичками
6	Можливість редагувати додані навички	Користувач існує	1. Користувач переходить на вкладку «Кабінет» 2. Користувач натискає на кнопку «Мої навички» 3. Користувач обирає потрібну навичку та натискає на кнопку «Редагувати» 4. Користувач змінює необхідні дані та натискає кнопку «Зберегти»	Повідомлення про успіх

№	Назва сценарію	Передумова	Сценарій дії	Результат
7	Можливість видаляти додані навички	Користувач існує	1. Користувач переходить на вкладку «Кабінет» 2. Користувач натискає на кнопку «Мої навички» 3. Користувач обирає потрібну навичку та натискає на кнопку «Видалити» 4. Користувач отримує попередження та натискає кнопку «Так, видалити!»	Повідомлення про успіх
8	Можливість пропонувати власні навички на обмін	Користувач існує	1. Користувач переходить на сторінку «Переглянути навички» 2. Користувач обирає деяку навичку 3. Користувач натискає кнопку «Запропонувати обмін навичками» 4. Користувач обирає власну навичку та натискає кнопку «Подати запит на обмін»	Повідомлення про успіх

№	Назва сценарію	Передумов а	Сценарій дії	Результат
9	Можливість залишити коментар до опублікованих навичок	Користувач існує	1. Користувач переходить на сторінку «Переглянути навички" 2. Користувач обирає деяку навичку 3. Користувач прогортає інформацію вниз та бачить поле «Напишіть коментар» 4. Користувач залишає коментар та натискає кнопку «Відправити»	На сторінці з вибраним навиком з'являється залишений коментар
10	Користувач має можливість оцінити навичку	Користувач існує	1. Користувач переходить на сторінку «Переглянути навички" 2. Користувач обирає деяку навичку та натискає кнопку «Лайк»	Кількість лайків залишена на навичці збільшилась
11	Користувач має можливість переглянути вподобані навички у своєму кабінеті	Користувач існує	1. Користувач переходить на вкладку «Кабінет» 2. Користувач натискає на кнопку «Вподобані навички»	Відкрилась вкладка з вподобаними навичками

Висновки до розділу 7

У цьому розділі магістерської дисертації було перевірено систему на відповідність до поставлених вимог. У процесі тестування веб-орієнтованої системи обміну навичками та знаннями проводилась перевірка на наявні критичні недоліки. Також було перевірено правильне функціонування основних функціональних можливостей системи. Тестування веб-орієнтованої системи було проведено за допомогою димового методу, оскільки даний метод зручно використовувати для версій систем, які тільки-но були розроблені та до яких можуть виникнути питання щодо їх стабільності. Результатом роботи є розроблена таблиця тестових випадків, яка допомогла здійснити перевірку системи та підтвердити працездатність всіх її функцій, а також стабільність та надійність.

8 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

8.1 Опис ідеї проєкту

Варто почати з того, що проєкт є достатньо перспективний, оскільки подібні системи існують в недостатній кількості та в поганій якості виконання. Для початку необхідно сформулювати ідею самого стартап-проєкту, яка включатиме зміст ідеї, напрями застосування та основні вимоги, які отримає користувач у ході використання. Даний опис зручно реалізувати за допомогою побудованої табл. 8.1.

Таблиця 8.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вимоги до користувача
Зміст ідеї полягає у створенні веб-орієнтованої системи, яка дозволить користувачам об'єднувати зусилля в отриманні нових знань, досвіду та навичок. Користувачі зможуть шукати цікаві навички, створювати свої та проводити обмін навичками з іншими користувачами системи. Також користувачі зможуть обирати навички на основі їх рейтингу, додавати у список вподобаний, залишати коментарі.	1. Самонавчання студентів. Студенти зможуть безкоштовно отримувати нові знання та досвід.	1. Студент (користувач) має мати бажання вчитись чомусь новому та запропонувати свою навичку на обмін
	2. Підвищення кваліфікація працівників різних сфер. Працівники зможуть безкоштовно ділитись своїм професійним досвідом та в обмін отримувати нові знання.	2. Кваліфікація працівника (користувача) має бути достатньою для обміну навичками та знаннями.

	3. Самоосвіта зацікавлених людей, групи людей, яка прагне поділитись своїми напрацювання та хобі.	3. Група зацікавлених людей (користувачі) мають мати своє хобі та отриманий з досвідом багаж знань.
--	---	--

Далі буде проведено аналіз потенційних техніко-економічних переваг ідеї. На цьому етапі проводиться порівняльний аналіз показників існуючих рішень на ринку, який має визначити гірші, аналогічні та кращі значення для власної ідеї проекту. Даний аналіз зручно описати за допомогою побудованої табл. 8.2.

Таблиця 8.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Техніко- економічні характеристики ідеї	Системи конкурентів			W (слабка сторона)	N (нейтральн а сторона)	S (сильна сторона)
	Мій проект	SkillShar e	skillharb our			
Пошук галузей для обміну навичками та знаннями	+	+	+		+	
Зручна функціональна можливість обміну навичками	+	–	–			+
Можливість вибрати навичку за рейтингом	+	+	–			+

Техніко-економічні характеристики ідеї	Системи конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	Мій проєкт	SkillShare	skillharbour			
Можливість залишати коментарі	+	+	–			+
Обмін знаннями на безкоштовній основі за допомогою інтуїзму людей	+	–	–			+

8.2. Технологічний аудит ідеї проєкту

Даний підрозділ включає проведення аудиту технологій, які допомагають реалізувати проєкт. Технологічний аудит ідеї проєкту зручно представити за допомогою побудованої табл. 8.3.

Таблиця 8.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Веб-орієнтована система обміну навичками та знаннями	Python, Django, Angular	Наявні	Доступна
2		PHP, Laravel, React	Наявні	Доступна
3		JavaScript, Node.js, Vue.js	Наявні	Доступна
Обрана технологія реалізації ідеї проєкту: JavaScript, Node.js, Vue.js				

З побудованої табл. 8.3 можна зробити висновок, що всі технології наявні та доступні, а отже реалізація проєкту можлива. Отже, вибір технологій слід здійснити на принципі простоти та зручності їх процесі розробки. Вибір було здійснено на користь JavaScript, Node.js та Vue.js.

8.3. Аналіз ринкових можливостей запуску стартап-проєкту

Визначення ринкових можливостей, які використовують під час впровадження, та загроз, які перешкоджають впровадженню проєкту наведено у табл. 8.4-8.13.

Таблиця 8.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4-8 (якщо рахувати онлайн курси)
2	Загальний обсяг продаж, грн/ум.од	30 000+
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Присутня конкуренція, необхідність багатофункціональної пропозиції
5	Специфічні вимоги до стандартизації та сертифікації	Вимоги до безпеки даних користувачів
6	Середня норма рентабельності в галузі (або по ринку), %	15–25% (залежить від бізнес-моделі: підписки, реклами тощо)

З наведеної табл. 8.4 можна зробити висновок, що даний ринок є привабливим, якщо згодом перейти на модель підписки. Далі необхідно визначити характеристику потенційних груп клієнтів.

Таблиця 8.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Швидке вивчення чи удосконалення знань в певній галузі	Студенти. Люди працездатного віку, які планують змінити місце роботи, перейти на вищу посаду чи просто знайти роботу. Пенсіонери.	Свою першу роботу зазвичай шукає молоде покоління, яке орієнтується на високій заробітній платі, однак у них недостатній досвід. Користувачі від 25-40 років) більш впевнені в собі, тому вони залюбки скористаються можливістю змінити місце роботи на більш високооплачуване. Пенсіонери уже не шукають заробітку, однак теж можуть скористатись сервісом для отримання знань в цифровій галузі. Також вони можуть поділитись своїм життєвим досвідом.	- відсутність помилки в роботі сервісу; - зручність та швидкість під час користування; - робота системи на різних пристроях та браузерах. - хороша рейтингова складова; - зручність у пошуку навичок.

Далі можна провести аналіз ринково середовища, який складається з таблиць факторів загроз та факторів можливостей.

Таблиця 8.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Відсутність робочих місць	Складне економічне становище в країні.	Переорієнтація на більш потрібні та дефіцитні галузі. Залучення більшої кількості професіоналів з дефіцитними професіями.
2	Висока конкуренція на ринку	Клієнти можуть обирати інші системи через їх популярність та стабільне існування на ринку.	Введення акційних пропозицій та можливостей, які будуть доступні тільки для нових клієнтів.
3	Недостатні функціональні можливості системи	Користувач може помітити, що системи конкурентів надають більше функціональних можливостей та вибрати сервіс конкурентів.	Аналіз функціональних можливостей конкурентів. Розробка та впровадження кращих функціональних можливостей, ніж ті, що присутні в конкурентів.

Таблиця 8.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання інтересу до неформальної освіти	Інтерес до самоосвіти набирає обертів через важливість економії часу.	Збільшення серверів та підтримки великої кількості користувачів.
2	Отримання фінансування від донорів	Якщо система матиме шлях розвитку, то може зацікавити інвесторів.	Постійно працювати для розвитку системи, проваджувати нові креативні рішення.

Таблиця 8.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Чиста конкуренція	Подібних систем на ринку мало або вони перейши в магазин з продажу онлайн курсів.	Щоб утримати користувачів, необхідно постійно аналізувати ринок потреб та впроваджувати їх у розробленій системі.
Як мінімум національний рівень	Подібні системи широко популярні в нашій країні та за межами.	Компанія може аналізувати ринки інших країн та впроваджувати затребувані функціональні можливості.

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Міжгалузева	Дана характеристика дозволяє різко збільшити кількість користувачів.	Потребує від компанії постійного аналізу всіх галузей, що з'являються на ринку.
Товарно-видова	Веб-система мусить постійно змагатись із схожими платформами.	Компанії необхідно впроваджувати такий функціонал, якого немає ні в кого на ринку.
Нецінова	Якість сервісу, унікальний дизайн.	Компанія повинна розвивати партнерство з освітніми організаціями, вкладати кошти в створення унікального дизайну.
Немарочна	Бренд може бути не надто впізнаваним.	Компанія має добре реалізувати наявні функціональні можливості, впізнаваність бренду буде згодом.

Таблиця 8.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари заміники
Складові аналізу	SkillShare skillharbour	Оскільки існує велика кількість подібних систем, але вони працюють у форматі продаж курсів, то потенційних конкурентів не визначено.	Відсутні	Студенти, працюючий прошарок, пенсіонери	Відсутні
Висновки	Інтенсивність конкурентів середня, однак вони зацікавлені більше в прибутку.	Якщо не розвивати свою систему, то потенційні конкуренти можуть з часом її перевершити.	Дана система не потребує постачальників.	Клієнти є важливою складовою веб-системи, оскільки вона триматиметься саме завдяки ентузіазму людей.	Не несуть впливу на систему.

Визначення та обґрунтування факторів конкурентоспроможності наведено в табл. 8.10.

Таблиця 8.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Цінова складова	Безкоштовна основа хоч і з деякими платними функція працювати краще, ніж повністю платна система.
2	Унікальність пропозиції	Унікальні функціональні можливості зацікавлять велику кількість користувачів.
3	Зручність у використанні	Зручний дизайн заохочує з приємністю користуватись системою.
4	Кількість користувачів	Кількість користувачів відіграє вирішальну роль для правильного функціонування системи.

Оскільки фактори конкурентоспроможності визначені, то можна проводити аналіз сильних та слабких сторін стартап-проектів (табл. 8.11).

Таблиця 8.11 – Порівняльний аналіз сильних та слабких сторін «Веб-орієнтованої системи обміну навичками та знаннями»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з «Веб-орієнтованою системою обміну навичками та знаннями»						
			-3	-2	-1	0	+1	+2	+3
1	Цінова складова							+	
2	Унікальність пропозиції					+			
3	Зручність у використанні							+	
4	Кількість користувачів				+				

Далі можна скласти SWOT-аналіз – це фінальний етап ринково аналізу можливостей впровадження проєкту (табл. 8.12).

Таблиця 8.12 – SWOT- аналіз стартап-проєкту

Сильні сторони: - зручна функціональна можливість обміну навичками; - лаконічний інтерфейс та дизайн; - можливість вибрати навичку за рейтингом.	Слабкі сторони: - відсутній імідж компанії; - залежність від кількості користувачів;
Можливості: - додавати нові функціональні можливості, щоб вийти на ринок.	Загрози: - економічні проблеми в країні; - відсутність фінансових ресурсів для удосконалення системи.

Після цього на основі SWOT- аналізу розробляються альтернативи ринкового впровадження (табл. 8.13).

Таблиця 8.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розміщення рекламної кампанії в соціальних мережах	Нижче середньої	1-3 місяці
2	Проведення мастер-класів та вебінарів для демонстрації роботи платформи	Середня	2-4 місяці
3	Найм працівників популярних сфер діяльності для проведення пробних обмінів	Висока	6-8 місяців

З табл. 8.13 видно, перший варіант можна реалізувати за найменший строк і він є певною мірою ефективний, якщо правильно до цього підійти, однак в довготривалій перспективі найкраще підійдуть варіанти 2 та 3. Варіант 3 можна сказати, що найефективніший, однак він найбільш затратний та має великі строки реалізації. Напевно найбільш доцільним вибрати варіант під номером 2, він не гарантує високої ефективності, але строк реалізації швидший.

8.4. Розроблення ринкової стратегії проєкту

Таку стратегію зручно описувати за допомогою табл. 8.14-8.17.

Таблиця 8.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Студенти	Висока. Студенти найбільш мотивовані до оволодіння новими навичками.	Високий	Висока	Легка
2	Молодь	Висока. Молодь має великі плани на життя, тому готова швидко вивчати нові технології.	Високий	Середня	Легка
3	Люди від 35- до 45 років	Середня. Не всі готові кидати все та змінювати професію.	Середній	Низька	Середня

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
4	Пенсіонери	Низька. Незначна кількість пенсіонерів зацікавлена в продовженні навчання.	Низький	Низька	Складність висока
Які цільові групи обрано: Студенти, Молодь					

Далі варто сформулювати базову стратегію розвитку (табл. 8.15).

Таблиця 8.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Проведення мастер-класів та вебінарів для демонстрації роботи платформ и	Концентрований маркетинг	Можливість продемонструвати користувачам функціонал. Отримати конструктивну критику.	Стратегія спеціалізації. Орієнтуватись на найбільш затребувані функціональні можливості, які відзначають користувачі. Розширюватись зручність користування наявними можливостями.

Визначаю базову стратегію конкурентної поведінки (табл. 8.16).

Таблиця 8.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні, схожі системи уже існують.	Компанія може як шукати нових споживачів, так і забирати в конкурентів.	Компанія буде копіювати основні характеристики товару клієнта, але тільки на початку. Далі необхідно додавати свої креативні рішення, інакше клієнти підуть до інших конкурентів.	Стратегія наслідування лідеру

Визначена стратегія позиціонування представлена у табл. 8.17.

Таблиця 8.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1	Швидка та зручна система. Наявність достатньої кількості навичок для обміну. Працездатність системи на будь-якому девайсі.	Компанія аналізує функціональні можливості систем-конкурентів та покращує свої.	Доступність. Мінімальні витрати часу. Вибірання навички на основі рейтингу.	Зручність. Мобільність. Економія часу.

8.5. Розроблення маркетингової програми стартап-проєкту

Щоб сформувати маркетингову концепцію товару, необхідно підсумувати результати конкурентоспроможності товару. Підсумування результатів наведено в табл. 8.18.

Таблиця 8.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидке опанування необхідних технологій	Зменшення затраченого часу на навчання, оскільки навчання проводитиме професіонал своєї справи	Простий інтерфейс та безкоштовність системи
2	Зручність у використанні	Адаптивність системи та лаконічний дизайн	Мінімальні вимоги до пристрою, швидка робота навіть при повільному з'єднанні
3	Легкий вибір навички	Підбір навичок заснований на рейтингу	Платформа пропонує власний рейтинг

Розробка трирівневої маркетингової моделі представлено в табл. 8.19.

Таблиця 8.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Веб-орієнтована система обміну навичками та знаннями. До основних функціональних можливостей системи належить: створення власної навички, надсилання запитів для обміну навичками, пошук необхідних навичок на основі рейтингу, додавання потрібних навичок в список вподобаних, можливість переглядати коментарі та залишати.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Створення власних навичок.	М	Вр/Тх
	2. Пошук необхідних навичок за рейтингом чи категоріями.	М	Вр/Тх
	3. Процес обміну навичками з іншими учасниками.	М	Вр/Тх
	4. Зміна даних в особистому кабінеті.	М	Вр/Тх
	5. Додавання навичок у список вподобаних	М	Вр/Тх
	6. Залишення коментарів	М	Вр/Тх
	Якість: створена система протестована на працездатність основних функцій та відсутність критичних недоліків.		
	Пакування: наявна документація у пояснювальній записці, цифровий продукт, доступ через веб-систему.		
	Марка: «Skill Exchanger»		

Рівні товару	Сутність та складові
III. Товар із підкріпленням	До продажу: відсутні вимоги
	Після продажу: технічна підтримка та розширення функціоналу.
За рахунок чого потенційний товар буде захищено від копіювання: реєстрація права на інтелектуальну власність	

Визначення меж встановлення ціни наведено в тал. 8.20.

Таблиця 8.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Безкоштовний план (фінансування за рахунок реклами)	Ціна стандартної підписки на місяць (надання додаткових функціональних можливостей)	Середній дохід користувачів, оскільки велике число учасників можуть скласти студенти.	Безкоштовний доступ (платне відключення реклами)

Формування системи збуту наведено у табл. 8.21.

Таблиця 8.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Користувачі платять тільки за можливість відключити рекламу та додаткові функції, які будуть включені згодом	Постачальник має надавати технічну підтримку 24/7, зберігати безперебійну роботу системи, виправляти наявні недоліки в системі та оновлювати функціональні можливості	Канал нульового збуту або однорідне ви	За допомогою веб-системи або сторонніх сервісів

Розроблення концепції маркетингових комунікацій наведено у табл. 8.22.

Таблиця 8.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Студенти, які шукають доступні знання	Соціальні мережі	Сучасний доступ до знань, швидкодія системи	Сповіщати про функціональні можливості системи	Навчайся легко

№ п/ п	Специфіка поведінки цілових клієнтів	Канали комунікацій, якими користуютьс я цілові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомлення	Концепція рекламного о звернення
2	Молодь, яка шукає швидкий вхід у деяку галузь	Пошукові системи, соціальні мережі	Зручність використання з будь-якого пристрою	Сповіщати про преваги системи	Швидкий старт для початківці в
3	Зацікавлені працівники організацій , які хочуть розширити свої професійні можливості	Корпоративн і заходи, професійні розсилки	Система для підвищення кваліфікації	Просування важливості розвитку свого професіоналізм у	Вчись з будь-якого пристрою

Висновки до розділу 8

У процесі написання цього розділу дипломної дисертації було проведено маркетинговий аналіз стартап-проекту, метою якого є визначення принципової можливості ринкового впровадження. У ході роботи було визначено зміст ідеї, напрями застосування та проблеми, які вирішує розроблений продукт. Також було визначено попереднє коло конкурентів, що існують на ринку, проведено порівняльний аналіз, встановлено сильні, слабкі та нейтральні характеристики ідеї проекту. Проведено аудит технологій, визначено їх наявність та доступність,

проаналізовано ринкові можливості для запуску стартап-проєкту. При розробці стартап-проєкту було з'ясовано характеристику потенційних клієнтів, складено фактори загроз та можливостей. При завершенні даного розділу було розроблено ринкові стратегії та маркетингову програму стартап-проєкту.

Результат проведеного аналізу демонструє, що розроблена система та її стартап-проєкт має значний потенціал для виходу на ринок. Для виконання поставленої мети варто приділити значну увагу збільшенню функціональних можливостей.

ВИСНОВКИ

Результатом написання даної магістерської дисертація є розроблена веб-орієнтована система обміну навичками та знаннями. Дана система допоможе здійснити підбір навичок відповідно до рейтингу користувачів.

На початку роботи над дисертацією було визначено цільову аудиторію веб-орієнтованої системи, типи навичок та знань, які найбільш затребувані, та формат взаємодії між користувачами. Також було проаналізовано існуючі на сьогодні програмні рішення. Для аналізу було обрано дві найбільш схожі системи: SkillShare і skillharbour. Проведення аналізу орієнтувалось на огляді найбільш затребуваних функціональних можливостях, а також на перевагах та недоліках. Результатом даного аналізу стала постановка задачі на розробку власної системи.

Далі було сформовано функціональні та нефункціональні вимоги до майбутньої системи. Розроблені функціональні вимоги повинні враховувати те, що система має витримувати значне навантаження, мають забезпечувати швидкий відгук на будь-які дії клієнта, працювати при нестабільному Інтернет-з'єднанні та коректно відображатись на основних сучасний операційний системах чи веб-браузерах.

В магістерській дисертації було описано математичне забезпечення, яке відіграє важливу роль у процесі роботи системи, оскільки саме завдяки використанню математичних розрахунків визначається рейтинг користувача для роботи системи підбору навичок.

При подальшій роботі над дисертацією, було спроектовано базу даних, наведені сценарії використання системи, сплановано архітектуру, шляхом побудови UML-діаграм. Важливим етапом проектування веб-орієнтованої системи виступив підрозділ вибору технологій, у процесі роботи над яким вибрано такі технології, як: JavaScript, Node.js та Vue.js. У процесі роботи над магістерською дисертацією наведено особливості розробки серверної та клієнтської частин, в ході якої було спроектовано веб-орієнтовану систему обміну навичками та знаннями.

Після етапу розробки було проаналізовано спроектований інтерфейс

користувача, перевірено адаптивність та кросплатформність веб-системи.

Проведено тестування системи за допомогою димового методу, результатом роботи якого стала розроблена таблиця тестових випадків, яка допомогла здійснити перевірку системи та підтвердити працездатність всіх її функцій, а також стабільність та надійність.

На заключному етапі проєктування веб-орієнтованої системи було розроблено стартап-проєкт, результат аналізу якого продемонстрував, що розроблена система має значний потенціал для виходу на ринок при використанні правильного підходу до залучення ресурсів та клієнтів.

Таким чином запропонована веб-орієнтована система обміну навичками та знаннями відрізняється своєю інноваційністю, оскільки надає можливість користувачам миттєво переглядати, додавати навик та виконувати підбір на основі розробленого методу знаходження рейтингу користувача. Переваги запропонованої системи в порівнянні з існуючими рішеннями виявляються в її здатності швидкого підбору навичок та обміну контактними даними між користувачами.

Роботу розробленої платформи було перевірено, та встановлено, що запропонована система відповідає функціональним вимогам, а отже можна зробити висновок, що мету магістерської дисертації було досягнуто.

Подальший розвиток системи полягає в розширенні функціональних можливостей, а саме: інтеграції з різними популярними соціальними платформами, які забезпечать системі аудиторію та розвиток. Також важлива подальша робота над алгоритмом підбору навиків для користувачів, який включатиме використання штучного інтелекту. Такий підхід забезпечить максимально точний результат для користувача у процесі підбору навичок, оскільки дозволить максимально проаналізувати попередні взаємодії у ході обмінів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як зробити обмін знаннями ефективним [Електронний ресурс]. — Режим доступу: <https://mind.ua/openmind/20231798-spilkuemos-v-onlajni-yak-zrobiti-obmin-znannyami-efektivnim> — (14.10.2024).
2. Онлайн-спільнота для навчання [Електронний ресурс]. — Режим доступу: <https://www.skillshare.com/en/> — (14.10.2024).
3. Skill Harbour [Електронний ресурс]. — Режим доступу: <https://skillsharbour.com/> — (17.10.2024).
4. Що таке функціональні вимоги [Електронний ресурс]. — Режим доступу: <https://visuresolutions.com/uk/%D0%B1%D0%BB%D0%BE%D0%B3/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8/> — (18.10.2024).
5. Функціональні та нефункціональні вимоги [Електронний ресурс]. — Режим доступу: <https://www.guru99.com/uk/functional-vs-non-functional-requirements.html> — (18.10.2024).
6. Опорний конспект лекцій [Електронний ресурс]. — Режим доступу: http://dspace.wunu.edu.ua/bitstream/316497/41356/1/FCIT_kKN_bIPZ_dPBD_LEC.pdf — (23.10.2024).
7. Фізичне проектування структури бази даних [Електронний ресурс]. — Режим доступу: https://rdb.dp.ua/uk/chapter_04/ — (28.10.2024).
8. Як будувати UML-діаграми [Електронний ресурс]. — Режим доступу: <https://dou.ua/forums/topic/40575/> — (05.11.2024).
9. Діаграма послідовності [Електронний ресурс]. — Режим доступу: <https://www.maxzosim.com/sequence-diagrams/> — (07.11.2024).
10. Діаграма діяльності в UML: символ, компоненти та приклад [Електронний ресурс]. — Режим доступу: <https://www.guru99.com/uk/uml-activity-diagram.html> — (10.11.2024).

11. Які мови програмування треба знати для створення сайтів [Електронний ресурс]. — Режим доступу: <https://foxminded.ua/movy-prohramuvannia-dlia-stvorennia-saitiv/> — (17.11.2024).
12. PHP або Python: що вибрати і в чому між ними різниця [Електронний ресурс]. — Режим доступу: <https://goit.global/ua/articles/php-abo-python-shcho-vybraty-i-v-chomu-mizh-nymy-riznytsia/> — (19.11.2024).
13. Python - мова, що перевершує очікування [Електронний ресурс]. — Режим доступу: <https://nt.ua/blog/python-is-a-language-that-exceeds-expectations> — (20.11.2024).
14. Що таке JavaScript [Електронний ресурс]. — Режим доступу: <https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/> — (20.03.2024).
15. У чому переваги Node.js? [Електронний ресурс]. — Режим доступу: <https://artjoker.ua/blog/v-chem-preimushchestva-nodejs/> — (20.11.2024).
16. Детальний огляд та розбір Node.js [Електронний ресурс]. — Режим доступу: <https://wezom.com.ua/ua/blog/vse-cho-nuzhno-znat-o-nodejs> — (21.11.2024).
17. Фреймворки у мовах програмування [Електронний ресурс]. — Режим доступу: <https://cloud.itstep.org/blog/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them> — (21.11.2024).
18. Що таке React JS і для чого він потрібен? [Електронний ресурс]. — Режим доступу: <https://dan-it.com.ua/uk/blog/cho-takoe-react-js-i-dlja-chego-on-nuzhen/> — (21.11.2024).
19. Сучасні JavaScript фреймворки, їх переваги та недоліки [Електронний ресурс]. — Режим доступу: <https://it-blog.in.ua/suchasni-javascript-freymvorky-yikh-perevahy-ta-nedoliky/> — (21.11.2024).
20. Що таке Vue.js? [Електронний ресурс]. — Режим доступу: <https://lemon.school/blog/shcho-take-vue-js> — (21.11.2024).
21. Найкращий редактор коду для JavaScript [Електронний ресурс]. — Режим доступу: <https://dou.ua/forums/topic/47918/> — (21.11.2024).

22. Як писати REST API на Node.js правильно [Електронний ресурс]. — Режим доступу: <https://devzone.org.ua/post/iak-pysaty-rest-api-na-nodejs-pravylno> — (21.11.2024).

23. Перевірка адаптивності сайту за допомогою браузера [Електронний ресурс]. — Режим доступу: <https://webtune.com.ua/statti/internet-marketing/yak-pereviryty-adaptyvnist-za-dopomogoyu-brauzera/> — (21.11.2024).

24. Випробування диму - типи, процес, інструменти та інше [Електронний ресурс]. — Режим доступу: <https://www.zaptest.com/uk/%D0%B4%D0%B8%D0%BC%D0%BE%D0%B2%D1%96-%D0%B2%D0%B8%D0%BF%D1%80%D0%BE%D0%B1%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B3%D0%BB%D0%B8%D0%B1%D0%BE%D0%BA%D0%B5-%D0%B7%D0%B0%D0%BD%D1%83%D1%80%D0%B5%D0%BD> — (21.11.2024).