**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *Потебенька Єгора Дмитровича*
(ПІБ)

**академічної групи** *122-20-2*
(шифр)

**спеціальності** *122 Комп'ютерні науки*
(код і назва спеціальності)

**освітньої програми** *Комп'ютерні науки*
(назва освітньої програми)

**на тему:** *Розробка інтерактивної Q&A платформи для IT-спільноти з використанням Next.js*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| кваліфікаційної роботи | *доц. Спірінцев В.В.* | | | |
| **розділів:** | | | | |
| спеціальний | *доц. Спірінцев В.В.* | | | |
| економічний | *доц. Касьяненко Л.В.* | | | |
| | | | | |
| | | | | |
| **Рецензент** | | | | |
| **Нормоконтролер** | *доц. Гуліна І.Г.* | | | |

**Дніпро**
**2024**

**Міністерство освіти і науки України**
НТУ **«Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

# ЗАВДАННЯ
## на кваліфікаційну роботу

*бакалавра*
(назва освітньо-кваліфікаційного рівня)

**студента** *122-20-2* (група) *Потебенька Єгора Дмитровича* (прізвище та ініціали)

**тема кваліфікаційної роботи** *Розробка інтерактивної Q&A платформи для IT-спільноти з використанням Next.js*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

| Розділ | Зміст виконання | Термін виконання |
|---|---|---|
| ***Спеціальний*** | ***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.***<br>***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми*** | ***14.06.2024 р.*** |
| ***Економічний*** | ***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки*** | ***20.06.2024 р.*** |

Завдання видав _____ (підпис) *доц. Спірінцев В.В* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Потебенько Є.Д* (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

# РЕФЕРАТ

Пояснювальна записка: 136 с., 50 рис., 13 табл., 6 дод., 59 джерел.

Об’єкт розробки: інтерактивна Q&A платформа для IT-спільноти з використанням Next.js.

Мета кваліфікаційної роботи: забезпечення онлайн-простору для українських IT-спеціалістів, з метою обміну знаннями та спільного вирішення актуальних задач, що сприятиме об’єднанню фахівців і покращенню результативності галузі.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованої інформаційної системи, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи щодо створення додатку та розраховано час на його реалізацію.

Практичне значення полягає у створенні веб-орієнтованого середовища для обміну питаннями й відповідями, що забезпечує: спрощений процес пошуку рішень у IT-сфері; покращення розуміння складних концепцій; структурований репозиторій якісної інформації; засоби для розвитку професійних навичок; співпрацю між фахівцями та пошук спеціалістів.

Актуальність розробки інтерактивної Q&A платформи для українських IT-спеціалістів визначається усуненням дефіциту знань, зміцненням стійкості галузі та адаптивності спільноти, розвитком локальних талантів.

Список ключових слів: Q&A ПЛАТФОРМА, IT-СПІЛЬНОТА, МЕТА-ФРЕЙМВОРК, FULL-STACK РОЗРОБКА, TYPESCRIPT, DBAAS, SAAS, ІНТЕРАКТИВНІСТЬ, AGILE, ТЕСТУВАННЯ, UI, АДАПТИВНИЙ ДИЗАЙН, МАРШРУТИЗАЦІЯ, АВТЕНТИФІКАЦІЯ, ХМАРНІ ТЕХНОЛОГІЇ.

# ABSTRACT

Explanatory note: 136 pp., 50 fig., 13 tbl., 6 extra, 59 sources.

The object of development: an interactive Q&A platform for the IT community using Next.js.

The purpose of the diploma project: to provide an online space for Ukrainian IT specialists to exchange knowledge and collaboratively solve current challenges, fostering community among professionals and enhancing the industry's productivity.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic and clarifies the formulation of the problem.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the statement of the problem is formulated, the requirements for software implementation, technologies and software are indicated.

In the second section, the available solutions are analyzed, a platform for development is selected, the design and development of a web-oriented information system is carried out, the operation of the system, the algorithm and structure of its functioning, as well as the call and loading of the application are described, the input and output data are determined, the composition of the parameters of the technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating an application is calculated and the time for its creation is calculated.

The practical value lies in the creation of a web-oriented information system for question-and-answer exchange that provides: a simplified process for finding solutions in the IT field; improved understanding of complex concepts; a structured repository of quality information; tools for developing professional skills; collaboration between specialists and expert discovery.

The relevance of developing an interactive Q&A information system for Ukrainian IT specialists is determined by addressing knowledge gaps, strengthening industry resilience and community adaptability, and fostering local talent development.

List of keywords: Q&A PLATFORM, IT COMMUNITY, META-FRAMEWORK, FULL-STACK DEVELOPMENT, TYPESCRIPT, DBAAS, SAAS, INTERACTIVITY, AGILE, TESTING, UI, ADAPTIVE DESIGN, ROUTING, AUTHENTICATION, CLOUD TECHNOLOGY.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

| | |
|---|---|
| БД (DB) – | База даних (Database) |
| ПЗ – | Програмне забезпечення |
| ПК – | Персональний комп’ютер |
| СУБД (DBMS) – | Система управління базою даних (Database Management System) |
| ACID – | Atomicity, Consistency, Isolation, Durability |
| BFD – | Back-end For Database |
| BFF – | Back-end For Front-end |
| CSR – | Client-Side Rendering |
| DBaaS – | Database as a Service |
| E2E – | End-to-End |
| ERD – | Entity Relationship Diagram |
| ISR – | Incremental Static Regeneration |
| JSON – | JavaScript Object Notation |
| MQL – | MongoDB Query Language |
| Q&A – | Questions and Answers |
| SaaS – | Software as a Service |
| SQL – | Structured Query Language |
| SSG – | Static Site Generation |
| SSR – | Server-Side Rendering |
| TDD – | Test-Driven Development |
| UI – | User Interface |
| UML – | Unified Modelling Language |
| UX – | User Experience |
| WYSIWYG – | What You See Is What You Get |

# ВСТУП

Сфери комп'ютерних наук і розробки програмного забезпечення за своєю природою є заплутаними, тому потребують постійного навчання. Однак, у сучасному світі, де інформація доступна у безмежних обсягах як у друкованих, так і в цифрових форматах, виділення якісних знань набуває критичного рівня важливості. Через велику кількість дезінформації, люди шукають обізнаних осіб для консультації і вирішення певного діапазону завдань. Проте, нинішнім рішенням, як-от онлайн-форуми, бракує ефективності з позицій структурованості та якості комунікації. Унаслідок цього користувачі часто губляться в нескінченних дискусіях, що перешкоджає продуктивному обміну знань.

Кваліфікаційна робота пропонує веб-орієнтований застосунок ITHelps, спрямований на полегшення процесу вирішення проблем у IT-сфері. Основна ідея полягає у створенні відкритої платформи, призначеної для IT-фахівців, де можна обмінюватися питаннями і відповідями, а також мати довічний доступ до всіх наявних рішень. Q&A платформа буде зосереджена на українському ринку та спрямована на потреби однієї з рушійних галузей економіки країни. Наприклад, продукт може стати в нагоді системним архітекторам для вирішення питань, пов'язаних із шаблонами проектування. Розробники можуть її використовувати для усвідомленого вибору інструментів із огляду на досвід інших користувачів. Крім того, збагачений актуальними концепціями та найкращими практиками додаток стане цінним ресурсом протягом навчання.

Актуальність розробки інтерактивної Q&A платформи для українських IT-спеціалістів пояснюється наступними твердженнями:

а) усунення дефіциту знань. З огляду на тривалий конфлікт, така веб-система слугуватиме життєво важливим ресурсом для вивчення різноманітного досвіду й заповнення утворених прогалин у знаннях;

б) зміцнення стійкості та адаптивності. У часи невизначеності, пропонуючи надійне джерело знань і можливість для спільного вирішення

проблем, платформа дозволить об’єднати українську IT-спільноту, і отже продовжувати рух галузі вперед;

в) розвиток локальних талантів. Стартап-культура може вдосконалюватися завдяки підтримці інновацій та суспільному зростанню.

Об’єктом дослідження є інтерактивна Q&A платформа для IT-спільноти з використанням Next.js.

Метою кваліфікаційної роботи є розробка веб-орієнтованої системи для обміну знаннями, щоб вдосконалити ефективність вирішення проблем у сфері інформаційних технологій і об’єднати українських фахівців. Для її досягнення потрібно реалізувати наступні задачі:

– проаналізувати наявні на українському та світовому ринках Q&A платформи, визначити їхні переваги, недоліки та кращі практики в розробці;

– сформулювати основні вимоги до функціональних можливостей, інформаційної безпеки, складу й параметрів технічних засобів, сумісності з іншими системами та ПЗ;

– чітко визначити функціональне призначення;

– обрати стек технологій для розробки;

– спроектувати структуру веб-застосунку, алгоритми взаємодії з користувачами, організацію вхідних і вихідних даних;

– розробити веб-орієнтовану Q&A платформу ITHelps із забезпеченням потужного UX та можливості масштабування.

Практичне значення полягає у створенні веб-орієнтованого середовища для обміну питаннями й відповідями, що забезпечує: спрощений процес пошуку рішень у IT-сфері; покращення розуміння складних концепцій; структурований репозиторій якісної інформації; засоби для розвитку професійних навичок; співпрацю між фахівцями та пошук спеціалістів.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

## 1.1. Загальні відомості з предметної галузі

### 1.1.1. Характеристика області застосування

Тематику розробленого продукту можна віднести до предметних областей дванадцятої галузі. У межах поточного контексту, для більш точної класифікації слід розглянути наступні задіяні підсекції:

– людино-комп’ютерна взаємодія. Ця область зосереджена на вивченні та проектуванні інтерфейсів для взаємодії людей із комп’ютерами. Бакалаврський проєкт передбачає розробку інтуїтивно зрозумілого інтерфейсу, що забезпечує безперебійну взаємодію між користувачами та платформою, дозволяючи їм ефективно задавати та відповідати на запитання;

– розробка веб-додатків. Застосунок ITHelps насамперед є інтернет-платформою, яка створена з використанням сучасного фреймворку Next.js й інших веб-орієнтованих інструментів;

– система управління знаннями. Ця IT-структура призначена для збору, зберігання, спільного використання організаційних знань та інформаційних ресурсів. Вона безпосередньо пов’язана з ідеєю кваліфікаційної роботи, оскільки розроблена платформа функціонує як сховище знань, де цільова авдиторія ділиться своїми досвідом і рішеннями, сприяючи створенню єдиної IT-спільноти;

– гейміфікація. Це означає використання ігрових практик і механізмів у неігровому контексті для залучення, заохочення й мотивації користувачів до розв’язання певного діапазону проблем. Підхід є суттєвим у контексті ITHelps, бо користувачі мають змогу розвиватися і змагатися на платформі, завдяки комплексним системам репутації, рейтингів, а також нагород;

– взаємодія людини з ШІ. Ця сфера вивчає та проектує методи ефективної комунікації і співпраці між людьми та системами штучного інтелекту. У кваліфікаційній роботі ця модель використовується для реалізації функції автоматичної генерації відповідей, що допоможе користувачам організовувати якіснішу комунікацію.

Отже, бакалаврський проєкт – комплексна веб-платформа, що охоплює та інтегрує численні дисципліни дванадцятої галузі.

### 1.1.2. Аналіз наявних альтернатив

З метою розуміння сучасного стану Q&A платформ для IT-спільнот на світовому та українському ринках, у цьому розділі проведено аналіз найбільш популярних представників, виокремлено їхні переваги та недоліки, оцінено рівень вирішення ними поставлених завдань, проаналізовано технічні протиріччя та визначено певні прогалини у знаннях.

Одним із усесвітньо відомих рішень є веб-орієнтована система Stack Overflow (рис. 1.1) [1], де розробники обмінюються досвідом через запитання та відповіді виняткова англійською мовою (рис. 1.2).

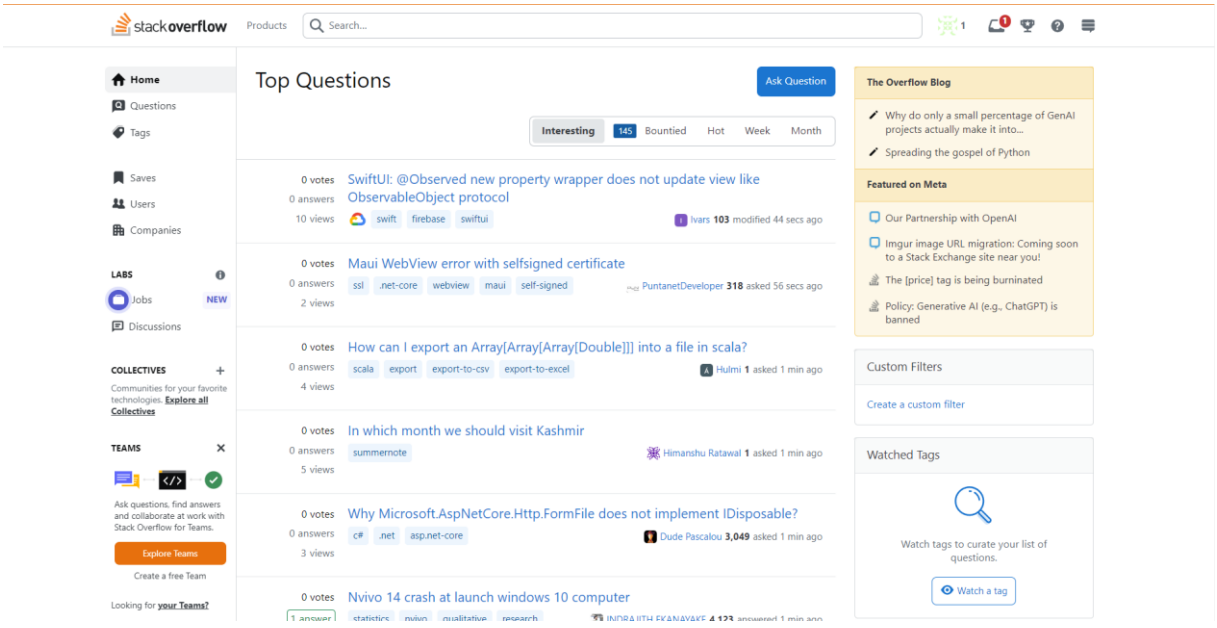


Рис. 1.1. Головна сторінка Stack Overflow


Рис. 1.2. Приклад прийнятного політикою обговорення

За інформацією від CEO компанії [2], наразі платформа є одним із найпопулярніших ресурсів серед розробників. Цьому сприяє наявність близько 50 мільйонів запитань і відповідей, що охоплюють широкий спектр тем у сфері інженерії програмного забезпечення. Веб-застосунок щомісячно обслуговує близько 100 мільйонів відвідувачів, що робить його одним із найпопулярніших у світі [3]. Окрім цього, за результатами міжнародних опитувань «2023 Developer Survey» [4] і «2022 Developer Survey» (рис. 1.3) [5], у яких взяло участь 160,000 розробників, було визначено, що Stack Overflow – другий за популярністю онлайн-ресурс серед тих, хто навчається.

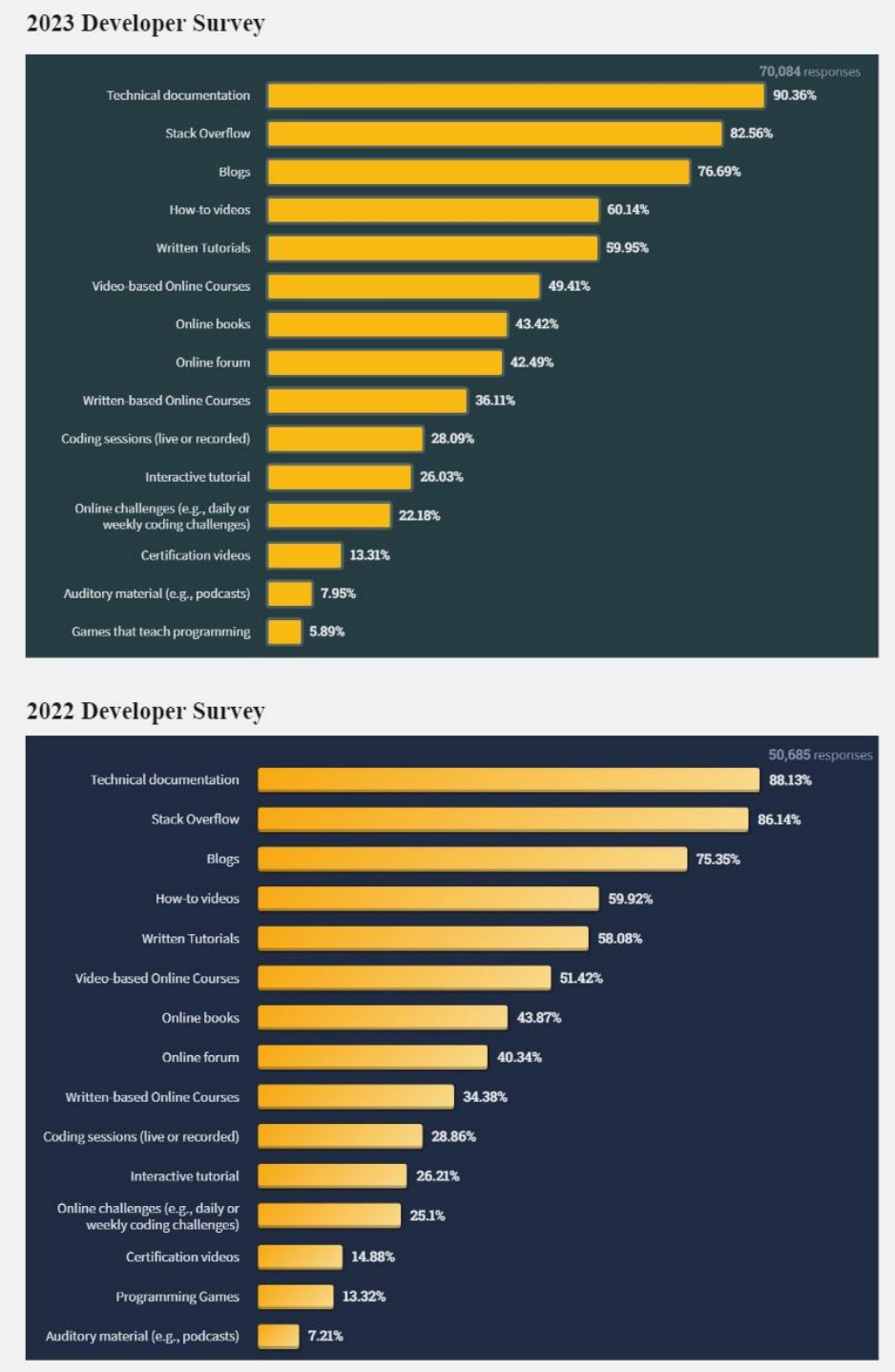


Рис. 1.3. Результати «2023 Developer Survey» і «2022 Developer Survey»

Хоча Stack Overflow є важливою платформою для програмістів, насправді вона має низку недоліків пов’язаних із модерацією:

– недружня атмосфера. Проблема особливо відчутна для новачків, які часто стикаються з негативною реакцією. Таке неприємне середовище

перешкоджає навчанню та знеохочує тисячі нових людей користуватись платформою;

– неефективна модерація вмісту. Користувачі з високим рейтингом іноді видають ідеї з відповідей інших учасників за свої, що ставить під сумнів здатність платформи підтримувати чесне та етичне середовище;

– жорсткі вимоги до релевантності контенту. Оскільки платформа створена винятково для розробників, загальні питання будуть швидко закриті, через відсутність опису конкретної інженерної проблеми.

З метою виявлення технічних вразливостей, необхідно проаналізувати організацію внутрішньої системи та використані технології. З інтерв'ю керівниці відділу розробки [6], платформа обробляє понад 6000 запитів на секунду, 2 мільярди переглядів сторінок на місяць, а перший контент вдається показати за 12 мілісекунд. Попри вражаючі показники, архітектура Stack Overflow [7] існує протягом 15 років і являє собою гігантську монолітну програму, що працює локально. Одна програма в інформаційній службі Інтернету, що працює на дев'яти веб-серверах і одному SQL-сервері (з додаванням одного резервного), керує 200 сайтами. До того ж, використовується дворівневий кеш:

– перший рівень розташований на сервері SQL з великим об'ємом оперативної пам'яті (1.5 ТБ). Завдяки цьому, 30% звернень до БД обробляються з оперативної пам'яті;

– другий рівень складається з двох серверів Redis (основний і репліка).

Крім того, архітектура поєднує 3 сервери обробки тегів і 3 сервери Elastic Search, які обробляють 34 мільйони пошукових запитів щодня (рис. 1.4).

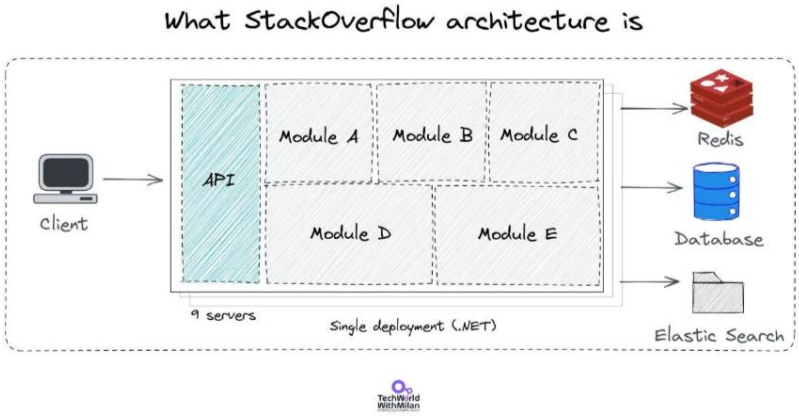


Рис. 1.4. Архітектура Stack Overflow

Варто зазначити, що для розробки використовується наступний стек технологій:

– C# + ASP.NET MVC – фреймворк для побудови веб-застосунків за шаблоном проектування «Model-View-Controller»;

– Dapper ORM – продукт об’єктно-реляційного відображення для платформи .NET;

– StackExchange Redis – високопродуктивний клієнт .NET для роботи з Redis;

– MiniProfiler – інструмент профілювання ASP.NET MVC додатків і запитів до баз даних;

– Jil – засіб десеріалізації JSON, тобто його перетворення в об’єкт;

– StackExchange.Exceptional – внутрішній обробник SQL Server помилок;

– Sigil – бібліотека генерування CIL, проміжної мови, розробленої компанією Microsoft для платформи .NET;

– NetGain – швидкісний WebSocket сервер;

– Opserver – моніторингова система, що призначена для аналізу широкого спектру показників, пов’язаних з роботою інфраструктури компанії;

– Bosun – система моніторингу серверної частини.

З вищенаведеної інформації можна підсумувати, що системна архітектура Stack Overflow продемонструвала вражаючу продуктивність, масштабованість і надійність протягом 15 років. Проте, результати аналізу Google Lighthouse (рис. 1.5), інструменту для оцінки якості веб-сторінок, виявили моменти для покращення.

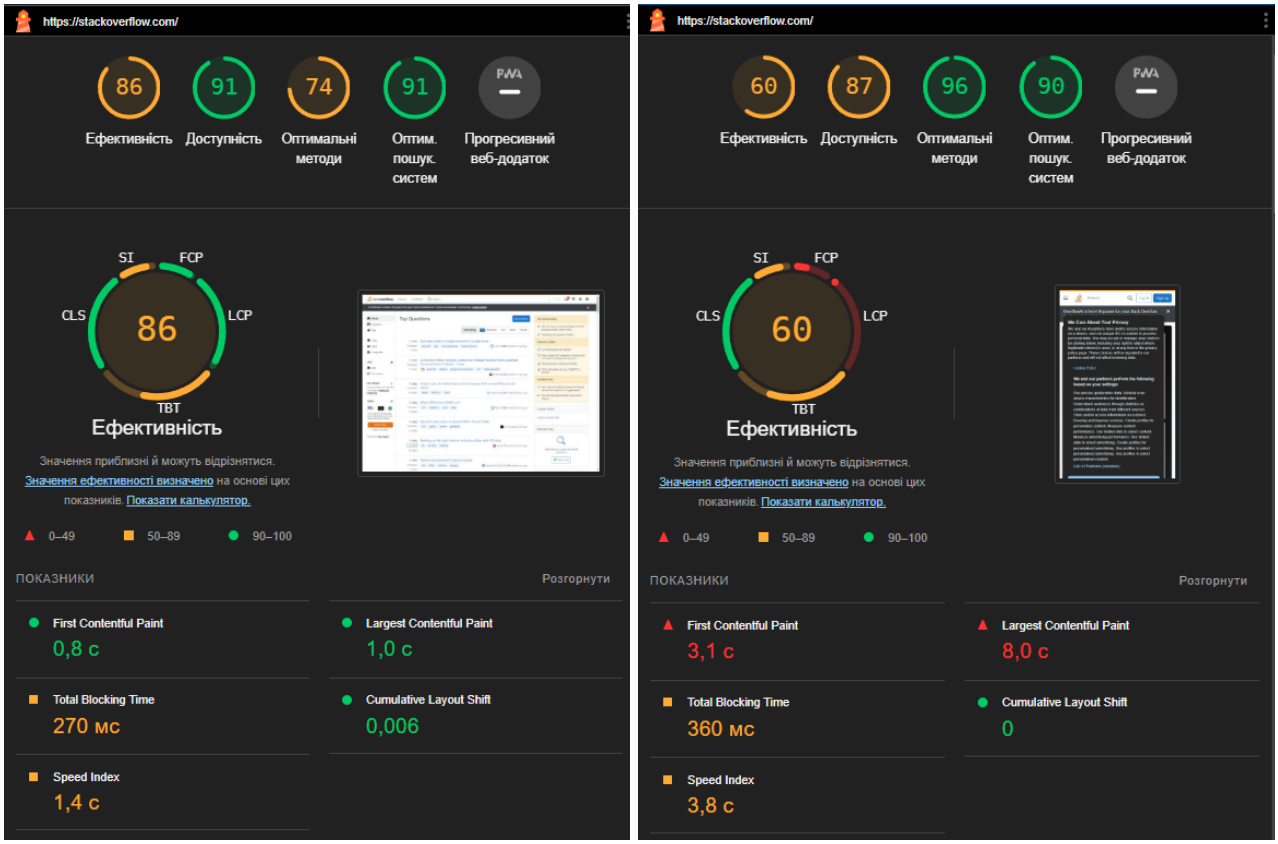


Рис. 1.5. Оцінка ефективності роботи веб-застосунку Stack Overflow інструментом Google Lighthouse на ПК і мобільних пристроях

Дослідження були проведені в браузері Google Chrome, використовуючи режим інкогніто для забезпечення точніших результатів. З огляду на аналіз, можна зробити певні висновки:

а) показник використання оптимальних методів на комп'ютерних пристроях знаходиться у середньому діапазоні. Справа в тому, що використання традиційних фреймворків, як-от ASP.NET MVC, може бути не настільки

ефективним, порівняно з сучасними інструментами, через необхідність споживання більшої кількості ресурсів;

б) середні значення індексу швидкості 1,4с на комп'ютерних пристроях і 3,8с на мобільних вказують на те, що під час завантаження сторінок обробляються великі об'єми коду. Традиційне архітектурне рішення відрізняється від підходу, який використовується у багатьох сучасних веб-додатках, побудованих за односторінковим принципом. Stack Overflow завантажує нові HTML, CSS і JavaScript-файли під час обробки кожної нової сторінки, тоді зазначений підхід дозволяє підвантажувати потрібні дані порційно, без необхідності перезавантаження веб-сторінки;

в) загальний час блокування вказує на те, що сторінка не реагує на дії користувача протягом 270мс на ПК і 360мс на мобільних девайсах, зумовлений недостатньо дієвим кешуванням статичних ресурсів. Цей недолік можна було б виправити використанням різноманітних стратегій рендерингу. Наприклад, сучасні фреймворки пропонують вибір між SSG, SSR, CSR, ISR [8].

Серед основних функціональних можливостей, окрім обміну питаннями та відповідями, учасники можуть їх редагувати, голосувати, покращувати свою репутацію та отримувати винагороди, переглядати популярні тематики та профілі користувачів, шукати інформацію за категоріями та фільтрувати її.

Отже, ретельно проаналізувавши Q&A платформу Stack Overflow, можна підсумувати, що незважаючи на всесвітню популярність і стабільну довготривалу роботу, існує потенціал для покращення рівня вирішених завдань.

На українському ринку технологічна галузь продовжує залишатися однією з ключових для української економіки, згідно з результатами всеукраїнського дослідження «IT Research Ukraine 2023», проведеного наприкінці 2023 року спільнотою IT-компаній «Львівський IT Кластер» (рис. 1.6) [9].

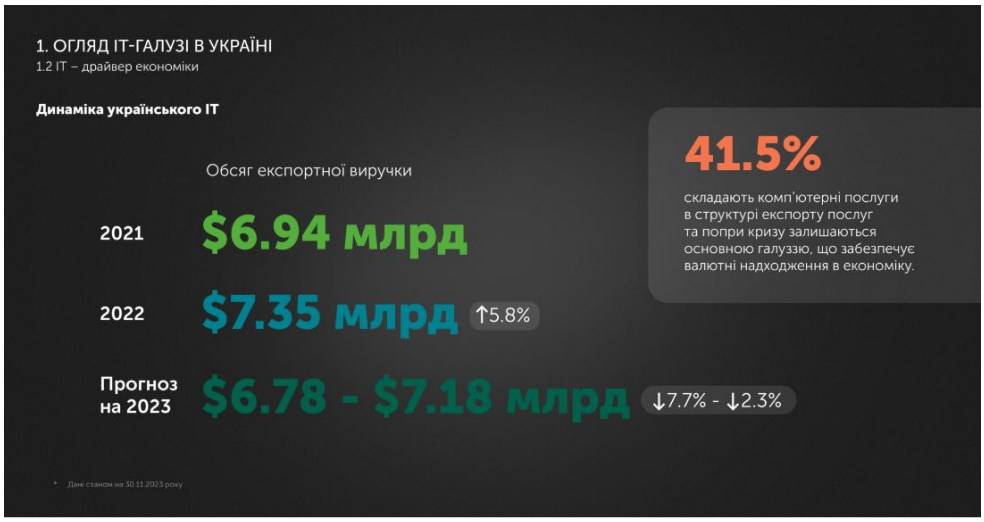


Рис. 1.6. Актуальна динаміка українського IT-ринку

Однак, попри стрімку популярність сектору інформаційних технологій і жваву спільноту, не існує спеціалізованих Q&A платформ, як Stack Overflow. Тому, пропонується розглянути онлайн-спільноту DOU (рис. 1.7) [10], що об’єднує понад 500 тисяч користувачів IT-індустрії.

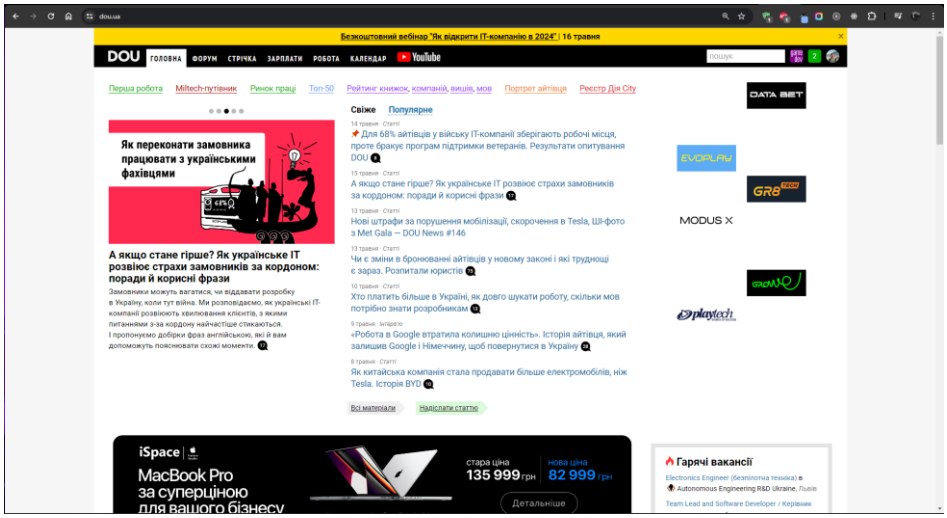


Рис. 1.7. Головна сторінка DOU

Мережа пропонує різноманітні ресурси, але особливу увагу необхідно надати розділу «Форуми». Цей сегмент сайту найбільше нагадує формат спілкування на Stack Overflow, однак потрібно чітко окреслити відмінності, наведені в таблиці 1.1.

Таблиця 1.1

**Порівняння Q&A форматів на DOU і Stack Overflow**

| Аспект | Платформи | |
|---|---|---|
| | DOU | Stack Overflow |
| Фокус і зміст | Охоплює широкий спектр тем у IT-індустрії | Технічні питання, пов'язані з програмуванням і розробкою ПЗ |
| Організація та структура | Вільні дискусії, що можуть перерости в довші неформальні розмови | Суворий Q&A формат. Питання організовані за категоріями, а відповіді показують рішення конкретних проблем |
| Мова та спільнота | Українська й англійська. Різноманітна спільнота українських IT-фахівців. | Англійська. Міжнародна спільнота розробників і технічних експертів. |

З огляду на порівняльну таблицю, DOU-форуми сприяють більш неформальному та комплексному обговоренню теми інформаційних технологій, тоді як платформа Stack Overflow орієнтована на швидке вирішення конкретних технічних проблем. Через різний погляд на процес спілкування і переважну кількість місцевих користувачів, на платформі DOU комунікація більш приємна, а вимоги до модерації не такі суворі.

Отже, проаналізувавши найпопулярніші продукти на глобальному і локальному ринках, виявилося, що в досліджуваній галузі існує значний потенціал для розвитку. Зважаючи на монопольне становище Stack Overflow на світовому ринку та відсутність прямих аналогів в Україні, виникає гостра

потреба в поглибленні знань і розв’язанні технічних суперечностей. У час, коли об’єднання ІТ-фахівців набуває особливої актуальності, а сфера комп’ютерних наук є одним із рушіїв економіки країни, реалізація мети кваліфікаційної роботи може суттєво покращити загальну ситуацію. Більш того, з огляду на успішний досвід Stack Overflow, подібна система має всі шанси стати однією з найбільш поширених в Україні. У результаті, розробка платформи ґрунтуватиметься на концепції суспільного єднання, подібної до DOU-товариства, з урахуванням досвіду архітектурних рішень Stack Overflow.

## 1.2. Призначення розробки та галузь застосування

У бакалаврській роботі розглядається інтерактивна Q&A платформа ITHelps у ролі об’єкта впровадження розроблюваної веб-орієнтованої інформаційної системи, з метою обміну досвідом, покращення результативності галузі інформаційних технологій та згуртування українських ІТ-спеціалістів.

Причини виникнення необхідності створення мережі ITHelps на українському ринку:

а) нестача централізованого сховища знань. За умов відсутності єдиної системи, цінна інформація та досвід залишаються несистематизованими та розсіяними. Відокремлена платформа слугувала б комплексним репозиторієм, що значно полегшило б ІТ-фахівцям процес пошуку якісних рішень;

б) потреба ефективного обміну знаннями. Працівники ІТ-сфери часто стикаються з подібними проблемами або мають запитання, на які вже отримали відповіді інші. Завдяки такій автоматизації, продукт вирішить проблему дублювання роботи, що значно підвищить суцільну продуктивність;

в) необхідність у якісній підготовці нових кадрів. Платформа ITHelps має всі шанси стати одним із найпопулярніших ресурсів для навчання та наставництва серед початківців і студентів України, повторивши успіх Stack Overflow на світовому ринку;

г) мовна і культурна відповідність. З огляду на наявну негативну атмосферу в мережі Stack Overflow, розробка платформи, адаптованої до української мови, культури та практик, може сприяти кращому спілкуванню та задоволенню потреб локальних спеціалістів.

Завдяки універсальності Q&A платформи ITHelps, орієнтованої на сферу інформаційних технологій, можна навести декілька сценаріїв застосування:

а) розробка ПЗ. ITHelps – цінний ресурс для розробників, які шукають рішення проблем, пов’язаних із програмуванням, найкращими практиками та галузевими знаннями в різноманітних бібліотеках чи фреймворках;

б) технічна підтримка. Платформу можна використовувати як інструмент для усунення несправностей, де учасники можуть звернутися за допомогою в подоланні технічних перешкод, пов’язаних із програмним чи апаратним забезпеченням;

в) онлайн-навчання й освіта. Інформаційну систему можна використовувати як допоміжний інструмент для навчальних платформ, курсів або навчальних закладів, надаючи студентам доступ до величезної бази досвіду ІТ-фахівців;

г) пошук кадрових ресурсів. Платформа згуртовує ІТ-спеціалістів, стимулюючи співпрацю та відкриваючи двері до нових горизонтів професійного розвитку. Крім цього, завдяки рейтинговій статистиці та активності учасників на платформі, компаніям буде значно легше знаходити кваліфікованих фахівців.

Зважаючи на вищенаведені приклади, можна зробити висновок, що Q&A платформа ITHelps може бути цінним інструментом для компаній, які залежать від ІТ-процесів, а також для навчальних закладів і платформ.

Основна термінологія містить загальні терміни зі сфер інформаційних технологій і веб-розробки. Ключові слова: Q&A платформа, ІТ-спільнота, мета-фреймворк, full-stack розробка, TypeScript, DbaaS, SaaS, інтерактивність, Agile, тестування, UI, адаптивний дизайн, маршрутизація, автентифікація, хмарні технології.

Терміни, використані у бакалаврській роботі, і потрібні задля розуміння елементної та технічної структури системи:

– спільнота – одна з найголовніших сторінок ITHelps, де можна переглянути всіх учасників системи;

– колекція – частина платформи, що відповідає за набір збережених запитань;

– тег – ключове слово, яке допомагає категоризувати та організовувати питання на платформі ITHelps;

– фільтрація – гнучкий інструмент, що дозволяє персоналізувати свій досвід на платформі та відобразити зміст згідно зі встановленими параметрами;

– пагінація – метод поділу великих обсягів інформації, наприклад питань або відповідей, на логічні та зручні для сприйняття сторінки, позитивно впливаючи на перегляд контенту та оптимізацію;

– система відзнак – механізм, створений для нагородження користувачів за їхній внесок на платформі;

– глобальна пошукова система – потужний інструмент, що надає можливість легко та швидко знаходити необхідну інформацію в межах усієї мережі;

– компонент – незалежний блок коду, який реалізує частину інтерфейсу користувача в React-застосунках і зазвичай використовується багаторазово;

– стратегія рендерингу – план, що визначає, як і коли буде відображатися вміст на веб-сторінці;

– рендеринг на стороні клієнта (CSR) – підхід до відображення вмісту веб-сайту за допомогою JavaScript, замість попередньо сформованого HTML на сервері;

– серверний рендеринг (SSR) – створює HTML-контент на сервері для передавання клієнтській стороні;

– статична генерація сайту (SSG) – передбачає попередню підготовку усіх сторінок як статичних HTML-файлів під час процесу збірки;

– інкрементальна статична регенерація (ISR) – поєднує SSR і SSG підходи, тобто дозволяє попередньо візуалізувати статичний контент під час збірки, періодично оновлюючи його актуальними даними;

– маршрутизація – фундаментальна концепція веб-розробки, що описує логіку переміщення користувачів між різними сторінками веб-сайту;

– мета-фреймворк – фреймворк вищого рівня, що розширяє можливості та структуру системи, на якій він базується;

– односторінковий веб-застосунок – веб-програма, що спочатку завантажує лише один HTML-документ, а потім динамічно оновлює потрібні частини вмісту, коли це потрібно, без потреби реактивації сторінки;

– нативність – природна сумісність або відповідність технології до певного середовища чи платформи;

– модуль – самостійний компонент програмного коду, який може бути імпортований та експортований для повторного використання. У такий спосіб утворюється модульна організація проєктів;

– кластер – група взаємопов’язаних серверів, які працюють разом як єдина система;

– транзакція – логічна одиниця роботи в БД, яка складається з однієї або декількох операцій, що виконуються як єдине ціле, гарантуючи цілісність і узгодженість даних;

– атомарність – передбачає повне виконання транзакції або навпаки;

– узгодженість – переведення БД із одного валідного стану в інший;

– ізольованість – передбачає повну незалежність виконання транзакцій;

– довговічність – забезпечення постійного зберігання змін після успішного завершення транзакції.

## 1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма спеціальності 122 «Комп’ютерні науки»;

– навчальний план і графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» №469-с від 23.05.2024;

– завдання на кваліфікаційну роботу на тему «Розробка інтерактивної Q&A платформи для IT-спільноти з використанням Next.js».

## 1.4. Постановка завдання

Протягом кваліфікаційної роботи розглядається створення веб-орієнтованої інтерактивної Q&A системи ITHelps.

Головна мета: забезпечити онлайн-простір для українських IT-спеціалістів, де можна обмінюватися знаннями та спільно вирішувати актуальні задачі, що сприятиме об’єднанню фахівців і покращенню результативності галузі.

Призначення: усунення недоліків традиційних дискусійних форумів, як-от організованості інформації та контролю якості спілкування, шляхом реалізації централізованої системи для обміну запитаннями й відповідями.

Технічно-економічна сутність завдання: забезпечити безперервність технологічного прогресу методом впровадження локалізованої платформи ITHelps у масове використання, де українські фахівці зможуть підтримувати колективні знання та досвід на актуальному рівні. Для цього надати необхідний функціонал та інтуїтивно зрозумілий інтерфейс з метою забезпечення злагодженого процесу співпраці.

Розробка такої платформи є доцільним і перспективним рішенням, адже демонструючи у такий спосіб експертизу українських фахівців, можна залучити

інвестиції, створити робочі місця й можливості для наукової діяльності, стимулюючи зростання технологічного та економічного потенціалу країни.

Веб-додаток ITHelps насправді виходить за межі індивідуального користування та може бути застосований різними технологічними об’єктами управління, підрозділами та підприємствами. Нижче перелічені структури, якими може застосовуватись розроблена система:

– компанії, що спеціалізуються на розробці. У подібних фірмах програмісти часто потребують ресурсів для швидкого пошуку ефективних рішень і вивчення технічних аспектів;

– служби технічної підтримки. Завдяки швидкому пошуку інформації, розроблена система може стати ідеальним інструментом для фахівців, яким необхідно оперативно знаходити рішення поширених програмних і апаратних проблем;

– укладачі технічної документації. Технічні автори можуть використовувати веб-застосунок для дослідження концепцій програмування, синтаксису та найкращих практик, що сприятиме створенню точних і сучасних документів;

– науково-дослідні та академічні установи. Дослідники і вчені можуть використовувати ITHelps як джерело для аналізу актуальних підходів до вирішення проблем і викликів, з якими стикаються IT-спеціалісти;

– організації з підбору персоналу. Профілі учасників можуть аналізуватися рекрутерами для оцінки технічних навичок, досвіду та активності потенційних кандидатів.

ITHelps – складна інформаційна система, що містить наступні структурні об’єкти:

– облікові записи користувачів. Стан цього пункту характеризує інформація про кількість зареєстрованих учасників, їхню активність і репутацію, а також збережені запитання;

– питання. Цей об’єкт характеризується загальною кількістю, їхньою популярністю (число переглядів, вподобань, відповідей), розподілом голосів («за» і «проти»), статусом («найновіші», «рекомендовані», «часті», «без відповіді»), обсягом нових питань на день/тиждень/місяць, доданими тегами, актуальністю;

– відповіді. Стан поточного структурного елемента можна описати загальною кількістю, обсягом нових відповідей на день/тиждень/місяць, розподілом голосів («за» і «проти»), якістю (згідно з числом високих балів), датою додавання;

– теги. Їхній стан можна визначити кількістю унікальних тегів, популярністю (за кількістю питань) і загальним охопленням (відсоток запитань із відповідними тегами);

– пошук і навігація. Цей об’єкт характеризується обсягом і шаблоном пошукових запитів, числом елементів пошуку на сайті, маршрутами навігації та шляхом клієнта;

– залучення спільноти. Стан поточного структурного елементу описується участю користувачів у обговореннях питань та їхньою взаємодією з активністю інших учасників;

– технологічна інфраструктура. Визначається такими показниками: ефективність, доступність, оптимальні методи, оптимізований пошук систем;

– якість і релевантність вмісту. Характеризується відсотком запитань із застарілою інформацією, співвідношенням кількості голосів «за» і «проти» серед питань і відповідей, висвітленням тенденцій ІТ-сфері.

Вихідна інформація в межах веб-орієнтованої платформи ITHelps – це структуровані та опрацьовані дані, що надаються користувачам у зручному для сприйняття вигляді. Вона формується на основі вхідних даних, наданих учасниками спільноти (питання, відповіді, результати голосування, теги тощо) і являє собою базу якісних рішень наявних проблем, впорядковану за допомогою тегів, рейтингів і прийнятих відповідей. Головне призначення – із забезпеченням

якості, структурувати та класифікувати основний вміст, задля комфортного сприйняття та зручної навігації на платформі.

Для ефективного збору вхідної інформації, окрім розробки спеціальних форм для введення даних, також треба використати багатофункціональний редактор тексту для оформлення повноцінних питань і відповідей. З цієї причини, основною вимогою є створення зручного та зрозумілого інтерфейсу користувача. З метою полегшення процесу заповнення даних і зменшення кількості помилок, необхідно логічно назвати пункти у формах, надати широкий спектр підказок і реалізувати ненав’язливу валідацію. Попри запропоновану систему мінімізації похибок та неактуальності введених даних, їх контроль та коригування є суттєвими. Тож, усі сформульовані питання, відповіді та пов'язані з ними додаткові деталі можна буде редагувати та видаляти. У такий спосіб, учасникам буде зручно заповнювати відомості, а програмі їх організувати.

У системі чітко визначено розподіл функцій між персоналом і технічними засобами. Це гарантує, що учасник має доступ до необхідних йому інструментів та інформації для ефективного виконання задач на платформі.

Для ролі незареєстрованого користувача потрібно впровадити створення облікового запису шляхом надання основних відомостей: унікального імені користувача, адреси електронної пошти та пароля, а також, за бажанням, ім’я та прізвища.

Для зареєстрованих учасників треба розробити описаний нижче функціонал:

– доступ до власного профілю з інформацією про персональні дані, репутацію, дату приєднання, нагороди, кількість поставлених запитань і наданих відповідей;

– доступ до власного облікового запису через веб-інтерфейс платформи керування користувачами Clerk [11];

– редагування персональної інформації в обліковому записі безпосередньо на платформі або через інтерфейс Clerk;

– видалення облікового запису за допомогою інструменту Clerk;

– авторизація за допомогою електронної пошти або імені користувача, та пароля;

– інтеграція з сервісами GitHub та Google для спрощеного входу в систему;

– відновлення забутих облікових даних та зв'язок із технічною підтримкою;

– детальний огляд усіх наявних запитань і відповідей;

– створення, модифікація та вилучення запитань;

– публікація, коригування та видалення відповідей;

– можливість формування відповідей за допомогою систем штучного інтелекту;

– присвоєння до трьох тегів для кожного опублікованого запитання;

– додавання власних чи сторонніх запитань до персональної колекції;

– перегляд списку збережених питань;

– фільтрація записів у колекції за датою додавання, кількістю отриманих голосів, переглядів чи відповідей;

– надання позитивної або негативної оцінки будь-яким запитанням або відповідям;

– відображення створених запитань за конкретним тегом.

Для обох вищезазначених категорій користувачів необхідно забезпечити наступні опції:

– зміна візуального оформлення (світлий, темний, системний варіанти інтерфейсу);

– огляд списку усіх наявних запитань із доступом до їхньої статистики;

– пошук запитань із застосуванням фільтрів для демонстрації найновіших, рекомендованих, часто відвідуваних або без відповідей питань;

– огляд списку зареєстрованих користувачів;

– пошук учасників спільноти за іменем користувача або іменем і прізвищем;

– фільтрація наявних користувачів за рівнем активності та датою створення облікового запису;

– перегляд профілів зареєстрованих осіб із інформацією про персональні дані, репутацію, дату приєднання, нагороди, кількість поставлених запитань і наданих відповідей;

– доступ до переліку всіх наявних на платформі тегів;

– перегляд найпопулярніших тегів;

– пошук запитань за певним тегом;

– пошук тегів за назвою;

– фільтрація тегів в алфавітному порядку, за популярністю чи датою створення;

– відображення найактуальніших запитань;

– глобальний пошук на платформі за запитаннями, відповідями, користувачами чи тегами;

– кросплатформеність для забезпечення сумісності з різними типами пристроїв (мобільні телефони, планшети, персональні комп'ютери).

Для комплексної візуалізації розподілу функціоналу між ролями, нижче наведено діаграму прецедентів (англ. Use case diagram) (рис. 1.8), створену за допомогою уніфікованої мови моделювання і сервісу PlantUML Server [12].

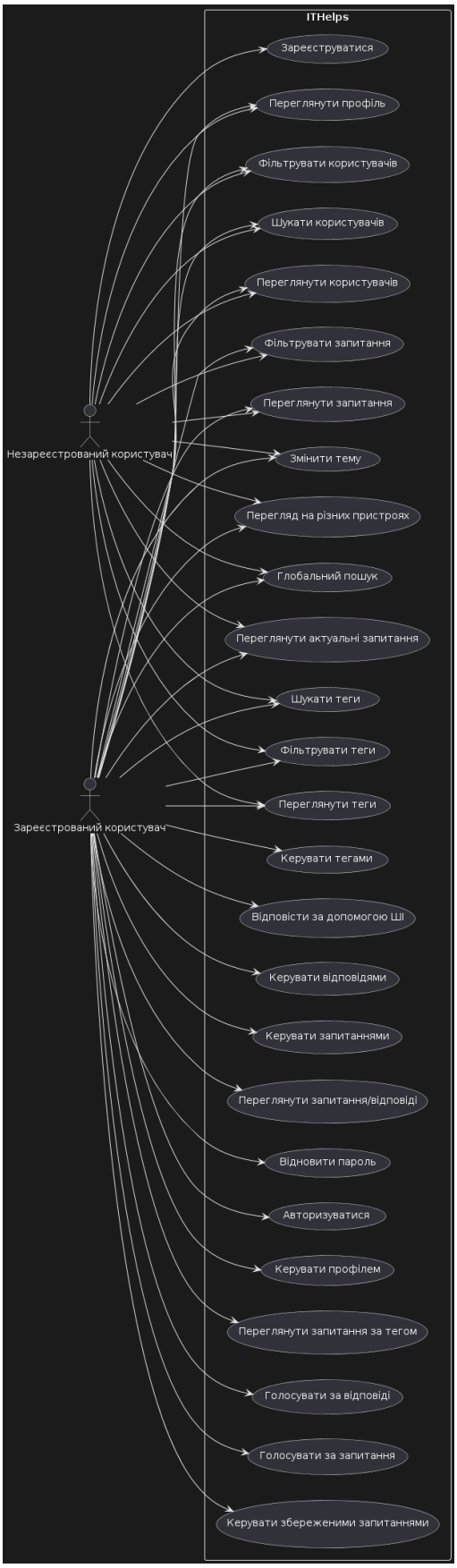


Рис. 1.8. Діаграма прецедентів ITHelps

## 1.5. Вимоги до програми або програмного виробу

### 1.5.1. Вимоги до функціональних характеристик

Уведення даних на веб-платформі ITHelps має здійснюватися користувачами у стандартному порядку через розроблений веб-інтерфейс із використанням клавіатури. З метою оптимізації процесу, треба застосувати різноманітні форми з полями для текстових відповідей відкритого типу, а також у відповідних частинах застосунку надати доступні опції. Наприклад, під час авторизації чи створення акаунту, учасники можуть скористатися сторонніми сервісами, як-от Google і GitHub, що більш зручно. Аналогічно, у разі пошуку користувачів слід забезпечити можливість вибору із списку вже зареєстрованих учасників для покращення UX.

Для створення та форматування запитань і відповідей необхідно застосувати багатофункціональний редактор типу WYSIWYG. Таке рішення повинне підтримувати розширені списки, автоматичне створення посилань, керування списками, вставки посилань, зображень, спеціальних символів, попереднього перегляду, закладок, пошуку та заміни тексту, візуального відображення блокових елементів, вставки зразків коду, переходу в повноекранний режим, вставки поточного часу та медіа-контенту, а також створення таблиць. Панель інструментів повинна містити кнопки для відміни/повтору дій, вставки зразків коду, виділення тексту напівжирним/курсивом, зміни кольору шрифту, вирівнювання абзаців, створення маркованих і нумерованих списків. Для відображення дат у різних частинах веб-застосунку потрібно використати наступні формати:

– створення облікового запису: [Місяць повністю прописом] [рік] (наприклад, «Травень 2024»);

– публікація запитання: запитав/ла [кількість] [одиниця часу в однині/множині] (наприклад, «запитав/ла 3 дні тому»);

– публікація відповіді: відповів/ла [кількість] [одиниця часу в однині/множині] (наприклад, «відповів/ла 3 дні тому»).

Структуру даних слід розділити на два рівні: основні та допоміжні елементи. Основні відповідатимуть за фундаментальні аспекти системи та її призначення, наприклад, відповіді, користувачі та теги. Допоміжні стосуватимуться другорядних процесів, що підтримують основні функції, наприклад, колекція збережених питань, система нагород, репутація.

Залежно від типу користувача, мають відрізнятися не лише доступні функції, але й представлення даних. Наприклад, тоді як незареєстровані користувачі матимуть доступ тільки до попереднього огляду запитань, авторизовані учасники зможуть детально розглянути кожне питання, надати відповідь і проголосувати.

Організація даних на платформі потребує відповідних можливостей фільтрації та пошуку. Наприклад, базовими параметрами фільтрації запитань можуть бути: дата публікації, популярність, кількість відповідей, теги тощо. Пошук повинен відбуватися за ключовими словами у змісті, а також за тегами. Крім того, необхідно надати можливість знаходити учасників за іменем користувача чи повним ім’ям.

Зважаючи на введені дані мають виконуватися різноманітні операції та візуалізації, щоб допомогти користувачам швидко знаходити потрібну інформацію та отримувати уявлення про загальні тенденції на платформі. Наприклад, згідно з опублікованими учасниками обговореннями потрібно показувати й оновлювати списки найпопулярніших тем і тегів.

Подання даних користувачам здійснюється в кількох основних формах: списки запитань/відповідей/тегів/користувачів, окремі сторінки для детального перегляду цих сутностей, особисті профілі з персональною інформацією та статистикою. Для забезпечення ефективної роботи рекомендується використати нереляційну модель бази даних MongoDB та розгорнути її у хмарному середовищі MongoDB Atlas [13].

### 1.5.2. Вимоги до інформаційної безпеки

Насамперед потрібно зазначити, що інформаційна безпека – це стан захищеності системи від загроз, який визначається рівнем потенційної шкоди через використання некоректної інформації, негативного інформаційного впливу, протиправного застосування технологій або порушення цілісності, конфіденційності та доступності даних.

Головною вимогою щодо організації безпеки даних користувачів є обов'язкове створення облікових записів та захищений вхід у систему. Для реалізації цього функціоналу рекомендовано використати інструмент керування обліковими записами Clerk. Ця платформа забезпечує інструментарієм для захисту процесів реєстрації, автентифікації та управління акаунтами.

Передусім, під час створення облікового запису або автентифікації відбувається перевірка введених даних на відповідність вимогам безпеки. Зі сторони клієнта, обов'язкові поля, як-от ім'я користувача, пошта та пароль перевіряються на довжину та існування необхідних і заборонених символів. Серверна частина, для запобігання створенню дубльованих профілів, перевіряє базу на наявність однакових облікових даних. Більш того, Clerk перевіряє введений пароль на збіги зі скомпрометованими списками, а також блокує широковідомі та вразливі комбінації, наприклад «password», «123456», «qwerty» тощо. До того ж, як це показано на рис. 1.9, є можливість авторизації через надійних провайдерів ідентифікації, як-от Google і GitHub, що додатково підсилює безпеку завдяки делегуванню цього процесу.

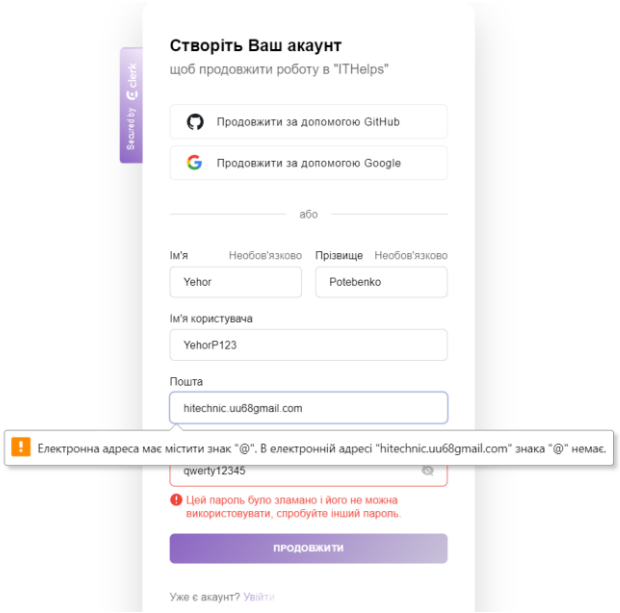


Рис. 1.9. Валідація даних протягом реєстрації

Під час передачі даних, між клієнтом і сервером використовуються захищені канали через протокол HTTPS. Варто зазначити, що підтримується додатковий рівень безпеки завдяки двоетапній верифікації. Після реєстрації на вказану електронну адресу відправляється одноразовий код підтвердження, який необхідно ввести для завершення активації облікового запису (рис. 1.10).

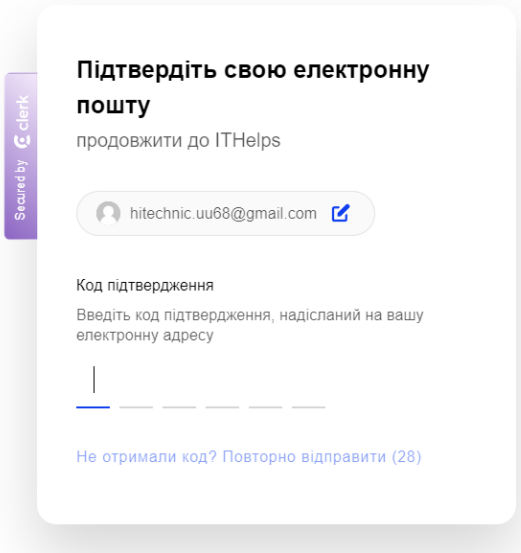


Рис. 1.10. Запит на підтвердження електронної пошти

Важливо підкреслити, що для синхронізації актуальної інформації облікових записів між створеною БД та вбудованою базою даних користувачів у Clerk, потрібно налаштувати постійну та безпечну взаємодію. З цієї причини, потрібно застосувати вебхуки, створені для реалізації захищеного способу передачі даних між двома системами в реальному часі, через HTTPS. Оскільки такий спосіб вимагає HTTPS-з'єднання, необхідно розгорнути веб-застосунок на надійному хостингу. Для цього рекомендується використати платформу Vercel [14], спеціалізовану на безпечному розгортанні веб-додатків, особливо розроблених на Next.js.

Особливу увагу також потрібно надати змінним оточення та стратегії їх збереження. Оскільки розроблений веб-додаток ITHelps взаємодіє зі сторонніми сервісами, вразливу інформацію, як-от API-ключі та паролі необхідно зберігати у файлі .env. Замість публікації цієї інформації безпосередньо у віддаленому репозиторії, Vercel пропонує окреме безпечне сховище, що гарантує захист даних та безперебійну роботу додатку для споживачів.

### 1.5.3. Вимоги до складу та параметрів технічних засобів

У межах бакалаврської роботи передбачено, що інформаційна система ITHelps матиме безсерверну монолітну архітектуру та буде розгорнутою на хмарній платформі Vercel.

Найнижчі характеристики для запуску продукту на мобільних пристроях:

– 4-ядерний процесор із частотою від 1.5 ГГц;

– 3 ГБ оперативної пам'яті;

– Android 8.0 або iOS 12;

– сучасний браузер із підтримкою ES6 (ECMAScript 2015) (наприклад, Google Chrome, Firefox, Microsoft Edge, Safari, Opera);

– 720p (1280x720 пікселів) роздільної здатності екрану;

– 3G-мережа.

Мінімальні вимоги для запуску веб-застосунку на персональних комп’ютерах:

– 2-ядерний процесор із частотою від 2.0 ГГц;
– 4 ГБ оперативної пам'яті;
– Windows 10 або macOS 10.13 High Sierra або сучасні дистрибутиви Linux (наприклад, Ubuntu 20.04);
– сучасний браузер із підтримкою ES6 (ECMAScript 2015) (наприклад, Google Chrome, Firefox, Microsoft Edge, Safari, Opera);
– інтегрований графічний процесор;
– 1366x768 пікселів роздільної здатності екрану;
– стабільне широкосмугове з’єднання з інтернетом.

Наведені вище технічні ознаки є мінімальними для роботи програмного продукту. Це означає, що у разі перевищення зазначених вимог, користування буде зручнішим. Отже, широкий спектр наявних електронних девайсів, що мають доступ до браузера та можливість введення даних, підходять для стабільної роботи з розробленою Q&A платформою.

### 1.5.4. Вимоги до інформаційної та програмної сумісності

Оскільки розроблена система використовуватиме сторонні сервіси для реалізації певних функціональних можливостей, насамперед треба розглянути їхні умови користування. Раніше зазначалось, що застосунок буде розгорнутий на платформі Vercel, база даних регулюватиметься інфраструктурою MongoDB Atlas, а керування обліковими записами відбуватиметься з допомогою сервісу Clerk. Для інтеграції перелічених інструментів достатньо скористатися запропонованими безкоштовними планами.

Можливості й обмеження стартового пакету Vercel [15]:

– імпорт будь-якого Git-репозиторія;
– автоматичні процеси неперервних інтеграції та доставлення;
– безсерверні обчислення, де ресурси виділяються за потреби;

– статистика трафіку й ефективності.

Ознаки та ліміти використаного плану MongoDB Atlas [16]:

– обсяг пам’яті від 512 МБ до 5 ГБ;

– спільна оперативна пам’ять між усіма користувачами.

Базовий пакет сервісу Clerk [17] дозволяє мати до 10 000 активних користувачів на місяць.

Після ретельного аналізу ринку схожих програмних реалізацій у підрозділі 1.1.2, окрім впровадження вищенаведених сервісів, вирішено використати наступний технологічний стек:

– Next.js – full-stack фреймворк для реалізації веб-орієнтованих додатків із підтримкою різноманітних стратегій рендерингу;

– TypeScript – мова програмування високого рівня, що розширяє JavaScript статичною типізацією з додатковими анотаціями типів;

– Tailwind CSS – CSS-фреймворк для швидкого написання стилів;

– shadcn/ui – колекція UI-компонентів;

– Zod – бібліотека для опису структури та перевірки даних;

– Vitest – сучасний фреймворк для тестування JavaScript застосунків;

– React Testing Library – бібліотека, що містить утиліти для тестування React-компонентів;

– ESLint – інструмент для аналізу коду JavaScript, що використовується для виявлення помилок і забезпечення дотримання узгодженого стилю програмування;

– Prettier – інструмент форматування коду;

– Git – система контролю версій для відстеження змін у файловій системі.

# РОЗДІЛ 2
# ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1. Функціональне призначення системи

Функціональне призначення розробленого програмного продукту можна визначити за наступними критеріями:

– організація окремого онлайн-простору, де представники IT-індустрії можуть обмінюватися питаннями й відповідями;

– створення великої бази даних проблем і рішень у галузі інформаційних технологій із можливістю зручного пошуку та фільтрації інформації;

– забезпечення інтерактивної системи голосування, що сприяє оцінюванню якості, точності, корисності контенту та впливає на його рейтинг;

– впровадження гнучких системи репутації та механізму нагород, які стимулюють користувачів до активної участі в розвитку спільноти;

– створення системи тегів, що полегшує класифікацію та пошук відповідної інформації;

– надання зручних інструментів для самостійного редагування та видалення вмісту, а також вдосконалення запитів за допомогою ШІ;

– забезпечення кожного користувача обліковим записом, що зберігає інформацію про активність на платформі та може бути орієнтиром для рекрутерів під час пошуку потенційного кандидата;

– впровадження багатофункціонального текстового редактора для чіткого та зручного оформлення вмісту.

Експлуатаційне призначення полягає в:

– прискоренні вирішення задач, що дозволяє користувачам суттєво зекономити час, необхідний для виявлення проблем і їх усунення;

– забезпеченні комплексного навчального ресурсу, де можна ознайомлюватися з актуальними темами, вивчати різноманітні методики та вчитися вирішувати проблеми;

– ефективному вирішенні, актуальної на локальному ринку праці, проблеми професійної ізоляції. Завдяки врахуванню специфіки місцевого середовища, розроблена платформа може стати місцем об’єднання та психологічного комфорту для української ІТ-спільноти в умовах національних викликів;

– підвищенні кар’єрної привабливості. Репутація на платформі ITHelps може стати потужним доповненням до резюме, а обліковий запис – більш помітним для потенційних роботодавців і рекрутерів;

– створенні безкоштовного ресурсу для розвитку як спеціаліста. Факт публікації інформації у вільному доступі робить платформу корисним інструментом для навчання і вдосконалення навичок цільової авдиторії;

– швидкому доступу до актуальних якісних даних, завдяки активній участі спільноти, системі голосування, а також потужним інструментам пошуку та фільтрації.

## 2.2. Опис застосованих математичних методів

Під час створення інтерактивної Q&A платформи ITHelps використовувались арифметичні операції та перетворення для роботи з датою та часом. Розроблена утиліта «getRelativeTime» обчислює проміжок часу, що минув із певної дати, та виражає цю різницю в зручному для читання форматі з використанням різних одиниць.

Формула (2.1) розраховує різницю в часі:

$$\Delta t = t_2 - t_1, \qquad (2.1)$$

де $t_1$ – початковий час,

$t_2$ – кінцевий час,

$\Delta t$ – різниця між $t_2$ і $t_1$.

Крім того, була використана універсальна формула (2.2) для конвертації мілісекунд у потрібну часову одиницю:

$$T = ms / (f_1 \times f_2 \times ... \times f_n \times 1000), \quad (2.2)$$

де T – кількість часу в бажаній одиниці,

ms – кількість мілісекунд,

$f_1, f_2, ..., f_n$ - фактори конвертації (множники) для кожної проміжної одиниці часу,

1000 – константа, яка завжди присутня, оскільки 1 секунда = 1000 мілісекунд.

Завдяки зазначеним математичним методам у додатку фіксується час, як це показано на рис. 2.1, коли користувач звернувся із запитанням або, як це показано на рис. 2.2, надав відповідь:

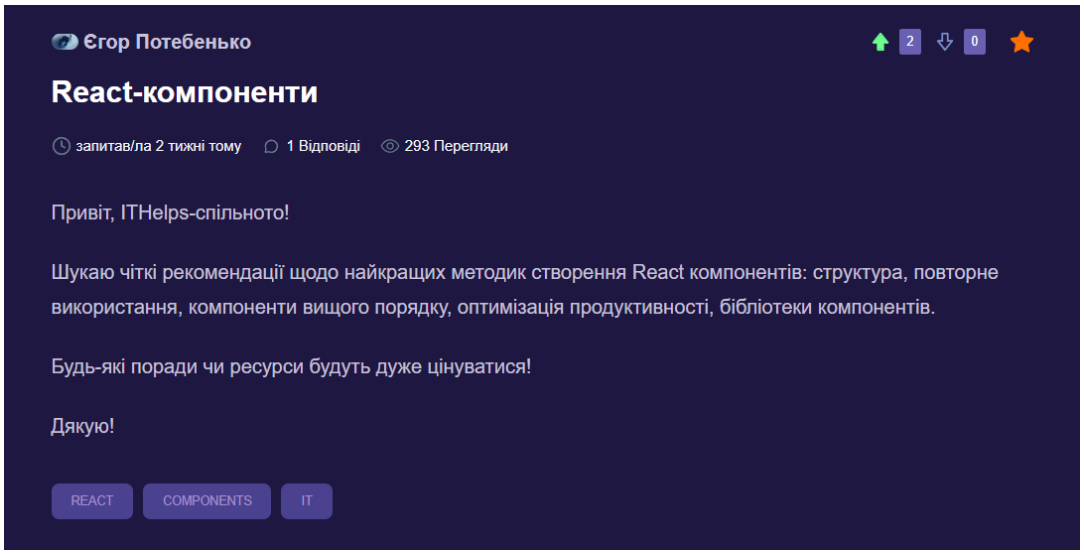


Рис. 2.1. Запит одного із користувачів із вказаною часовою позначкою «2 тижні тому»

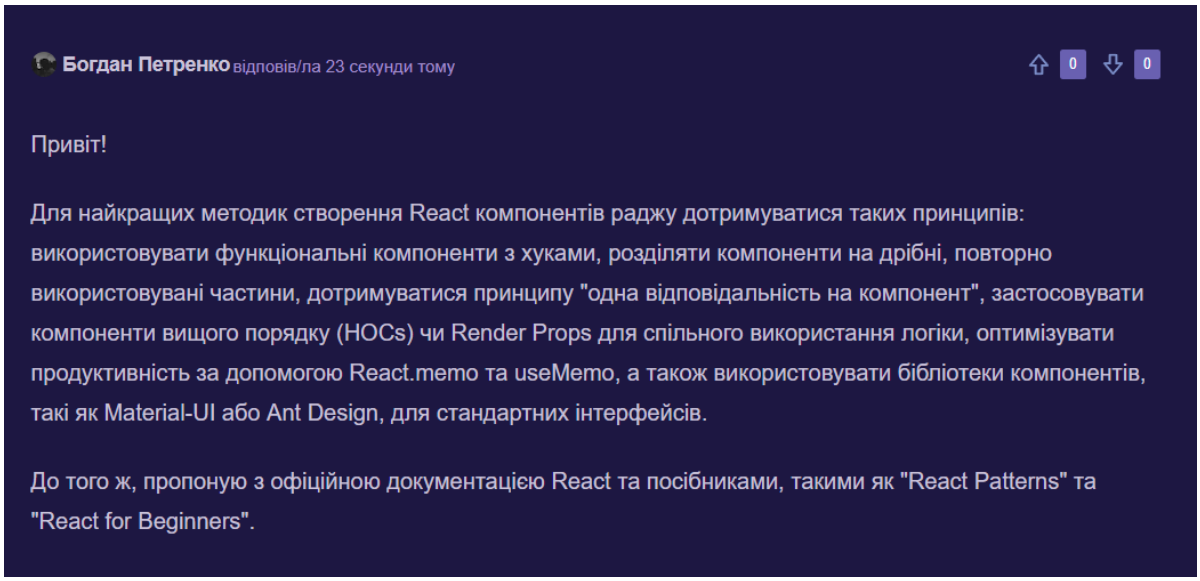


Рис. 2.2. Приклад відповіді учасника із вказаною часовою позначкою «7 хвилин тому»

Лістинг функції «getRelativeTime», реалізованої на TypeScript, із необхідними для правильного функціонування програмними одиницями:

**getRelativeTime** // утиліта, що визначає часовий інтервал від заданої дати до поточного моменту та представляє цю тривалість, застосовуючи різноманітні часові одиниці

```
// Гнучкий тип для відповідності ключів до значень
type Dynamic<Key extends PropertyKey, Value> = Record<Key, Value>;
// Усі можливі одиниці часу
type TimeUnitName =
    | 'second' | 'minute' | 'hour' | 'day'
    | 'week' | 'month' | 'year';
// Структура для зберігання інтервалу часу та його форм в українській мові
type TimeInterval = {
    value: number;
    forms: string[]; // [однина, 2-4, 5-0]
};
// Константа визначає об'єкт із динамічними типами, який пов'язує
// назви одиниць часу з їх значеннями у мілісекундах
const TIME_UNITS: Dynamic<TimeUnitName, number> = {
    second: 1000,
    minute: 60 * 1000,
    hour: 60 * 60 * 1000,
    day: 24 * 60 * 60 * 1000,
    week: 7 * 24 * 60 * 60 * 1000,
    month: 30 * 24 * 60 * 60 * 1000,
    year: 365 * 24 * 60 * 60 * 1000,
};
// Інтервали часу з їхніми формами однини чи множини
const TIME_INTERVALS: TimeInterval[] = [
```

```
    { value: TIME_UNITS.year, forms: ['рік', 'роки', 'років'] },
    { value: TIME_UNITS.month, forms: ['місяць', 'місяці', 'місяців'] },
    { value: TIME_UNITS.week, forms: ['тиждень', 'тижні', 'тижнів'] },
    { value: TIME_UNITS.day, forms: ['день', 'дні', 'днів'] },
    { value: TIME_UNITS.hour, forms: ['годину', 'години', 'годин'] },
    { value: TIME_UNITS.minute, forms: ['хвилину', 'хвилини', 'хвилин'] },
    { value: TIME_UNITS.second, forms: ['секунду', 'секунди', 'секунд'] },
];
// Визначає правильну форму слова залежно від значення
const defineTimeUnitCountableForm = (
    timeUnit: number,
    [one, few, many]: string[]
): string =>
    timeUnit % 10 === 1 && timeUnit % 100 !== 11
        ? one
        : [2, 3, 4].includes(timeUnit % 10) &&
          ![12, 13, 14].includes(timeUnit % 100)
        ? few
        : many;
// Повертає час у відносному форматі
const getRelativeTime = (createdAt: Date): string => {
    const currentTimeMoment = new Date();
    const elapsedTime = currentTimeMoment.getTime() - createdAt.getTime();
    // Пошук найбільшої відповідної одиниці часу
    for (const { value, forms } of TIME_INTERVALS) {
        const timeUnit = Math.floor(elapsedTime / value);
        if (timeUnit >= 1)
            return `${timeUnit} ${defineTimeUnitCountableForm(timeUnit, forms)} тому`;
    }
    return 'щойно';
};
```

## 2.3. Опис використаних технологій та мов програмування

### 2.3.1. Вибір методології керування проєктом

Agile і Waterfall – дві найрозповсюдженіші методології управління проєктами [18]. Головна відмінність полягає в тому, що Waterfall – це послідовна система, яка вимагає завершення кожного етапу проєкту перед переходом до наступного, а Agile – гнучка, лояльна до змін модель, яка дозволяє працювати одночасно над різними етапами створення продукту. Обидві користуються популярністю в розробці ПЗ, але необхідно визначити доречнішу, відповідно до специфіки веб-орієнтованого додатку ITHelps. З цією метою наведено

порівняльну характеристику в таблиці 2.1, за основними показниками: графік, гнучкість, зворотній зв’язок.

Таблиця 2.1

**Порівняння Waterfall і Agile**

| Аспект | Методології | |
|---|---|---|
| | Waterfall | Agile |
| Графік | Фіксований, дата початку та кінця проєкту визначаються заздалегідь | Гнучкий у процесі розробки, адаптується до змін на будь-якому етапі |
| Гнучкість | Перш ніж розпочинати наступну фазу, потрібно завершити попередню | Закладена безпосередньо в методологію, цінуються короткі робочі цикли (спринти) |
| Зворотній зв’язок | Після затвердження вимог не передбачається | Важливо задовольнити потреби клієнта якомога раніше, що забезпечується їхніми регулярними відгуками та залученням у процес |

З таблиці видно, що Waterfall ідеально підходить для проєктів, де кінцевий результат визначено на початку, і отже можна заздалегідь розпланувати всі фази розробки ПЗ. Чіткі правила та вимоги необхідно розробити на початку, а результати кожного етапу та суворий процес розробки повинні гарантувати їх виконання. Державні організації, міністерства, аерокосмічна промисловість –

галузі, які часто використовують аналізовану методологію, оскільки жорстке дотримання вимог є запорукою безпеки.

З іншого боку, Agile – гнучкіша й динамічніша форма управління проєктами. Розробка може тривати довго, а структура та технології – суттєво змінюватися з часом. Така методологія підходить проєктам, де на початку чітко сформувати специфікації неможливо, і отже передбачається регулярне внесення змін і взаємозв'язок із зацікавленими сторонами. Тому цикл створення ПЗ не охоплює весь додаток одночасно, а розподіляється на короткі ітерації або спринти, зазвичай тривалістю два тижні. Під час кожного такого етапу розробляються невеликі частини функціоналу, які називаються інкрементами. Із презентацією кожного інкременту, програмний продукт розвивається поступово, а вимоги стають чіткішими.

Отже, у контексті створення програмного продукту для кваліфікаційної роботи, перевагу надано методології Agile, оскільки передбачається гнучке ставлення до коректувань, регулярні перегляди бачення фінального результату постачальника, і взаємодія з керівниками спеціального та економічного розділів.

### 2.3.2. Аналіз обраної мови програмування

У галузі веб-розробки, високорівнева мова програмування JavaScript посідає беззаперечне провідне місце, забезпечуючи функціонування клієнтської частини в 98,7%, наявних у цифровому світі, веб-сайтів [19]. Завдяки своїй гнучкості й універсальності вона стає фаворитом серед розробників, проте не варто ігнорувати специфічні аспекти. Оскільки мова особлива динамічною природою, тобто сутності можуть зберігати значення різних типів під час виконання програмного коду, цей факт може призвести до непередбачуваних результатів, помилок, а також ускладнення процесів розробки та тестування. Як це видно з рис. 2.3, змінна «x» може містити число, рядок або масив у різні моменти часу без декларування типів даних.

```
let x = 5; // наразі x - число
x = 'Hello'; // тепер x - рядок
x = [1, 2, 3]; // тепер x - масив
```

Рис. 2.3. Приклад поведінки динамічної типізації

До того ж, аналізована мова програмування є слабко типізованою, що дозволяє виконувати операції між різними типами даних без необхідності їхнього явного перетворення (рис. 2.4).

```
let num = 5; // num - число
let str = '10'; // str - рядок

let result = num * str; // JavaScript здійснює неявне перетворення типів
console.log(result); // Результат - 50
```

Рис. 2.4. Ілюстрація слабкої типізації

Як це показано на рис. 2.4, змінна «num» містить число 5, а змінна «str» – рядок «10». Зазвичай, у типових ситуаціях очікується помилка, оскільки сутності належать до різних типів даних. Проте, JavaScript виконує неявне приведення типів, яке в цьому випадку автоматично перетворює рядок на число й успішно виконує операцію.

Отже, неймовірно важливо використовувати мову обережно, особливо під час роботи в команді чи розробки масштабного проєкту, бо розглянута поведінка може спричинити серйозні наслідки.

Тому, для вирішення проблем динамічної природи та забезпечення додаткового рівня захисту, дедалі більше компаній використовують TypeScript для розробки комерційних проєктів. TypeScript – строго типізована мова програмування, яка розширяє JavaScript опціональною анотацією типів і синтаксисом для статичної типізації. Після 2016 року вона починає стрімко набирати популярність та наразі є лідером серед мов, що компілюються у JavaScript (рис. 2.5) [20].

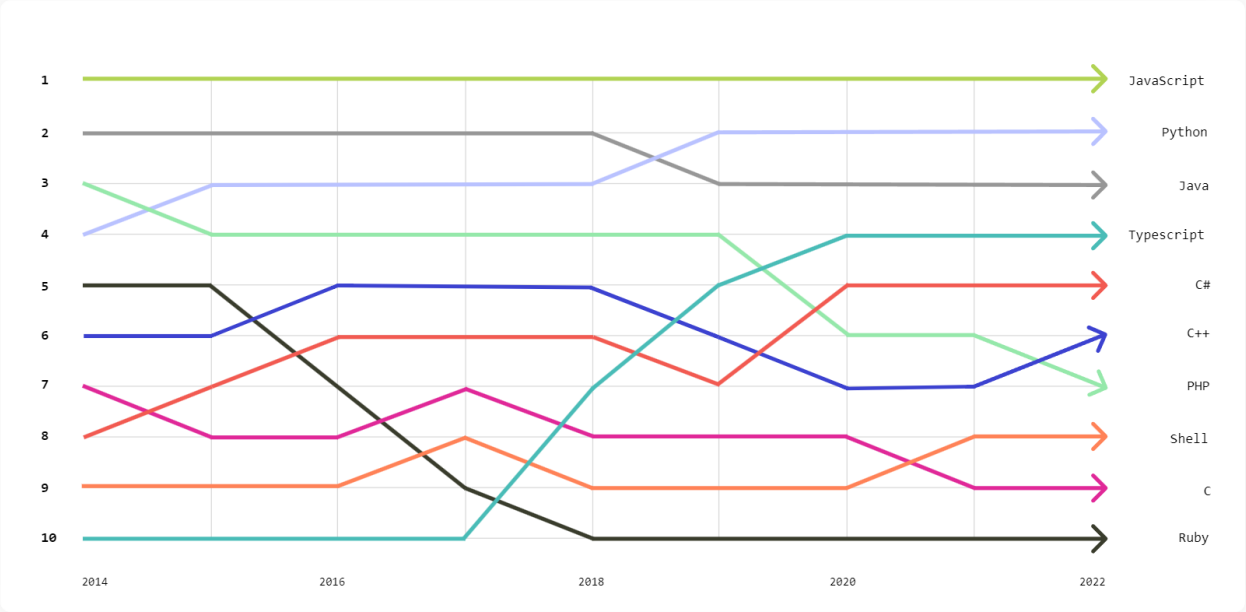


Рис. 2.5. Найпоширеніші мови програмування у 2022 році

Загалом, у період 2021-2022 років, рівень використання TypeScript у світі збільшився на 37.8% і продовжує зростати [20]. Досліджено, що зазвичай фахівці роблять такий вибір завдяки його численним особливостям [21]:

– статичні типи зменшують когнітивне навантаження, пояснюють використання сутностей та ефективно використовуються для документування функцій;

– анотації типів є необов'язковими завдяки підтримці автоматичного їх розпізнавання;

– підтримка повторно використовуваних інтерфейсів, що можуть мати безліч реалізацій;

– сучасні редактори коду, як-от використаний для створення проєкту кваліфікаційної роботи Visual Studio Code, підтримують плагіни TypeScript, що покращують процес розробки;

– популярні фреймворки і бібліотеки переводять кодову базу на TypeScript, завдяки чому зникає потреба в додатковому встановленні анотації типів.

До того ж, згідно з дослідженням Чжен Гао й Ерла Т. Барра з університетського коледжу Лондона, а також Крістіана Берда з Microsoft Research, виявлено, що TypeScript може виявити до 20% програмних дефектів [22]. Отже, зважаючи на те, що такий результат значно може прискорити процес тестування, яке зі свого боку впливає на вирішення 40-80% помилок [23], та інші вищезазначені переваги, обрано TypeScript для реалізації ITHelps.

### 2.3.3. Мотивація вибору фреймворку

У часи стрімкого розвитку веб-розробки, JavaScript-фреймворки є ключовими в створенні сучасних веб-застосунків. Вони пропонують систематизований підхід до побудови динамічних інтерфейсів користувача та вирішують багато низькорівневих проблем, що дозволяє розробникам зосередитися на реалізації бізнес-логіки. Інструменти надають різноманіття функцій, але найголовніша проблема, яку вони вирішують – це постійна синхронізація UI із його актуальним станом. Відсутність оновлення інтерфейсу користувача після кожної зміни в додатку негативно впливає як на зручність використання, так і на стабільність роботи. Впровадження такої важливої логіки за допомогою лише чистої мови програмування, згідно з висновками експертів [24], є нераціональним, а в контексті серйозних проєктів – неможливим. Через те, було проаналізовано найпопулярніші за використанням JavaScript-фреймворки, і надано перевагу React.js згідно з результатами опитування «State of JS 2022» (рис. 2.6) [25]:

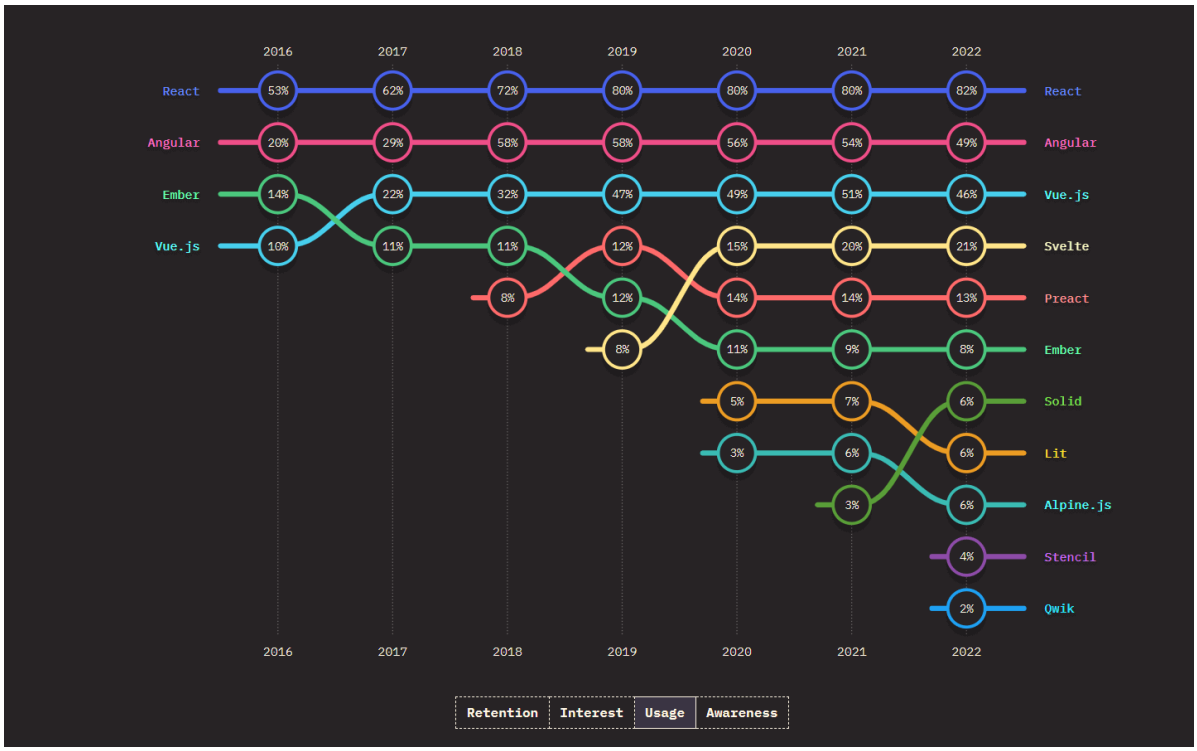


Рис. 2.6. Результати опитування «State of JS 2022»

Важливо зазначити, що React – не фреймворк, а бібліотека для побудови інтерфейсу користувачів, яка базується на компонентному підході, тобто розроблені UI-елементи можна контролювати, оновлювати та повторно використовувати. На відміну від фреймворків, які вимагають чіткої структури та правил для побудови проєкту, React є гнучким. Окрім цього, офіційна документація підкреслює потребу у встановленні сторонніх утиліт для реалізації не менш важливих функцій, як-от маршрутизації або збору даних із сервера [26]. Однак, для розробки повноцінної програми з React, вищезазначений ресурс рекомендує використовувати один із популярних full-stack фреймворків на вибір: Next.js або Remix.js.

Обидва варіанти є привабливими для створення сучасних веб-застосунків. Завдяки схожому інструментарію, незалежно від того, чи пріоритетною є гнучкість, стратегії рендерингу, досвід розробника, обидва мета-фреймворки пропонують унікальні переваги та можливості. Однак, оскільки Remix.js з’явився у 2021 році, він суттєво поступається довготривалому Next.js за показниками популярності (кількість користувачів та учасників спільноти), а

також стабільності експлуатації (якість документації, взаємодія зі сторонніми бібліотеками). Рівень підтримки можна оцінити числом зірок, наданих відповідним репозиторіям на платформі GitHub, яке на момент написання кваліфікаційної роботи в Next.js складає 122,1 тис., а у Remix – 28,2 тис [27, 28].

Отже, для забезпечення довготривалої роботи інтерактивної Q&A платформи ITHelps та можливості гнучкого вибору сторонніх інструментів протягом розробки, обрано full-stack мета-фреймворк Next.js.

### 2.3.4. Обґрунтування доцільності обраної СУБД

У сучасному цифровому світі головними моделями баз даних є реляційна та документо-орієнтована [29]. Для правильного вибору, який відповідатиме використаним технологіям і структурі веб-застосунку ITHelps, а також забезпечення оптимальної продуктивності й масштабованості, необхідно ретельно проаналізувати специфіку використання обох типів.

Реляційні БД зберігають інформацію в таблицях, які часто взаємозв’язані спільними даними (рис. 2.7). Вони використовують стовпці для визначення інформації, що зберігається, та рядки для безпосередньо значень. Найпоширеніший спосіб взаємодії з такою моделлю – це використання мови структурованих запитів SQL. Їх застосовують для виконання операцій створення, читання, оновлення, видалення записів.

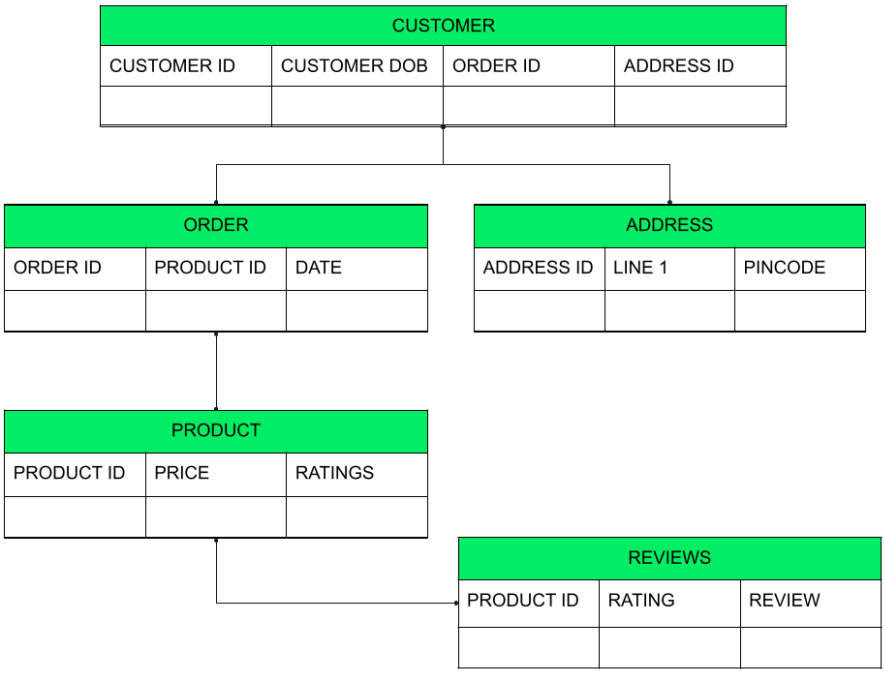


Рис. 2.7. Вигляд зберігання даних у реляційних БД

Переваги реляційної моделі:

– відповідність ACID. Атомарність, узгодженість, ізольованість, довговічність – стандарт, який гарантує надійність транзакцій. Загальний принцип полягає в тому, що якщо одна операція дала збій, то вся транзакція зазнає невдачі, а стан БД повертається до попереднього. Ця методика є чудовою там, де точність є критично важливою, наприклад, у фінансовій або банківській сферах;

– точність даних. Кожна таблиця має стовпець із унікальними значеннями, який називається первинним ключем. За потреби встановлення зв’язків, його можна використовувати в інших таблицях як зовнішній ключ. Така структура допомагає зберегти систему від повторюваних записів.

Недоліки аналізованого типу:

– масштабованість. Реляційні БД традиційно запускаються на окремій єдиній машині. Тож, у разі недостатніх показників потужності або потреби збільшення обсягу даних, необхідно покращувати апаратне забезпечення, що має обмеження і називається вертикальним масштабуванням;

– гнучкість. Визначення схеми, тобто стовпців і типів даних для стовпців, включно з будь-якими обмеженнями необхідно визначити на початку роботи, що не завжди можливо. Подальше внесення змін до структури є дуже складним завданням, бо вимагає модифікації усіх даних, що зі свого боку вимагає тимчасового відключення БД.

Підсумовуючи, специфіка реляційної моделі є недостатньо відповідною для динамічних проєктів, подібних до інтерактивної Q&A платформи ITHelps. Для типових застосунків можна пожертвувати жорстким дотримання принципів ACID задля кращої продуктивності, гнучкості, швидкості оновлення інформації та можливості необмеженого масштабування. Отже, для кінцевого рішення, необхідно розглянути документо-орієнтований принцип.

Документо-орієнтовані бази зберігають дані в JSON-подібних структурах, що підтримують різноманітні типи даних. Зберігання відбувається парами, подібними до «ключ-значення», що пришвидшує виконання операцій з БД, а взаємодія відбувається через API-запити (рис. 2.8).

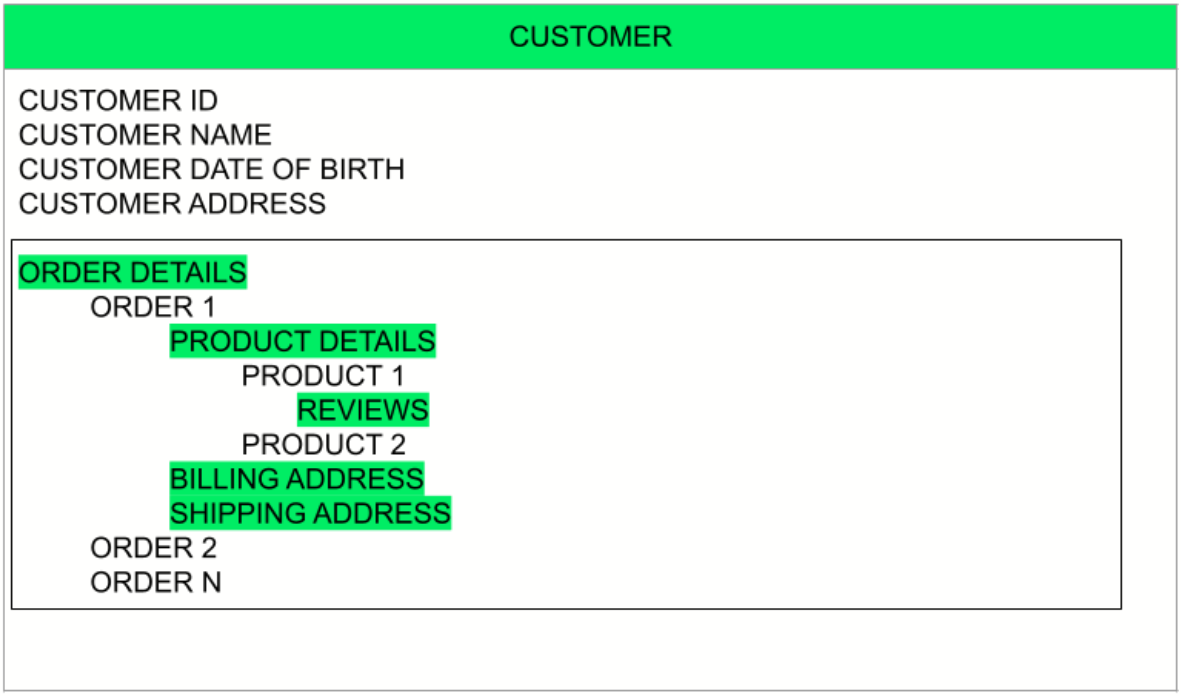


Рис. 2.8. Приклад зберігання даних у документі

Переваги використання документо-орієнтованої моделі, порівнюючи з реляційною:

– швидший темп розробки. Така модель краще підходить для Agile-методології;

– зручне управління різними структурами даних. Завдяки можливості ефективного зберігання структурованих, напівструктурованих і неструктурованих даних, модель дозволяє робити зміни в будь-який момент без значних ресурсних витрат;

– горизонтальне масштабування. Епоха розвитку хмарних обчислень дозволила таким базам реалізувати краще масштабовану архітектуру. Результат досягається завдяки розподіленню сховищ даних та їх обробки на великому кластері комп'ютерів. За необхідності, потужність покращується збільшенням кількості програмно-обчислювальних машин, що є безпроблемним у хмарному середовищі.

Серед основних недоліків аналізованої моделі можна виділити середній рівень надійності через відсутність підтримки ACID, а також неможливість взаємодії за допомогою мови SQL для кращої організації даних.

Отже, оскільки важливими складовими для забезпечення стабільної роботи та зручного процесу розробки Q&A платформи є: передбачувана Agile-методологією можливість змін; зручне масштабування для швидкої та надійної роботи з очікуваним великим об'ємом даних, через потенційну високу активність користувачів; безперебійна взаємодія з іншими технологіями; перевагу надано нереляційному документо-орієнтованому варіанту.

Згідно з результатами дослідження платформи світового масштабу Statista [30], що пропонує доступ до даних та інструментів бізнес-аналітики, проведеним під керівництвом Петрока Тейлора й опублікованим у червні 2024 року, MongoDB є лідером на ринку серед систем управління нереляційними базами даних (рис. 2.9) [31].

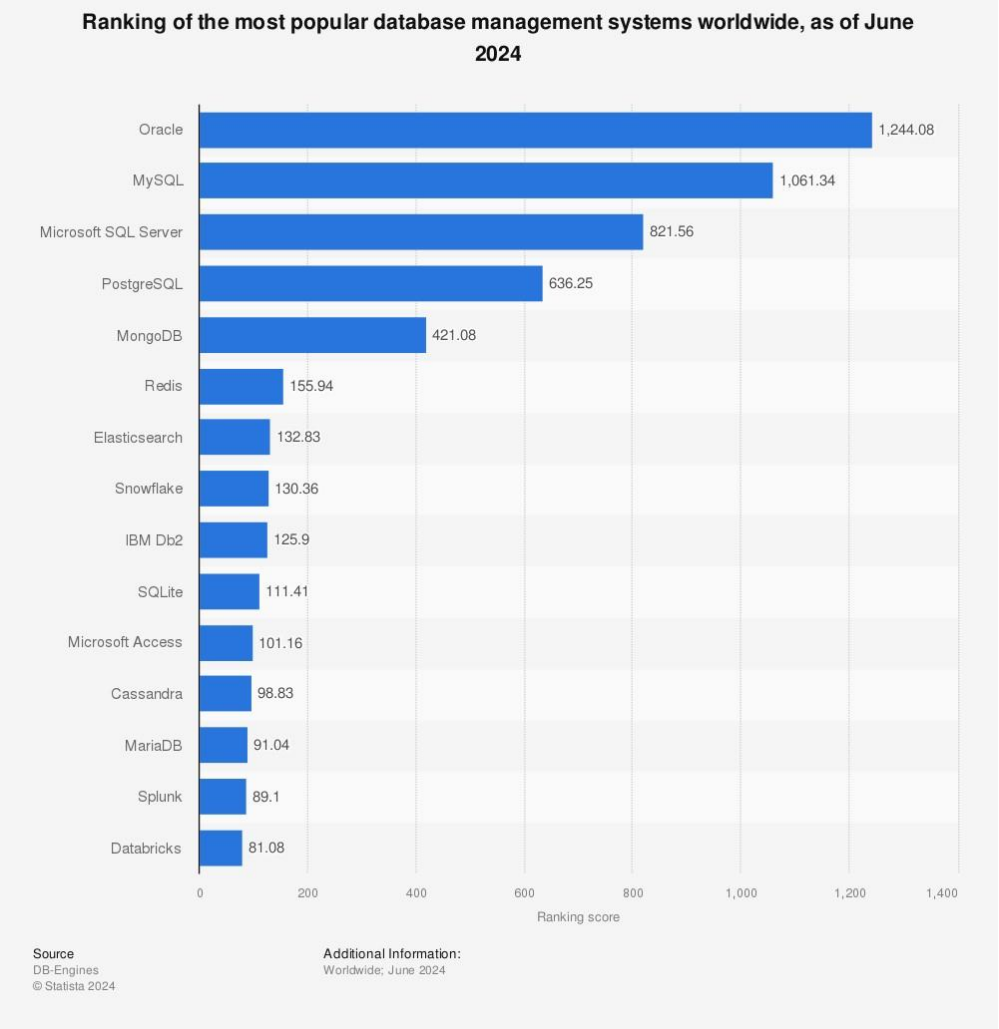


Рис. 2.9. Рейтинг найпопулярніших СУБД у світі

Такий показник можна пояснити відомою простотою використання, масштабованістю та гнучкістю [32]. MongoDB використовує бінарний формат JSON для зберігання, що значно швидше обробляє інформацію ніж звичайний, дозволяє зберігати складні ієрархії та різні типи даних в одному документі. Для взаємодії пропонується використання потужної мови запитів MQL, що є перевагою для JavaScript-розробників завдяки звичному синтаксису. База даних безпроблемно горизонтально масштабується шляхом додавання більшої кількості серверів до кластера, що дозволяє обробляти щораз вищі обсяги даних і трафік користувачів. Крім того, MongoDB має велику спільноту [33], ефективно взаємодіє з Next.js [34] і значною кількістю технологій. Завдяки своїм численним

перевагам, СУБД стає чудовим вибором для використання протягом розробки бакалаврського проєкту.

Хмарне рішення MongoDB Atlas обрано як DBaaS [35], завдяки наданню доступу до роботи з обраним типом БД без необхідності налаштування фізичного обладнання, встановлення ПЗ або управління складною інфраструктурою. Безкоштовний тарифний план хоч і має обмеження, але дозволяє повністю зосередитися на реалізації бізнес-логіки. Крім того, така структура дасть змогу взаємодіяти з іншими хмарними сервісами й розгорнути веб-додаток на хостингу.

Отже, з огляду на співвідношення переваг і недоліків, а також специфіку розроблюваного програмного продукту, обрано СУБД MongoDB та хмарний DBaaS MongoDB Atlas.

### 2.3.5 Характеристика обраних технологій тестування

Тестування є невід'ємною частиною процесу розробки ПЗ, що гарантує роботу функціоналу відповідно до очікувань. Воно слугує захисним заходом для виявлення вразливостей програми перед розгортанням. Завдяки комплексному підходу, можна заощадити на виправленні помилок і повторному тестуванні в майбутньому.

Існує дві основних методології: TDD і code-first. TDD – підхід, за якого тести створюються до початку розробки, задля забезпечення очікуваної поведінки написаного коду в майбутньому. Code-first – протилежний стиль, де спочатку реалізовується логіка, а потім її тестують.

У разі застосування першого методу, розробники зазвичай чітко дотримуються встановлених вимог щодо функціоналу, оскільки будь-які неочікувані зміни будуть моментально виявлені тестами. До того ж, заздалегідь створена структура тестів мотивує розділяти логіку на окремі незалежні частини, які можна інтегрувати в більші складові, що є запорукою чистого й модульного

коду. Тож, розглянутий стиль націлений на виявлення помилок на ранніх стадіях, визначення конкретної стратегії роботи, і стабільне функціонування застосунку.

Однак, застосування TDD стає серйозною перешкодою в проєктах, де неможливо заздалегідь точно сформувати специфікації. Більш того, цей підхід здебільшого уповільнює розробку та обмежує адаптацію до змін, що не узгоджується з динамічним і швидким доставленням функціоналу.

Оскільки реалізація Q&A платформи ITHelps управляється Agile-методологією, де гнучкість є критичною складовою, code-first стає оптимальнішим варіантом, особливо на ранніх фазах розробки. Проте, якщо інформації та впевненості достатньо щодо кінцевого вигляду інкременту, доцільніше застосувати TDD для забезпечення більш стабільної роботи коду.

Отже, доцільно поєднувати обидва підходи, надаючи пріоритет code-first, через динамічну природу створюваного продукту.

Для визначення основних типів тестування пропонується розглянути спеціальну піраміду (рис. 2.10) [36], що структурує тести за рівнями покриття та складається з юніт-, інтеграційних, E2E тестів.

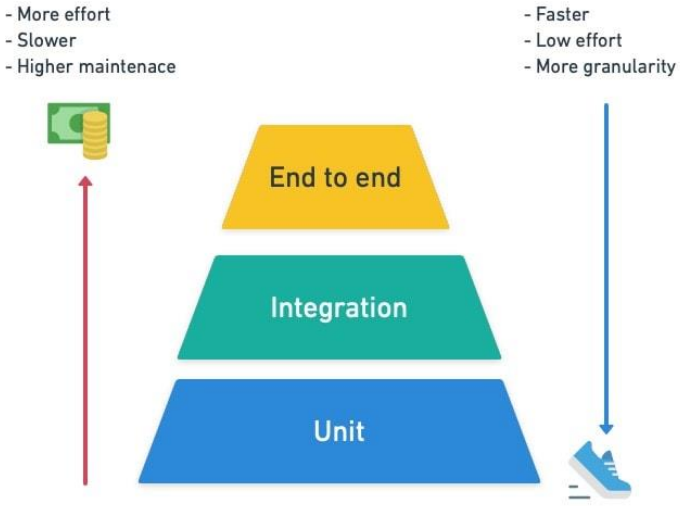


Рис. 2.10. Піраміда тестування

Як це видно з рис. 2.10, ширші рівні вимагають більшої кількості тестів, адже один, у порівнянні з іншими шарами, покриває меншу кількість

функціоналу, його ціна підтримки зростає з основи до верхівки, а час виконання, навпаки, рухається вниз.

Документація Next.js так визначає зазначені в піраміді типи [37]:

– юніт-тестування полягає в перевірці окремих блоків коду, тобто функцій, хуків, або компонентів;

– інтеграційне тестування описує взаємодію декількох програмних фрагментів;

– E2E-тестування імітує середовище, наприклад, браузер, та перевіряє правильність виконання сценаріїв користувачів, наприклад, процес реєстрації.

Враховуючи пріоритет Agile-методології в динамічному розвитку та швидкому випуску продукту, акцент зроблено на юніт- та інтеграційному тестуванні. Це забезпечує високу якість створених компонентів, а також ефективну взаємодію між ними, не перевантажуючи процес дорогими підходами.

Для реалізації визначених методів тестування, офіційна документація Next.js пропонує використати один із фреймворків: Vitest або Jest. Незважаючи на те, що Vitest і Jest підтримують компонентне тестування, рекомендується поєднати один із них із React Testing Library, що пропонує додаткові можливості для роботи з React-компонентами. Останнім часом комбінація з Vitest є пріоритетною на ринку [38], тому потрібно визначити його переваги над Jest:

– швидше виконання тестів. Vitest відомий своєю продуктивністю та оптимізацією запуску тестів, що гарантує ефективність незважаючи на розширення тестового набору [39];

– управління модулями. Jest підтримує CommonJS систему для імпорту й експорту внутрішніх і зовнішніх файлів, що є традиційним підходом, який здебільшого використовується в розробці back-end частини на Node.js. Vitest, з іншого боку, орієнтований на роботу з сучаснішими ECMAScript Modules, що робить його чудовим вибором для популярних фреймворків, як-от Next.js.

Хоча фреймворк має меншу спільноту через свою новітність, його популярність збільшується, завдяки ефективній та стабільній взаємодії з сучасними технологіями. Отже, Vitest і React Testing Library обрано інструментами для code-first юніт- та інтегрованого тестування протягом розробки веб-орієнтованої платформи ITHelps.

### 2.3.6. Огляд обраної платформи для розгортання

Компанія Vercel, виробник мета-фреймворку Next.js, пропонує для відповідних додатків власне хостингове рішення. Із метою підтвердження доцільності та ефективності його використання для деплою веб-орієнтованого застосунку ITHelps, необхідно проаналізувати позитивні й негативні сторони, а також зробити чіткі висновки.

Серед переваг можна виділити:

– нативність. Пропонується нативна, повністю керована інфраструктура, націлена на особливу оптимізацію Next.js застосунків;

– інтеграцію та співпрацю. Vercel взаємодіє з багатьма інструментами, як-от системи керування версіями, бази даних, системи ведення журналів і сповіщень тощо;

– швидкість і безпеку. Компанія забезпечує міжнародною серверною мережею, що сприяє прискореному доставленню та розподіленню вмісту на вебсайті. До того ж, наявний автоматичний HTTPS, що гарантує зручність і безпеку передачі даних.

– економічну ефективність завдяки наявності безкоштовного плану;

– продуктивність. Платформа особлива швидкою роботою серверів, низьким споживанням ресурсів і ефективним використанням енергії [40].

Серед недоліків варто підкреслити:

– обмежену back-end підтримку. Обмежені масштабування та підтримка мов програмування, які реалізують серверну інфраструктуру;

– комерція. Вартість підтримки та використання платформи для IT-компаній починається від 30 тис. доларів на рік [40].

Розглянувши наведену інформацію, можна підсумувати, що хмарне рішення Vercel є дієвим для розгортання інтерактивної Q&A платформи ITHelps, оскільки продукт розробляється з використанням full-stack фреймворку Next.js, який автоматично та захищено керує серверною логікою. Рішення скористатися безкоштовним тарифним планом прийняте з метою швидкого розгортання, ефективного заощадження часу та масштабування.

## 2.4. Опис структури системи та алгоритмів її функціонування

### 2.4.1. Визначення архітектури системи

Після вибору інструментарію для реалізації кваліфікаційної роботи, не менш важливим завданням є визначення системної архітектури. Враховуючи, що основним інструментом для розробки було обрано Next.js, мета-фреймворк, заснований на бібліотеці React, було проведено аналіз відповідних найпопулярніших архітектурних рішень [41]:

– монолітна архітектура. Класичний підхід, за якого вся програма будується як єдина, цілісна одиниця. У контексті Next.js це означає, що всі сторінки, компоненти та API-маршрути керуються в межах одного проєкту. Серед основних переваг, окрім простішого процесу розгортання, можна виділити спрощений процес розробки, завдяки чому програмісти можуть зосередитися на реалізації бізнес-логіки, не турбуючись про інтеграцію різних сервісів. Однак, оскільки всі компоненти знаходяться в одному проєкті, масштабувати такі застосунки складніше, а час збірки збільшуватиметься з розростанням кодової бази;

– мікросервісна архітектура. Модульний стиль, що розбиває програму на менші, самодостатні сервіси, кожний із яких відповідає за певну функціональність, а взаємодія між ними відбувається через API. На відміну від

попереднього варіанту, масштабування стає набагато легшим, процес розробки стає швидшим, бо команди можуть незалежно розробляти різні частини незалежно, а код – чистішим, бо кожен мікросервіс відповідає за один аспект, що сприяє чистоті коду. Проте, керування кількома службами потребує надійного зв'язку та координації, а їх розгортання та підтримка збільшують операційні витрати та зусилля з технічного обслуговування;

– безсерверна архітектура. Зосереджена на використанні безсерверних функцій, тобто розроблених утиліт, які розміщуються та обслуговуються на інфраструктурі хмарних обчислювальних компаній, як-от Vercel, для обробки back-end логіки. Завдяки автоматичному масштабуванню, безсерверні функції є чудовим варіантом для високодинамічних додатків, як-от ITHelps. Серед інших переваг цієї моделі є рентабельність, бо оплата йде тільки за ресурси, які споживаються, а також простота управління, бо хмарні постачальники керують базовою інфраструктурою, що дає можливість розробникам зосередитися на створенні бізнес-логіки. Основні недоліки полягають у затримці холодного старту, що означає затримку запуску безсерверних функцій після періодів бездіяльності, а також – прив'язці до постачальника, що може ускладнити перехід на іншу службу в майбутньому.

Отже, розуміючи сильні та слабкі сторони кожної стратегії, проаналізоване архітектурне рішення головного конкурента Stack Overflow у підрозділі 1.1.2, а також розмір проєкту, бажану продуктивність і вимоги до масштабування, обрано безсерверний монолітний підхід. Таке рішення (рис. 2.11) особливе тим, що кодова база веб-застосунку структурується за монолітними принципами, але також використовуються безсерверні технології для розгортання та виконання певної back-end логіки.

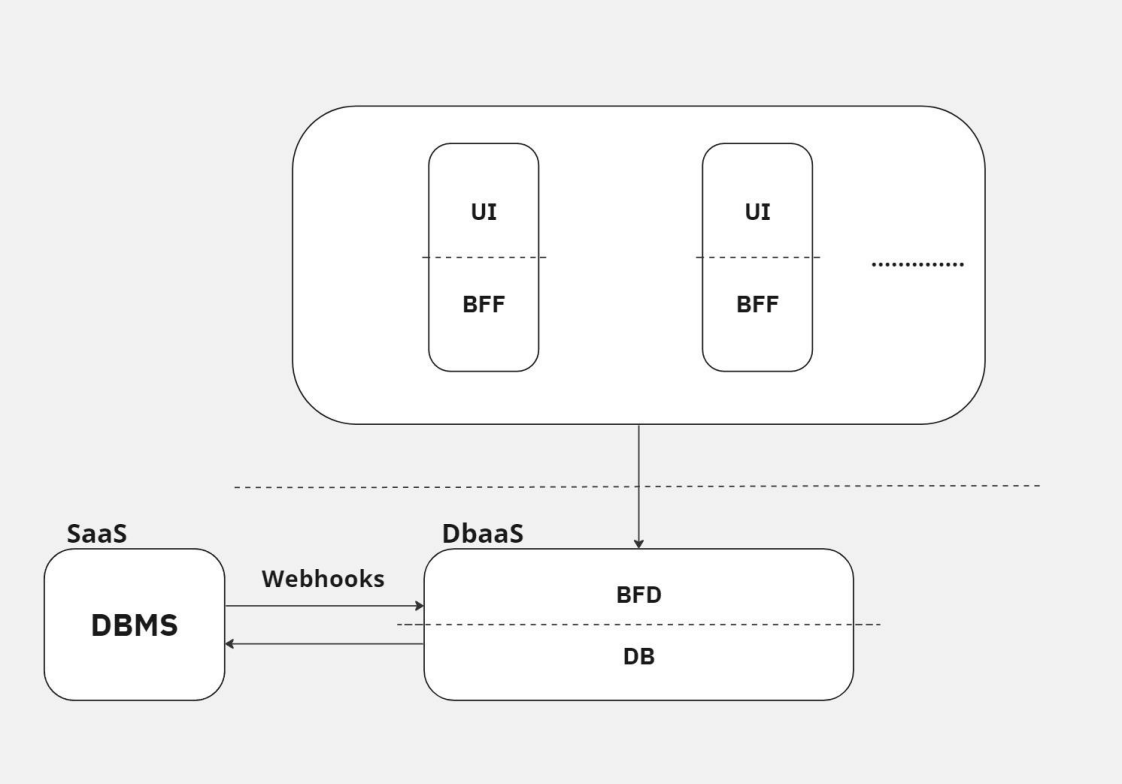


Рис. 2.11. Схема безсерверної монолітної архітектури ITHelps

Безсерверна монолітна архітектура притаманна веб-застосункам, розробленим із використанням мета-фреймворків. Цей інструмент унікальний тим, що він обробляє як front-end частину, так і back-end для користувацького інтерфейсу. Іншими словами, як це показано на рис. 2.11, верхня частина схеми вказує на клієнтську частину, що об’єднує компоненти UI, кожен із яких має відповідний BFF, побудовані за допомогою єдиної технології. У такий спосіб, без необхідності використання сторонніх інструментів для розробки серверної інфраструктури, мета-фреймворк значно покращує продуктивність завдяки зменшенню кількості коду та програмних залежностей.

Нижня частина схеми демонструє безсерверну частину архітектури, що містить сторонні сервіси з якими взаємодіє монолітний сектор додатку, а саме SaaS і DBaaS. Їх мета полягає в забезпеченні необхідної для застосунку функціональності, економлячи час розробників, адже замість написання back-end логіки, достатньо звернутися до цих сервісів за допомогою спеціальних запитів. У ролі DBaaS використовується платформа MongoDB Atlas, що

пропонує повністю керовану та розміщену у хмарі базу даних MongoDB. Clerk застосовується як хмарне SaaS рішення, створене для управління обліковими записами користувачів, а також реалізації процесів реєстрації та автентифікації. До того ж, між обома СУБД застосованих сервісів, необхідно налаштувати синхронізацію даних для забезпечення цілісності. Для реалізації поставленої задачі застосовано вебхуки [42] – HTTP-запити, що автоматично надсилаються між сервісами в момент виникнення подій, пов’язаних зі змінами в облікових записах користувачів.

### 2.4.2. Логічна структура програми

Каркас логічної системи динамічного веб-застосунку ITHelps утворюють 6 головних сторінок: «Домашня», «Спільнота», «Колекція», «Теги», «Профіль», «Запитай». Вони взаємодіють в єдиному середовищі та надають унікальні функціональні можливості різним групам учасників:

– домашня сторінка слугує для пошуку актуальних тем, ознайомлень з усіма обговореннями та активної участі в них;

– веб-сторінка спільноти дозволяє пошук однодумців і перегляду їхніх профілів;

– колекція слугує сховищем для збережених питань;

– сторінка з тегами забезпечує швидкий доступ до тематик за допомогою зберігання відповідних тегів у єдиному місці;

– перегляд власних репутації та профілю дозволений лише авторизованим учасникам платформи, тоді як інші можуть зареєструватися чи ввійти в обліковий запис;

– сторінка «Запитай» містить зручний інтерфейс створення запитань.

На відміну від підрозділу 1.4, який описує доступний функціонал для різних ролей, тут детально розглянуто шлях користувача (англ. User flow) на кожній із головних сторінок, який демонструє можливі сценарії поведінки з головними елементами інформаційної системи. У поточному контексті, шлях

користувача – це послідовність дій від ознайомлення до досягнення встановленої мети на платформі. Для його візуалізації, зручно застосувати діаграми діяльності, розроблені з використанням інструменту PlantUML Server, з якими можна ознайомитися в додатку В.

### 2.4.3. Технічна реалізація

#### 2.4.3.1. Файлова структура

Для досягнення максимальної ефективності у вирішенні поставлених задач, передовсім важливо зосередитись на організації файлової структури проєкту. Такий підхід сприяє створенню підтримуваного середовища, що полегшує масштабування проєкту та підвищує комфортність роботи. Раніше зазначалося, що однією з ключових особливостей фреймворків є встановлення конвенції для організації ієрархії проєкту, яка безпосередньо впливає на стабільну роботу інструменту. Тому, пропонується розглянути особливості структурування Next.js-проєктів на прикладі інтерактивної Q&A платформи ITHelps.

Фреймворк пропонує дві структури [43], але для використаної 14-ї версії, офіційна документація рекомендує застосувати сучасніший підхід, маршрутизатор «app». До обов’язкових папок і файлів верхнього рівня належать:

– app – файловий маршрутизатор, що забезпечує схему навігації веб-додатку. Кожна внутрішня папка є окремим маршрутом у навігаційній системі;

– public – містить статичні ресурси для відображення, зокрема, у поточному проєкті, зберігаються SVG-іконки та PNG-зображення;

– node_modules – папка для зберігання абсолютно всіх програмних залежностей;

– next.config.js – файл налаштувань фреймворку;

– package.json – файл із переліком проєктних залежностей і скриптів;

– middleware.ts – ПЗ-посередник для обробки запитів Next.js;

– .env – основний файл конфігурації змінних оточення;

– .env.local – файл для локального зберігання змінних оточення;

– .eslintrc.json – файл конфігурації ESLint;

– .gitignore – перелік файлів і папок для ігнорування системою управління версіями Git;

– next-env.d.ts – файл декларації типів для Next.js середовища;

– tsconfig.json – файл налаштувань TypeScript.

До того ж, створено додаткові елементи:

– components – папка, яка містить усі розроблені компоненти;

– constants – папка для зберігання незмінних даних;

– context – папка, призначена для зберігання та управління глобальними станами програми;

– coverage – використовується для зберігання звітів щодо покриття коду тестами;

– database – містить схеми для визначення структури даних різних моделей або сутностей у БД;

– lib – папка для багаторазово використовуваних утиліт з метою вирішення різноманітних завдань, наприклад, взаємодії з БД;

– styles – використовується для описання глобальних CSS-стилів;

– test – містить файли для налаштування середовища тестування;

– types – забезпечує повний набір типів для сутностей системи;

– tailwind.config.ts – конфігураційний файл фреймворку Tailwind CSS, що використовується для керування різними аспектами стилізації;

– vitest.config.ts – файл налаштувань фреймворку Vitest для тестування;

– yarn.lock – відіграє важливу роль у керуванні програмними залежностями.

Зі свого боку, папка app має такі основні складові:

– layout.tsx – забезпечує UI-обгортку для вмісту потрібних сторінок навігаційної системи;

– (auth) – містить реалізацію логіки створення облікових записів і реєстрації користувачів. Задля уникнення окремого маршруту «auth», використано синтаксис групування: «(назва групи)». У такий спосіб, папки всередині визначатимуть подальшу URL-структуру. Для розробки зазначеного функціоналу, офіційна інструкція платформи Clerk [44] рекомендує створити дві сторінки: «sign-in» і «sign-up». Кожна з них повинна мати наступну файлову структуру: «папка з назвою маршруту: sign-in або sign-up/[[...назва маршруту]]/page.tsx». Частина шляху «/[[...назва маршруту]]/» є конвенцією для визначення масиву динамічних URL-сегментів. page.tsx – єдина можлива назва файлу, для створення сторінки, що відповідає конкретному маршруту.

– (root) – групує всі сторінки веб-застосунку: «(home)», «ask-question», «collection», «community», «profile», «question», «tags». Папки «profile» і «question» на вищому рівні містять папки «[id]» і «edit». Конвенція «[маршрут]» використовується для створення динамічного сегменту, який у зазначених контекстах відображає сторінку конкретного профілю або запитання, згідно з унікальним ідентифікатором. Частина «edit» демонструє відповідну сторінку для редагування. Інші основні маршрути містять файли loading.tsx і page.tsx, де перший відповідає за миттєве відображення UI-елементів під час очікування даних із сервера.

– api – папка для реалізації кінцевих точок API. Така точка являє собою URL-адресу, на яку клієнтська сторона надсилає запити для взаємодії з сервером, з метою отримання даних або доступу до функцій. Програма містить дві кінцеві точки API: одна для генерації відповідей за допомогою ChatGPT, інша – для обробки вебхуків. Отже, кожна із відповідних папок містить файл із затвердженою фреймворком назвою: «route.ts».

Крім того, варто детальніше розглянути папку «components», де реалізовані компоненти розподілені за групами: «cards», «forms», «shared», «ui»:

– cards – група компонентів, які реалізують візуальне представлення різноманітних карток у інтерфейсі користувача, наприклад, відповідей, учасників, запитань;

– forms – група, яка зберігає реалізацію форм додатку, призначених для збору вхідної інформації, наприклад, щодо запитання, відповіді, або облікового запису;

– shared – набір компонентів, які багаторазово використовуються протягом написання програмного коду;

– ui – папка для зберігання колекції shadcn/ui компонентів.

Зі свого боку, компоненти належать відповідним папкам, що містять файли із наступним вмістом:

– програмний код, написаний із застосуванням синтаксичного розширення мови TypeScript для створення React і Next.js компонентів;

– об’єкт, який містить класи, що використовуються для стилізації елементів фреймворком Tailwind CSS;

– інтерфейс властивостей: засіб опису даних, які компонент очікує від батьківського компонента;

– файл із реалізацією тестових сценаріїв.

Наприкінці потрібно зауважити, що кожна група має index.ts файл для реекспорту, що покращує загальну організацію коду, оскільки всі компоненти групи стають доступними за одним спільним шляхом.

#### 2.4.3.2. Принципи проектування ПЗ

Необхідно зауважити, що разом зі створенням зручної файлової структури, важливо забезпечити високу якість коду. Для досягнення цієї мети застосовано принципи проектування програмного забезпечення SOLID:

– принцип єдиної відповідальності (англ. Single Responsibility Principle) стверджує, що компонент або програмний модуль повинен

реалізовувати одну задачу. Під час розробки, цей принцип застосовано для розподілення коду на спеціалізовані компоненти;

– принцип відкритості/закритості (англ. Open-Closed Principle) полягає в підтримці коду відкритим для розширення та закритим для модифікації. У контексті Next.js застосунку, ця мета досягається створенням абстрактних компонентів, які потім розширюються більш спеціалізованими для конкретних задач;

– принцип підстановки Лісков (англ. Liskov Substitution Principle) гарантує, що нащадки базових компонентів можуть бути використані взаємозамінно. Отже, у поточному контексті цей метод демонструється успадкуванням компонентів, де нащадок реалізує додаткову функціональність зі зберіганням основної структури;

– принцип розділення інтерфейсу (англ. Interface Segregation Principle) стверджує, що вузькоспрямовані інтерфейси кращі за універсальні. Під час розробки, це правило застосовувалось для створення спеціалізованих інтерфейсів властивостей для компонентів, що дотримуються принципу єдиної відповідальності;

– принцип інверсії залежностей (англ. Dependency Inversion Principle) забороняє модулям вищого рівня залежати від модулів нижчого. У реалізованому веб-застосунку це досягається шляхом створення компонента, який інкапсулює та виконує певну логіку. Потім він передає отримані дані своїм нащадкам за правилом односторонньої передачі властивостей.

Окрім вищезазначених, під час розробки, було використано норми KISS (Keep It Simple, Stupid), DRY (Don’t Repeat Yourself) і YAGNI (You Aren’t Gonna Need It), де KISS підкреслює важливість підтримки простоти коду, DRY закликає уникати повторень, а YAGNI застерігає від реалізації непотрібної функціональності.

Отже, із застосуванням цих принципів, вдалося ефективно реалізувати узгоджені задачі та забезпечити можливість подальшого масштабування. Лістинг коду програми наведено в додатку А.

#### 2.4.3.3. Алгоритм розв’язання типової проблеми

Ключовим аспектом розробки веб-додатків є забезпечення ефективної взаємодії між клієнтською частиною, сервером і базою даних. З цією метою, мета-фреймворк Next.js забезпечує розробників серверними діями [45].

Серверні дії (англ. Server Actions) – функції, які викликаються на клієнтській стороні, але виконуються на сервері, що забезпечує надійність і оптимізує процес розробки, усуваючи необхідність створення окремої кінцевої точки API. Їх можна використовувати в різноманітних сценаріях:

– пересилання інформації до БД. Можна записувати дані в базу безпосередньо з клієнтської частини, для чого достатньо лише визначити логіку в серверній дії;

– серверна логіка. Виконання будь-якої бізнес-логіки, пов’язаної з сервером, наприклад, відправлення електронних листів, створення файлів тощо;

– виклик зовнішніх API.

У контексті Q&A платформи ITHelps, пропонується розглянути алгоритм роботи цього методу на прикладі створення запитання (рис. 2.12).


Рис. 2.12. Алгоритм роботи Next.js full-stack застосунку на прикладі створення запитання у формі

Зі схеми видно, що після відправлення форми із запитанням, клієнтська сторона ініціює виклик серверної дії. Вона встановлює з’єднання з MongoDB і передає введені користувачем дані. Отже, в базі створюється новий документ, а на стороні клієнта відбувається оновлення інтерфейсу.

Для наочності, лістинг описаної серверної дії наведено нижче:

**inquire** // серверна дія створення запитання

```
'use server';
async function inquire(params: InquiryParameters) {
 try {
        // Встановлення з’єднання з БД
        establishDBConnection();
        const { subject, body, tags, creator, route } = params;
        // Створення запитання
        const raisedIssue = await Question.create({
                title: subject,
                content: body,
                author: creator,
        });
        const tagsCollection: string[] = [];
        // Створення тегів або їх отримання, якщо вони наявні у БД
        for (const tag of tags) {
                const retrievedTag = await getExistingTag(tag, raisedIssue);
                tagsCollection.push(retrievedTag._id);
        }
        await Question.findByIdAndUpdate(raisedIssue._id, {
                $push: { tags: { $each: tagsCollection } },
        });
        // Фіксація дії користувача "ask_question"
        await Interaction.create({
                user: creator,
                action: 'ask_question',
                question: raisedIssue._id,
                tags: tagsCollection,
        });
        // Збільшення репутації автора на 5 за активність на платформі
        await User.findByIdAndUpdate(creator, { $inc: { reputation: 5 } });
        revalidatePath(route);
 } catch (error) {
        return error;
 }
 }
```

### 2.4.4. Структура бази даних

Документо-орієнтована модель, що розглядається у кваліфікаційній роботі, керується хмарним середовищем MongoDB Atlas і містить 5 колекцій: «users», «questions», «answers», «interactions», «tags»:

– users – містить основні дані про облікові записи користувачів на платформі, а також посилання на ідентифікатори профілів у системі Clerk;

– questions – містить дані кожного запитання, враховуючи зміст, статистику, а також посилання на тих, хто відповів;

– answers – зберігає інформацію щодо кожної відповіді на платформі, враховуючи зміст, статистику, а також посилання на авторів;

– interactions – фіксує та записує взаємодію учасників з конкретним елементом;

– tags – зберігає інформацію щодо тегів на платформі та їх належність конкретним запитанням.

З метою візуалізації розробленої документо-орієнтованої моделі, наявних колекцій із структурами даних, а також відображення зв’язків між документами, створено ER-подібну діаграму з використанням веб-інструменту Prismaliser [46] (рис. 2.13).

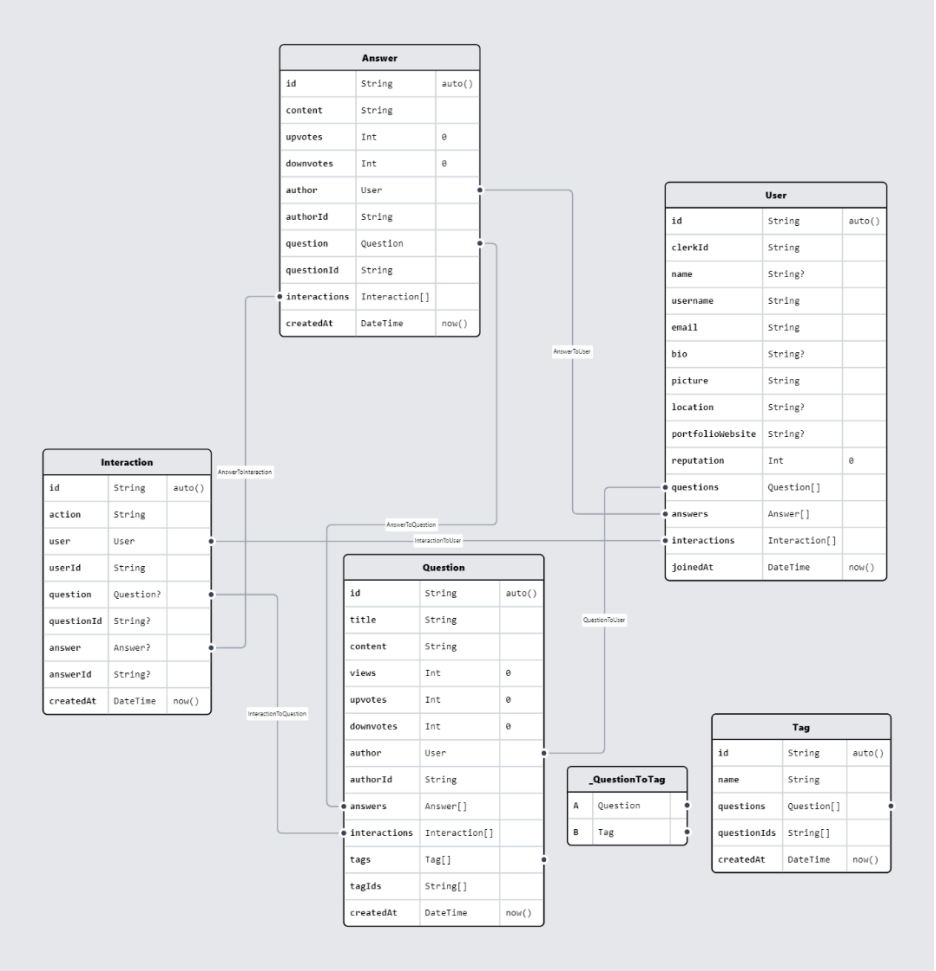


Рис. 2.13. ER-подібна діаграма створеної БД

Перш ніж розглянути тему зв’язків між зображеними документами, необхідно зазначити, що детальну структуру даних кожного, наведено в таблицях у додатку Г. Раніше зазначалося, що документи мають ієрархічну структуру, тому поняття, що використовуються в контексті реляційних БД, наразі неправильно застосовувати в традиційному вигляді. Тому, пропонується розширити концепцію та визначити дві категорії створення зв’язків: вбудовування документів та посилання на них.

Один може організовувати зв’язки з іншими документами або шляхом їхнього присвоєння значенням ключів, або шляхом зберігання посилань на них. Як це видно з рис. 2.13, використано другий підхід, де зберігаються лише унікальні ідентифікатори пов’язаних документів, повну інформацію з яких можна отримати програмними методами. Вибір такої стратегії зумовлений

наступними перевагами: забезпечення кращої цілісності даних, уникнення їхнього дублювання та запобігання надмірному ускладненню ієрархії документів. Ці фактори є критично важливими для ефективної роботи застосунку з високою активністю та високодинамічними змінами.

Як це показано на рис. 2.13, серед типів зв’язків використовуються:

– односпрямований (англ. One-to-Many) – пов’язує один документ із множиною іншого виду. «QuestionToUser» – доречний приклад, який ілюструє, що одне питання може належати лише одному автору, тоді як один користувач може бути автором безлічі запитань. Подібними зразками є: «AnswerToUser», «InteractionToUser», «AnswerToQuestion», «AnswerToInteraction», «InteractionToQuestion»;

– множинний (англ. Many-to-Many) – означає взаємодію між сукупністю документів двох колекцій. Яскравий приклад «_QuestionToTag» показує, що одне питання може мати декілька тегів, тоді як один тег може бути призначений декільком запитанням.

## 2.5. Обґрунтування та організація вхідних та вихідних даних програми

Необхідно зауважити, що процеси формування та попередньої підготовки вхідних і вихідних даних реалізовано з огляду на описані в підрозділі 1.5.1 вимоги. Загалом, взаємодія користувачів з компонентами платформи на клієнтській частині ініціює процес обробки даних сервером. Після успішної валідації, вхідні дані фіксуються в БД, а інтерфейс користувача оновлюється новими елементами. Однак, у разі помилки користувачі побачать відповідне повідомлення. Зазначений механізм обробки обох типів даних є важливою складовою для забезпечення зручності використання продукту та позитивного UX.

З цією метою в системі ITHelps реалізовано форми, які забезпечують оптимізацію різноманітних процесів. Більш того, можливості видалення,

редагування або вибору з наявних опцій зменшують ризик помилитись і полегшують взаємодію з системою. Робота відбувається з текстовими й числовими даними, а також датою.

Варто зазначити, що залежно від типу користувача, система надає різний рівень доступу до функціональності. Наприклад, незареєстрованим користувачам пропонується створити обліковий запис, тоді як учасники спільноти можуть переглянути інформацію щодо своїх профілів. Для ілюстрації підготовки, формування та обробки даних в зазначеному контексті, пропонується розглянути форму реєстрації (рис. 2.14).

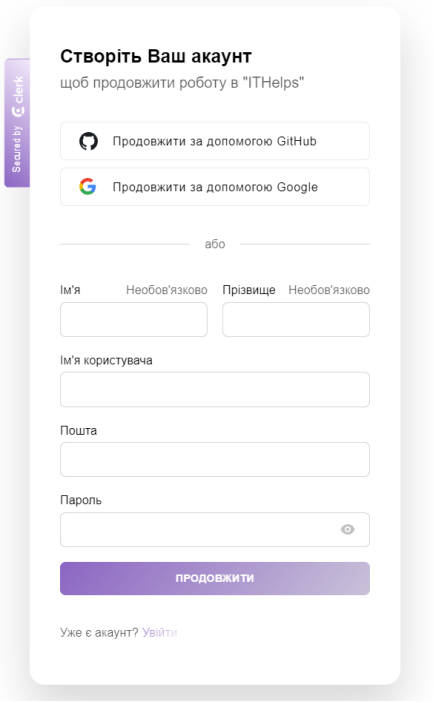


Рис. 2.14. Форма реєстрації в системі

Як це показано на рис. 2.14, є необов’язкові для заповнення поля та інтегровані додаткові сервіси GitHub і Google, що оптимізує процес для відповідної авдиторії. Уведена текстова інформація підлягає обов’язковій перевірці Clerk, згідно з описаними в підрозділі 1.5.2 правилами інформаційної безпеки. За умови успішної валідації, дані будуть записані до внутрішньої бази

Clerk і синхронізовані в окремому документі хмарного середовища MongoDB Atlas завдяки використанню вебхуків (рис. 2.15 – 2.16).

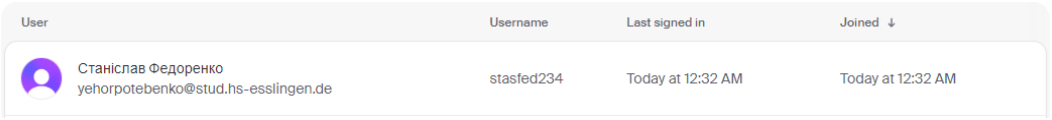


Рис. 2.16. Створений обліковий запис у базі даних системи Clerk


Рис. 2.16. Згенерований MongoDB документ із синхронізованими даними

Як це показано на рис. 2.17, вихідними даними є створена сторінка профілю користувача.

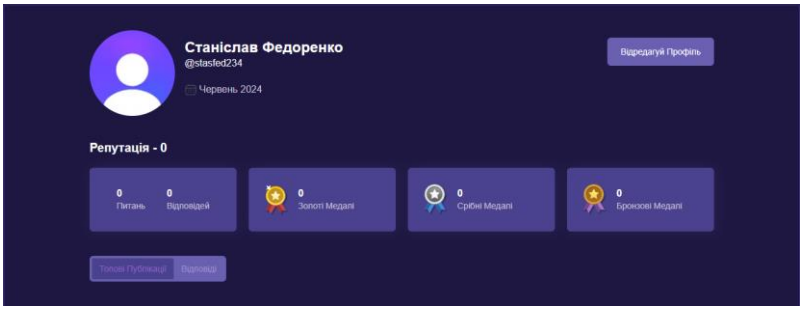


Рис. 2.17. Профіль члена спільноти

Для детальнішого ознайомлення з організацією даних у системі, додатково пропонується розглянути процес створення запитання. З метою надання можливості гнучкого форматування запитів, впроваджено WYSIWYG-редактор TinyMCE [47]. Таке рішення гарантує відповідність зовнішнього вигляду тексту в процесі редагування його кінцевому варіанту. TinyMCE обрано завдяки

зручній інтеграції, підтримці типізації та повному набору перерахованих інструментів у підрозділі 1.5.1. Отже, як це видно з рис. 2.18, форма містить необхідні поля та підказки для формування структурованих запитань.

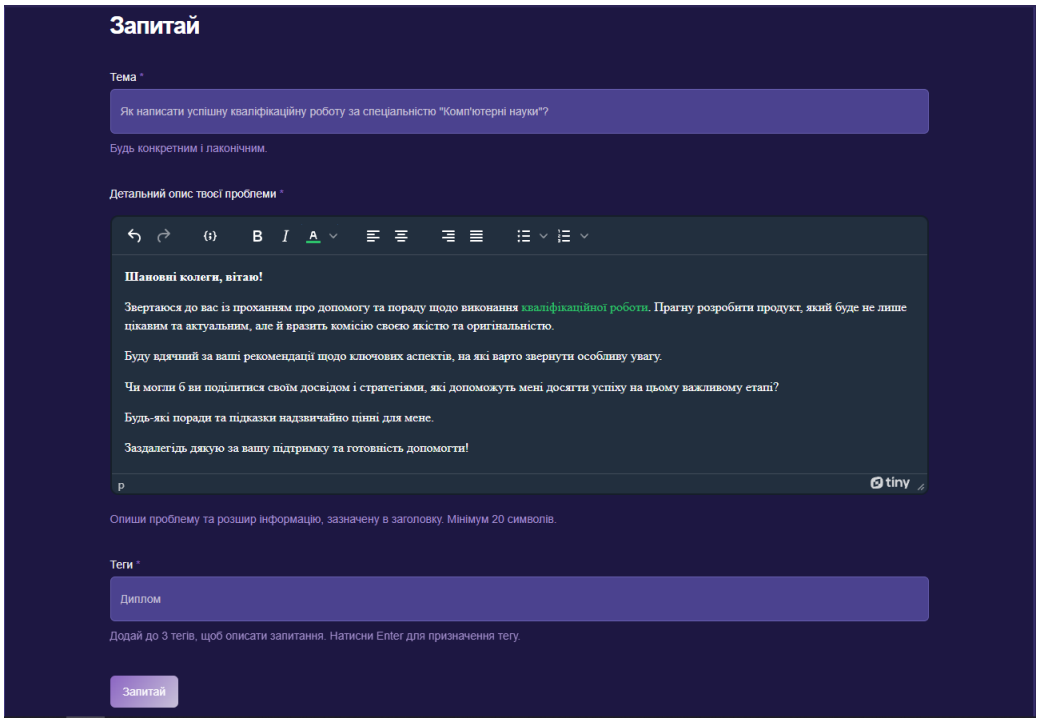


Рис. 2.18. Форма створення запитань

Після успішної обробки, MongoDB генерує окремий документ у відповідній колекції (рис. 2.19).

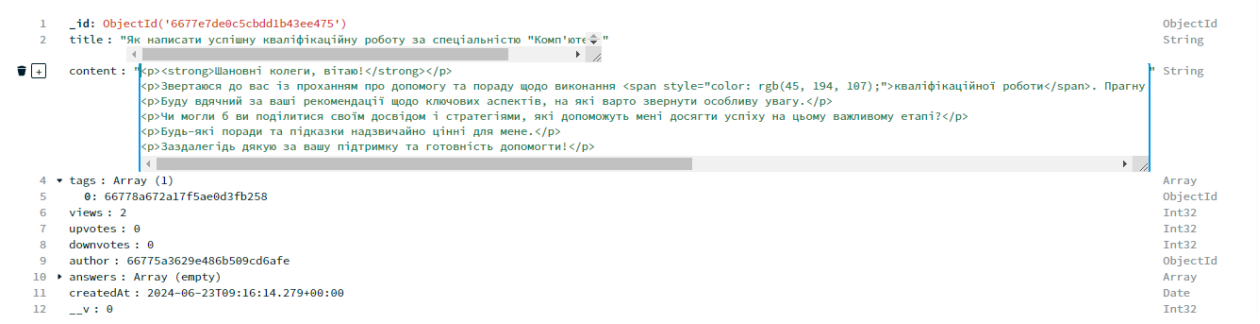


Рис. 2.19. Новий документ у колекції «questions»

Щодо вихідних даних, на головній сторінці з’явиться картка з попередньою інформацією про запитання, що містить тему, ім’я користувача, час публікації та

статистику (рис. 2.20). Крім того, як це показано на рис. 2.21, відкриття картки надає доступ до повної взаємодії зі створеним питанням.

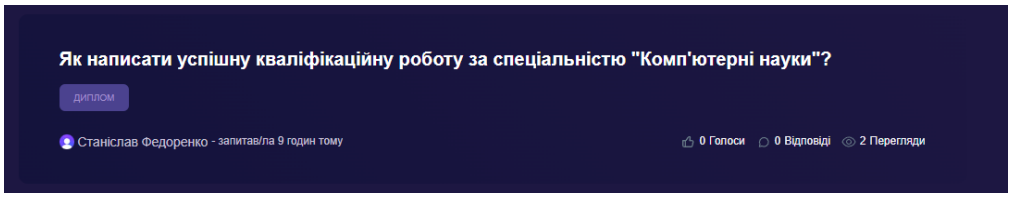


Рис. 2.20. Інформаційна картка запитання

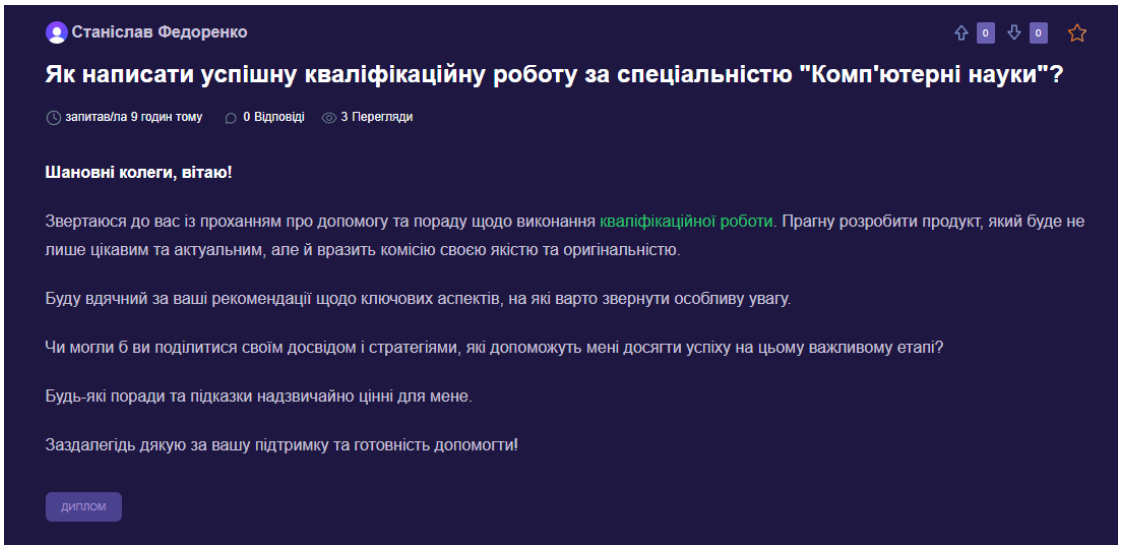


Рис. 2.21. Частина інтерфейсу для взаємодії із запитаннями

Отже, зазначені приклади управління даними окрім забезпечення їхньої цілісності, створюють зручне середовище для користувачів.

## 2.6. Опис розробленої системи

### 2.6.1. Використані технічні засоби

Для розробки та тестування веб-орієнтованого додатку ITHelps використовувалось наступне обладнання: ноутбук моделі «ASUS TUF Gaming 90NR02D1-M11380» [48], монітор «Samsung F24T350FHI» [49], смартфони «Samsung Galaxy M33 5G» [50] і «Apple iPhone 13» [51], дротова миша «ASUS

TUF Gaming M3» [52], механічна клавіатура «TECURS KB510» [53]. Технічні характеристики кожного засобу оформлені в табличному вигляді в додатку Д.

### 2.6.2. Використані програмні засоби

Протягом реалізації кваліфікаційної роботи застосовувались наступні програмні продукти:

– Miro – онлайн-простір для організації командної роботи. ПЗ дозволяє створювати нотатки та елементи інфографіки, що застосовувалось для планування ітерацій та створення візуального представлення ключових складових інформаційної системи;

– PlantUML Server – веб-застосунок для створення UML-діаграм у режимі реального часу;

– Visual Studio Code – невимогливий до ресурсів редактор вихідного коду, розроблений компанією Microsoft;

– Clerk – SaaS рішення, яке пропонує комплексний набір інтерфейсів користувача, гнучких API та адміністративних панелей для автентифікації користувачів і керування обліковими записами;

– Node.js – кросплатформне середовище виконання JavaScript-коду на сервері;

– yarn – пакетний менеджер, призначений для автоматизації процесу управління застосованими залежностями у проєктах;

– MongoDB Atlas – постачальник хмарних послуг DBaaS, який надає власну інфраструктуру для налаштування, розгортання та масштабування документо-орієнтованих MongoDB баз даних;

– Prismaliser – веб-додаток для моделювання ERD;

– Google Lighthouse – автоматизований інструмент, призначений для визначення якості роботи вебсайтів;

– Git – система контролю версій;

– GitHub – хмарний сервіс, створений для управління і збереження програмного коду, а також відстежування та контролювання змін у проєктах;

– Vercel – хмарна платформа з підтримкою безсерверних функцій, що використовується для розгортання веб-сайтів і є найбільш оптимізованою для Next.js застосунків.

Оскільки розроблений веб-додаток виконується в середовищі веб-браузера, а не безпосередньо на операційній системі, для забезпечення кросбраузерної сумісності застосунок запускався в Microsoft Edge, Firefox, Google Chrome, Opera, Safari (рис. 2.22).

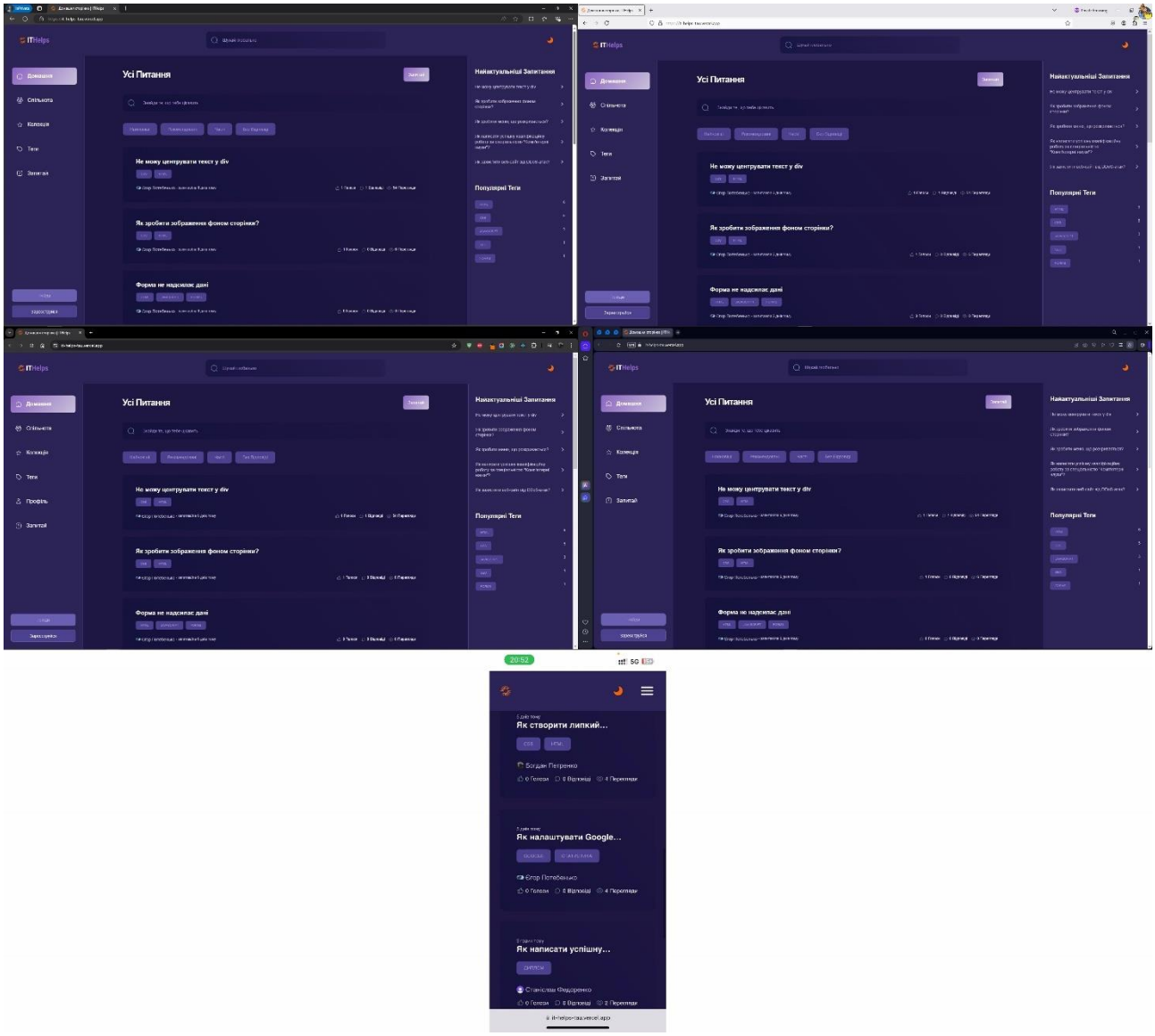


Рис. 2.22. Запуск ITHelps у найпопулярніших браузерах

### 2.6.3. Виклик та завантаження програми

З погляду розробника, для запуску реалізованого веб-застосунку локально, передусім необхідно розпакувати записаний на CD-диск архів із вихідним кодом проєкту. Для зручної роботи з файловою системою та кодом, рекомендується використати Visual Studio Code. Крім того, необхідно переконатися, що встановлені наступні програмні засоби:

- Node.js версії ≥ 18.17.1;
- npm версії ≥ 9.6.7 або yarn ≥ 4.3.1;
- Git.

У кореневій директорії проєкту потрібно встановити всі необхідні залежності командою «npm i» або «yarn». Варто зазначити, що програма взаємодіє з такими зовнішніми сервісами: Clerk, TinyMCE, MongoDB Atlas, OpenAI, і тому використовує змінні оточення, які необхідно самостійно налаштувати. Оскільки вони містять конфіденційні дані, їх приховано від загального огляду, тому потрібно самостійно зареєструватись у кожному із перерахованих сервісів. Після цього, треба створити файл зі змінними локального оточення .env.local і заповнити його за шаблоном .env.example.

Серед сценаріїв запуску програми, у файлі package.json наявні:

- «dev» – запускає додаток у режимі розробки;
- «build» – створює оптимізовану версію застосунку для розгортання;
- «start» – використовується для запуску веб-сервера, призначеного для обслуговування оптимізованої версії застосунку;
- «lint» – запускає ESLint, інструмент для перевірки коду на потенційні помилки та відповідність до правил форматування;
- «test» – запускає виконання тестів.

З метою використання ITHelps в ролі потенційного або наявного учасника спільноти, достатньо перейти за посиланням: https://it-helps-tau.vercel.app/.

### 2.6.4. Опис інтерфейсу користувача

Після переходу за посиланням до веб-орієнтованої Q&A платформи ITHelps, користувач отримує обмежений доступ до одних із основних частин інтерфейсу. Варто розпочати з того, що майже всі сторінки системи успадковують єдиний макет, який містить головні компоненти системи: логотип, основне навігаційне меню, глобальний пошук, переліки популярних запитань і тегів, режим відображення вмісту. Такий підхід забезпечує спільноту постійним доступом до ключових функцій і навігації протягом усього часу взаємодії з додатком, що позитивно впливає на користувацький досвід.

Логотип забезпечує впізнаваність і формує імідж програмного продукту. У цьому середовищі він також слугує посиланням на домашню сторінку, що постійно надає користувачам швидкий доступ до неї. Як це видно з рис. 2.23, компонент може або містити лише графічну складову, або поєднувати графіку з текстом.

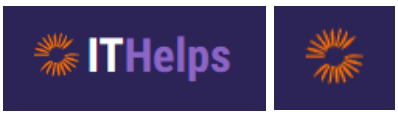


Рис. 2.23. Логотип ITHelps на ПК і мобільних пристроях

Основне навігаційне меню допомагає орієнтуватися та швидко знаходити потрібну інформацію. Елемент забезпечує доступ усім типам користувачів до наступних п’яти сторінок: «Домашня», «Спільнота», «Колекція», «Теги», «Запитай». Неавторизованим або незареєстрованим у системі пропонується увійти або створити обліковий запис, тоді як дійсні учасники мають доступ до додаткової сторінки «Профіль», де можна переглядати інформацію щодо власної активності. На наступній сторінці зображено вигляд адаптованого для різних пристроїв меню, яке можна спостерігати за умови відсутності профілю (рис. 2.24).

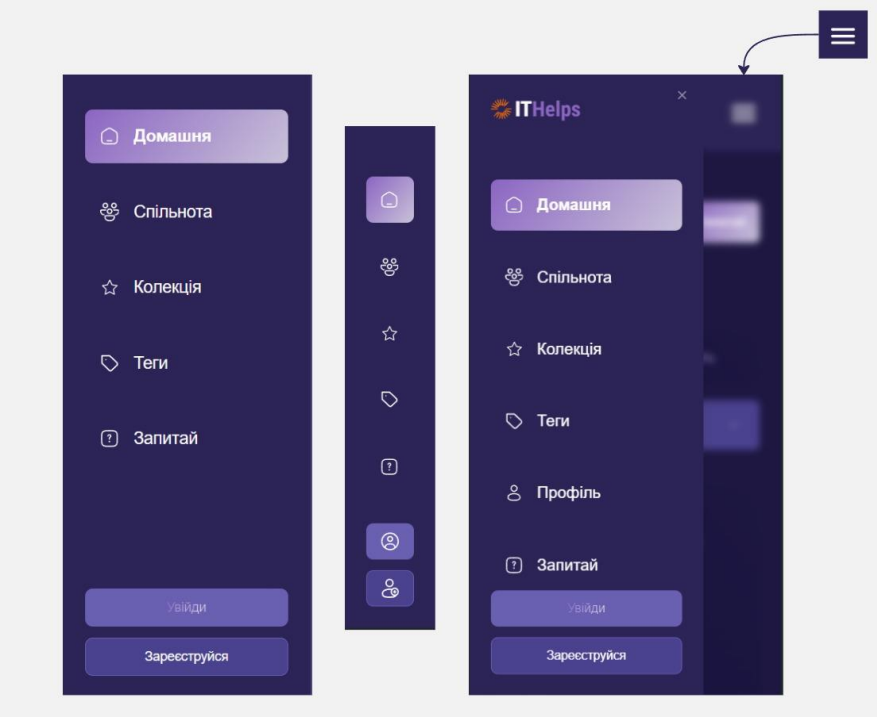


Рис. 2.24. Варіації навігаційного меню на ПК, планшеті та смартфоні

Глобальний пошук дозволяє знайти серед усіх наявних питань, відповідей, тегів, користувачів потрібну інформацію (рис. 2.25).

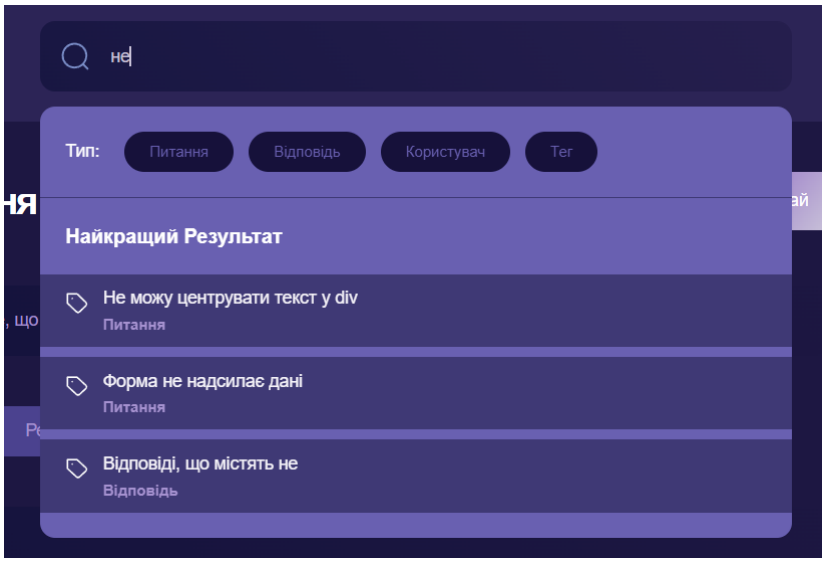


Рис. 2.25. Глобальний пошук

Секції з актуальними запитаннями та популярними тегами, кожна з яких містить не більше 5 інформаційних одиниць, доступні лише на екранах

комп’ютерів. Враховуючи відносно меншу важливість цієї інформації, на мобільних пристроях було вирішено надати перевагу простору та можливості зручнішого перегляду вмісту (рис. 2.26).

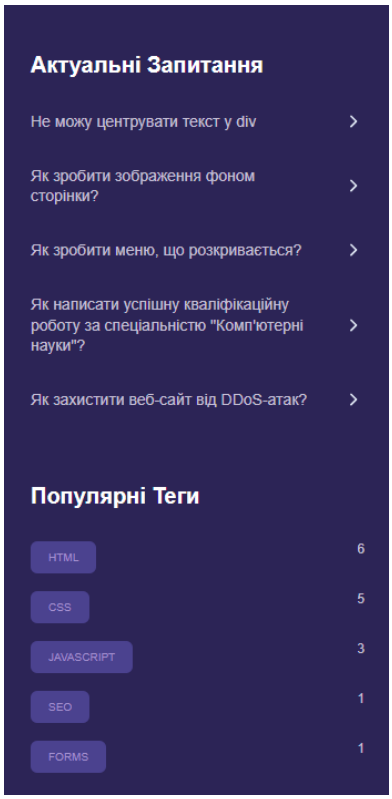


Рис. 2.26. Блоки з актуальними запитаннями та популярними тегами

Щоб задовольнити різні вподобання та залучити ширшу авдиторію, реалізовано можливість обирати колірну тему інтерфейсу: світлу, темну, або системну, яка адаптується до налаштувань браузера (рис. 2.27-2.28).

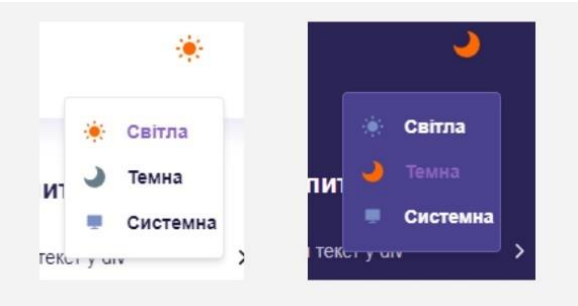


Рис. 2.27. Меню вибору колірної гами

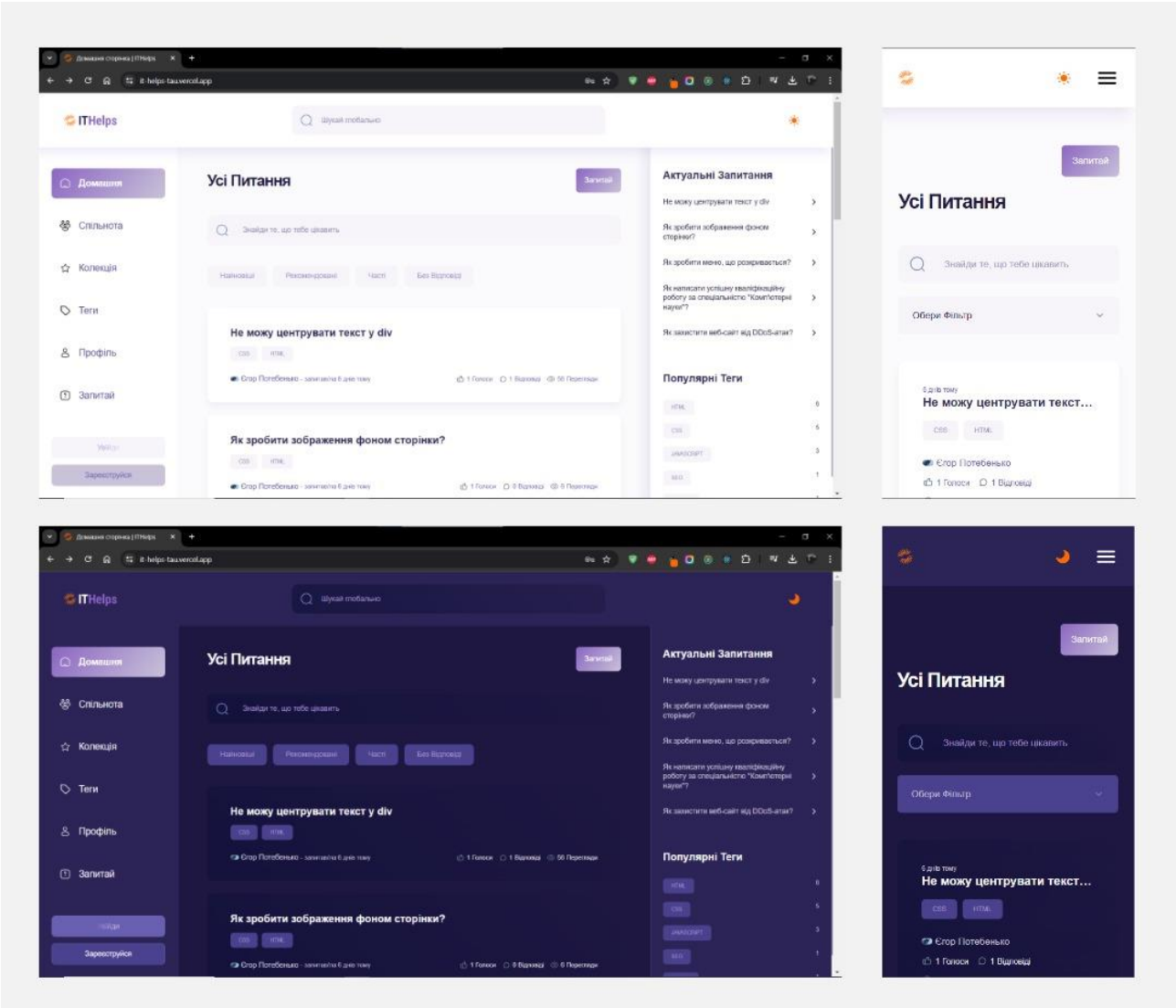


Рис. 2.28. Демонстрація домашньої сторінки у світлій та темній колірних схемах на ПК і смартфоні

У кваліфікаційній роботі зазначалось, що процеси реєстрації та автентифікації забезпечуються зовнішнім сервісом Clerk, який надає не лише відповідний функціонал, а й інтерфейс з можливістю налаштування (рис. 2.29).

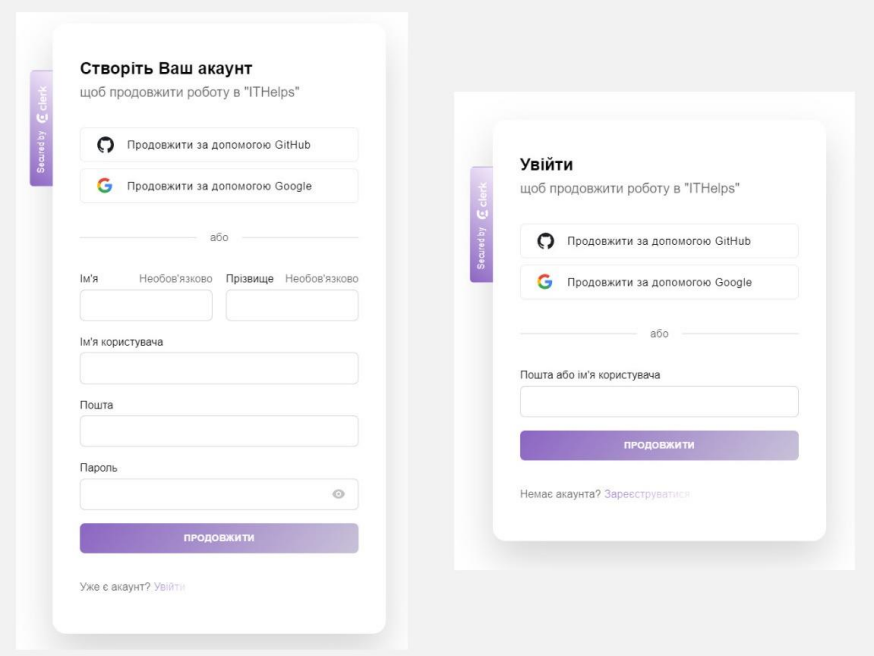


Рис. 2.29. Інтерфейс створення облікового запису та входу в систему

Успішне завершення зазначених етапів надає доступ до керування обліковим записом через інтегрований інтерфейс Clerk та спеціалізовану сторінку «Профіль» (рис. 2.30).

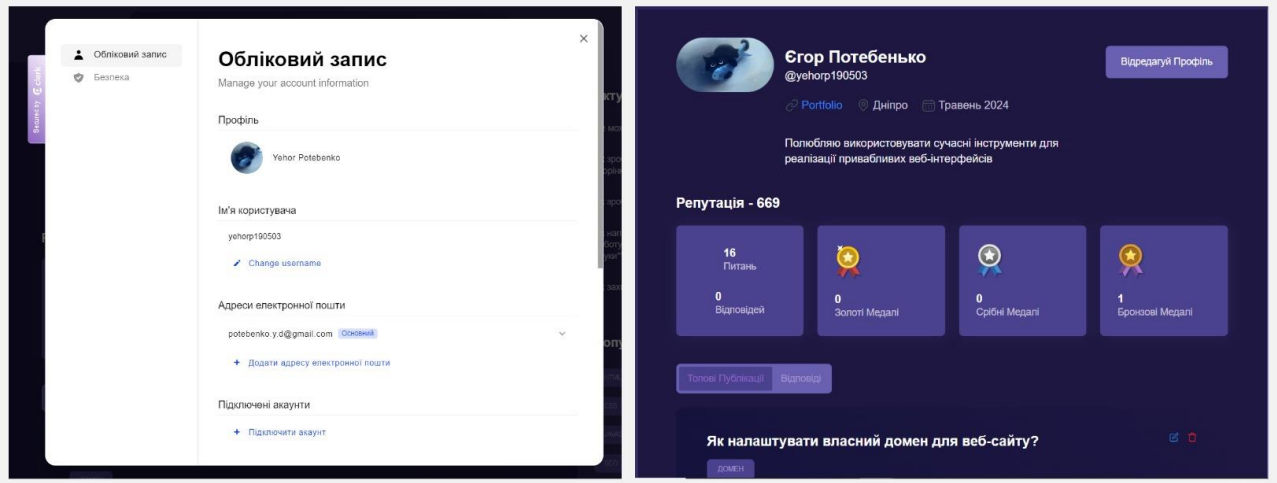


Рис. 2.30. Можливості управління акаунтом

Як це видно з рис. 2.30, користувач отримує бали репутації та медалі за активність на платформі. Наступне зображення ілюструє інтерфейси решти головних сторінок із використанням різних колірних гам на мобільних пристроях і ПК (рис. 2.31).

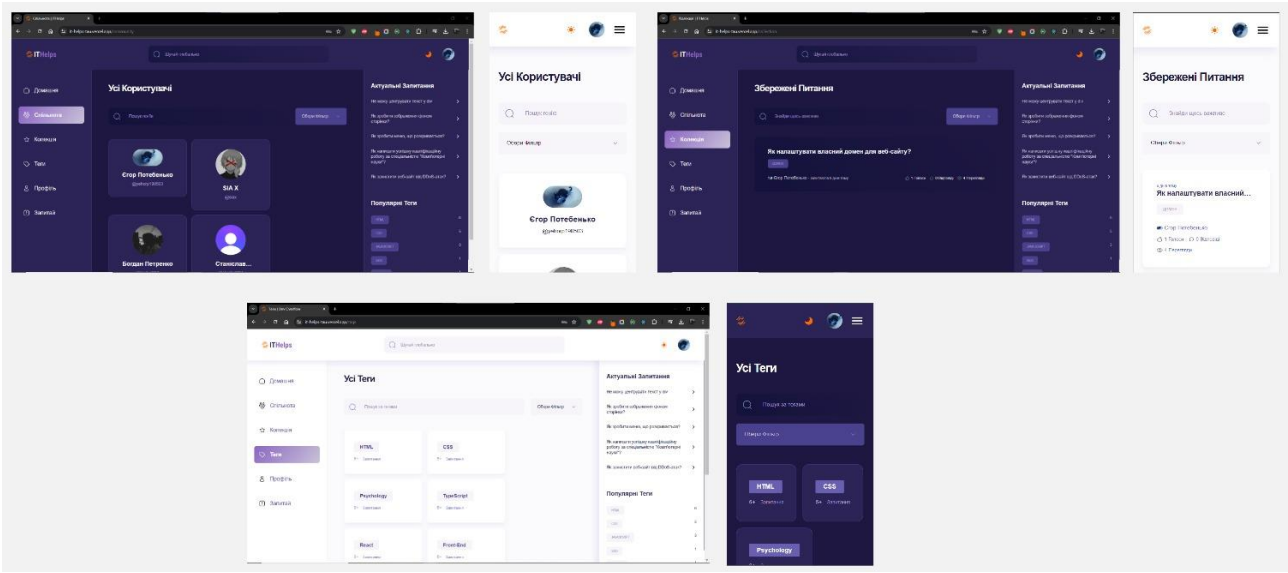


Рис. 2.31. Демонстрація сторінок «Спільнота», «Колекція», «Теги»

У вимогах до розробки бакалаврського проєкту зазначалася необхідність у реалізації фільтрації, компонентів локального пошуку інформації, пагінації для забезпечення зручної навігації, швидкого доступу до потрібних даних та ефективної роботи з великими обсягами інформації відповідно (рис. 2.32).

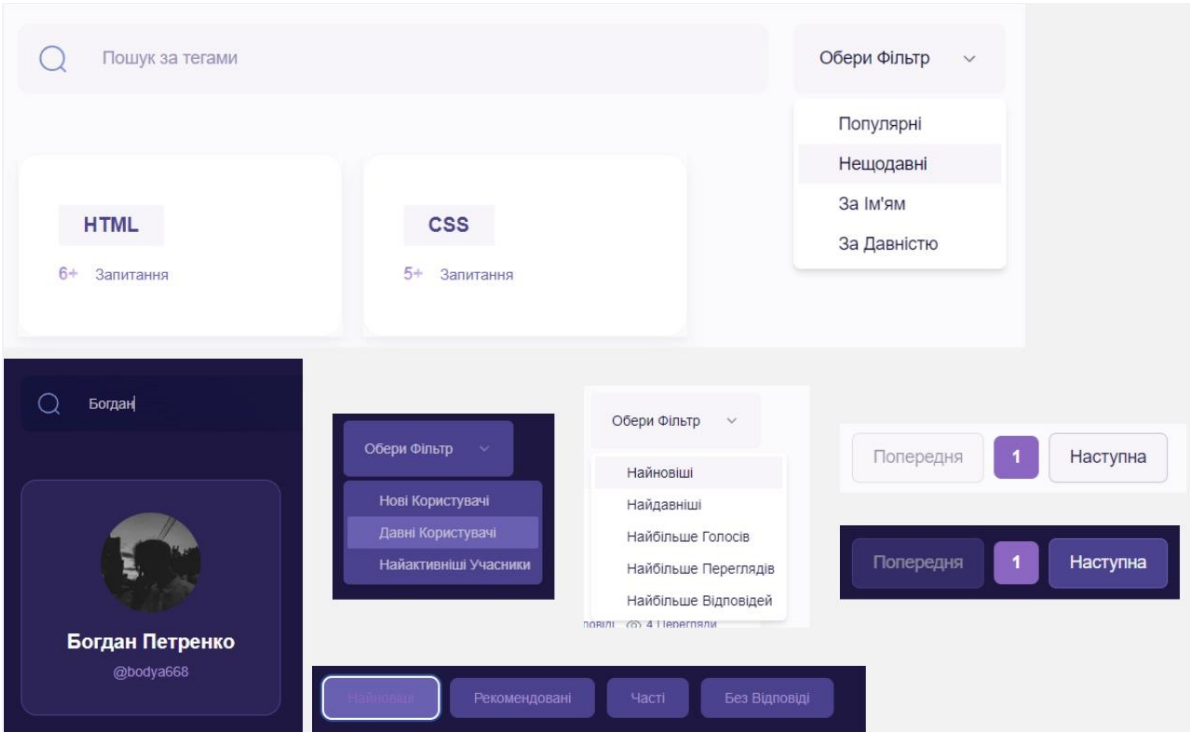


Рис. 2.32. Впроваджені засоби взаємодії з контентом

Реалізовано форми створення запитів, редагування обговорень та власної інформації, що дозволяє спільноті активно взаємодіяти, ділитися знаннями, підтримувати актуальність даних (рис. 2.33).

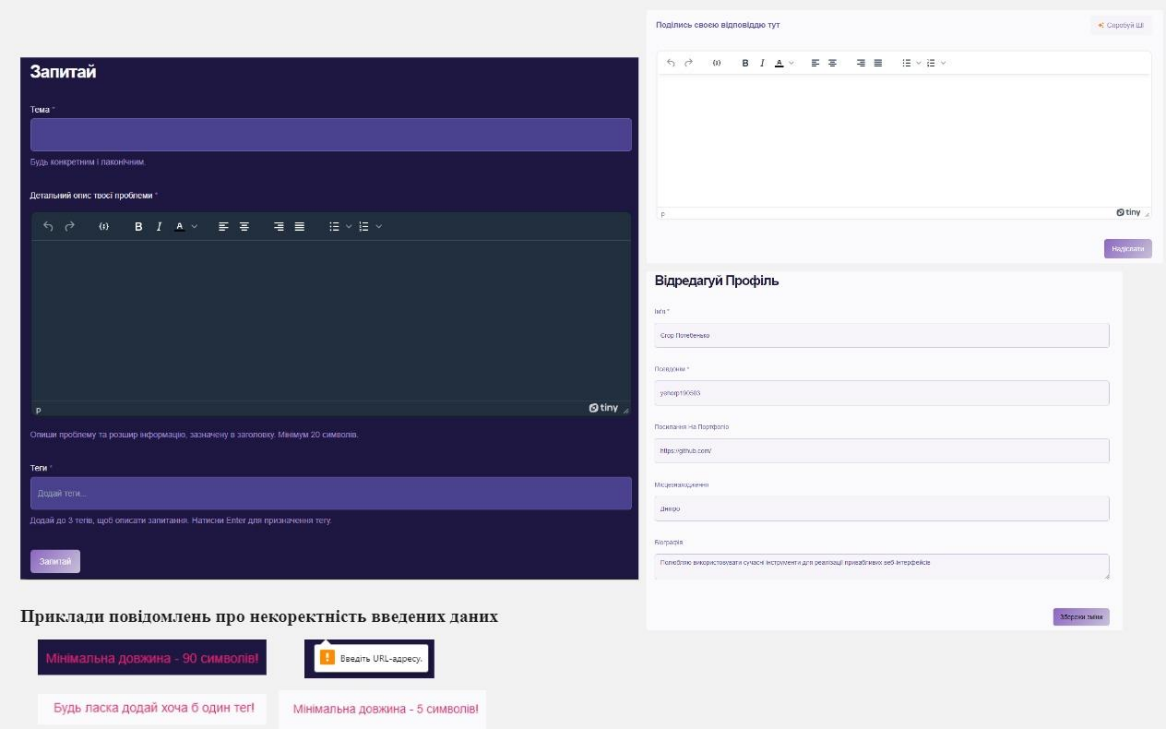


Рис. 2.33. Форми створення запитань, відповідей та актуалізації даних

До того ж, впроваджено елементи голосування за/проти запитань і відповідей, можливості збереження тематик та автоматичної генерації ШІ вмісту, що забезпечує якісну комунікацію. Після подібної взаємодії на екрані з’являється повідомлення (рис. 2.34).

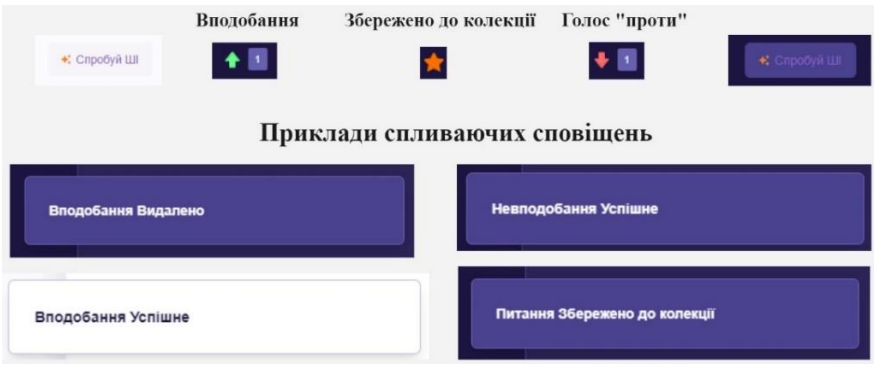


Рис. 2.34. Демонстрація інтерактивних елементів та спливаючих сповіщень

# РОЗДІЛ 3
# ЕКОНОМІЧНИЙ РОЗДІЛ

## 3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1080;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 317,11 грн/год;

Згідно з даними веб-порталу української IT-спільноти DOU [54], раніше ретельно проаналізованої у кваліфікаційній роботі, з кінця 2023 року, актуальна середня місячна зарплата студента на позиції молодшого інженера ПЗ, зі стажем роботи до двох років, що розробляє на мові програмування TypeScript у стартап-компанії, становить 735 $/місяць. У перерахунку на національну валюту за офіційним курсом НБУ [55], середнє значення протягом періоду реалізації продукту (01.09.2023 – 31.05.2024), дорівнювало 27 905,50 грн. Оскільки студент, зарахований на денну форму, може виконувати трудові обов’язки лише у вільний від навчання час [56], здебільшого компанії пропонують договір на умовах неповного робочого часу, тобто 88 год/місяць. Отже, годинна заробітна плата такого працівника – 317,11 грн/год.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
7. вартість машино-години ЕОМ – 0,81 грн/год.

Розрахунок показника ЕОМ виконано з урахуванням актуальних умов на період розробки веб-застосунку ITHelps, що тривала 9 місяців, з 01.09.2023 до 31.05.2024. У цей час тариф на споживання електроенергії – 2,64 грн за 1 кВт/год

[57]. Оскільки, використана під час розробки модель ноутбука «ASUS TUF Gaming 90NR02D1-M11380», у середньому споживає 75 Вт/год, вартість витрат на електроенергію під час роботи: 0,075 * 2,64 = 0,2 грн/год. Крім того, щомісячна абонентна плата за інтернет-пакет «Інтернет 500 Мбіт/с 440» від провайдеру Volia у м. Дніпро – 440 грн/місяць із ПДВ [58]. Врахувавши, що в період з 01.09.2023 до 31.05.2024 середня тривалість місяця була 30 днів, тобто 720 годин, вартість користування інтернетом: 440 / 720 = 0,61 грн/год. Отже, загальна вартість машино-години ЕОМ: 0,2 + 0,61 = 0,81 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою (3.1) [59]:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де $t_o$ – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

$t_u$ – витрати праці на дослідження алгоритму рішення задачі,

$t_a$ – витрати праці на розробку блок-схеми алгоритму,

$t_n$ – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм) визначається за формулою (3.2) [59]:

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де $q$ – передбачуване число операторів (1005),

$C$ – коефіцієнт складності програми (1,3),

$p$ – коефіцієнт корекції програми в ході її розробки (0,06).

$$Q = 1080 \cdot 1,3 \cdot (1 + 0,06) = 1488,24.$$

Витрати праці на вивчення опису задачі $t_u$ обчислюються за формулою (3.3) з урахуванням уточнення опису і кваліфікації програміста [59]:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин.} \qquad (3.3)$$

де $B$ – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2),

$k$ – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності, і у разі менше двох років, дорівнює 0,8.

$$t_u = (1488,24 \cdot 1,2) / (84 \cdot 0,8) = 26,58 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі встановлюються за формулою (3.4) [59]:

$$t_a = \frac{Q}{(20\ldots25) \cdot k}, \text{ людино-годин.} \qquad (3.4)$$

$$t_a = 1488,24 / (24 \cdot 0,8) = 77,51 \text{ людино-годин.}$$

Витрати на складання програми за готовою блок-схемою, також обчислюються за формулою (3.4):

$$t_n = 1488,24 / (22 \cdot 0,8) = 84,56 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання визначаються за формулою (3.5) [59]:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин.} \quad (3.5)$$

$t_{отл}$ = 1488,24 / (5 · 0,8) = 372,06 людино-годин.

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження одного завдання обчислюються за формулою (3.6) [59]:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл}, \text{ людино-годин.} \quad (3.6)$$

$t^{k}_{отл}$ = 1,5 · 372,06 = 558,09 людино-годин.

Витрати праці на підготовку документації, розраховуються за формулою (3.7) [59]:

$$t_{д} = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.7)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису,

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації.

Трудомісткість підготовки матеріалів і рукопису потрібно визначити за формулою (3.8) [59]:

$$t_{др} = \frac{Q}{(15..20) \cdot k}, \text{ людино-годин.} \quad (3.8)$$

$t_{др}$ = 1488,24 / (18 · 0,8) = 103,35 людино-годин.

З іншого боку, трудомісткість редагування, печатки й оформлення документації можна обчислити за формулою (3.9) [59]:

$$t_{до} = 0{,}75 \cdot t_{др}\text{, людино-годин.} \quad (3.9)$$

$t_{до}$ = 0,75 · 103,35 = 77,51 людино-годин.

Згідно з формулою (3.7) витрати праці на підготовку документації становитимуть:

$t_д$ = 103,35 + 77,51 = 180,86 людино-годин.

Отже, згідно з формулою (3.1), трудомісткість розробки ПЗ дорівнюватиме:

$t$ = 50 + 26,58 + 77,51 + 84,56 + 372,06 + 180,86 = 791,57 людино-годин.

### 3.2. Рахунок витрат на створення програми

Формула (3.10) показує, що загальна вартість створення ПЗ складається з оплати праці виконавця програми та витрат машинного часу, необхідного на налагодження програми на ЕОМ [59]:

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{, грн,} \quad (3.10)$$

де $З_{ЗП}$ – витрати на заробітну плату програміста,

$З_{МВ}$ – витрати машинного часу.

Заробітна плата виконавця визначається за формулою (3.11) [59]:

$$З_{ЗП} = t \cdot С_{ПР}\text{, грн,} \quad (3.11)$$

де $С_{ПР}$ – середня годинна заробітна плата програміста (317,11 грн/год).

$З_{ЗП}$ = 791,57 · 317,11 = 251 014,76 грн.

Вартість машинного часу, необхідного для налагодження програми на

ЕОМ, обчислюється за формулою (3.12) [59]:

$$З_{МВ} = t_{отл} \cdot C_{мч}, \text{ грн}, \quad (3.12)$$

де $C_{мч}$ – вартість машино-години ЕОМ, (0,81 грн/год).

$$З_{МВ} = 372{,}06 \cdot 0{,}81 = 301{,}37 \text{ грн}.$$

Отже, згідно з формулою (3.10), витрати на створення ПЗ дорівнюватимуть:

$$К_{ПО} = 251\ 014{,}76 + 301{,}37 = 251\ 316{,}13 \text{ грн}.$$

Очікуваний період розробки ПЗ розраховується за формулою (3.13) [59]:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}, \quad (3.13)$$

де $B_k$ – число виконавців (1);

$F_p$ – місячний фонд робочого часу (за 20-годинного робочого тижня $F_p$=88 годин).

$$T = 791{,}57 / 1 \cdot 88 \approx 9 \text{ міс}.$$

У підсумку, для реалізації інтерактивної Q&A платформи для IT-спільноти ITHelps планується задіяти 791,57 людино-годин. Очікуваний період досягнення мети становить приблизно 9 місяців, а бюджет – 251 316,13 грн.

# ВИСНОВКИ

Відповідно до завдання та мети кваліфікаційної роботи, розроблено веб-орієнтовану інтерактивну Q&A платформу ITHelps для української IT-спільноти.

Проаналізувавши схожі за призначенням рішення в досліджуваній галузі на світовому та локальному ринках, зроблено висновок, що на українському ринку спеціалізованих систем із описаним у роботі функціоналом не існує, а світові лідери мають ряд напрямків, в яких можна покращуватись. Тому створення та формування призначення програмного продукту ґрунтувалося на ідеї суспільного єднання IT-фахівців України, а також перевагах і вразливостях архітектурних рішень всесвітньо відомої мережі для розробників Stack Overflow.

Практичне значення застосунку полягає в забезпеченні цільової авдиторії окремим онлайн-простором для обміну знаннями й досвідом, яка пропонує: ефективний пошук рішень; доступ до якісної та актуальної інформації; можливості для професійного розвитку та співпраці.

Користувачам надається наступний перелік можливостей та інструментів:

– створення облікового запису та взаємодія з власним профілем;
– перегляд, створення й модифікація запитань і відповідей;
– зручний пошук інформації за тегами, учасниками, категоріями;
– системи голосування, репутації та нагород.

Платформа реалізовувалася з використанням мета-фреймворку Next.js і має безсерверну монолітну архітектуру. Під час розробки було інтегровано хмарні сервіси Clerk і MongoDB Atlas, для управління обліковими записами та хмарного розгортання документо-орієнтованих MongoDB БД відповідно.

З огляду на висновки в економічному розділі, планується задіяти 791,57 людино-годин, очікуваний період становить приблизно 9 місяців за умови неповної зайнятості, а бюджет – 251 316,13 грн.

Стратегія подальшого розвитку ITHelps передбачає розширення спектра функціональних можливостей, а також впровадження нових ролей користувачів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Stack Overflow / URL: https://stackoverflow.com/. Дата звернення: 08.05.2024.
2. Stack Overflow CEO on how it became the world’s most popular programming site / URL: https://www.zdnet.com/article/stack-overflow-ceo-on-how-it-became-the-worlds-most-popular-programming-site/. Дата звернення: 08.05.2024.
3. Спірінцев В.В., Потебенько Є.Д. Вплив ШІ-асистентів на комунікацію в ІТ-спільнотах. Наука, освіта та технології: актуальні проблеми теорії та практики: матеріали міжнар. наук.-практ. конф., м. Тампере, Фінляндія, 12 червня 2024. С. 68-70.
4. 2023 Developer Survey / URL: https://survey.stackoverflow.co/2023/. Дата звернення: 09.05.2024.
5. 2022 Developer Survey / URL: https://survey.stackoverflow.co/2022/. Дата звернення: 10.05.2024.
6. Engineering Stack Overflow with Roberta Arcoverde / URL: https://www.youtube.com/watch?v=4OfwvLClyLA. Дата звернення: 10.05.2024.
7. Stack Overflow Architecture / URL: https://medium.com/@techworldwithmilan/stack-overflow-architecture-1712fee5e932. Дата звернення: 10.05.2024.
8. Michele Riva. Real-World Next.js. Birmingham: Packt Publishing Ltd., 2022. 82 p.
9. Динаміка ІТ-індустрії під час війни: результати IT Research Ukraine 2023 / URL: https://itcluster.lviv.ua/dynamika-it-industriyi-pid-chas-vijny-rezultaty-it-research-ukraine-2023/. Дата звернення: 13.05.2024.
10. DOU / URL: https://dou.ua/. Дата звернення: 13.05.2024.
11. The most comprehensive User Management Platform / URL: https://clerk.com/. Дата звернення: 14.05.2024.
12. PlantUML at a Glance / URL: https://plantuml.com/. Дата звернення: 14.05.2024.

13. Database. Deploy a multi-cloud database. / URL: https://www.mongodb.com/products/platform/atlas-database/. Дата звернення: 15.05.2024.

14. Official Vercel Page / URL: https://vercel.com/. Дата звернення: 16.05.2024.

15. Vercel Pricing / URL: https://vercel.com/pricing. Дата звернення: 16.05.2024.

16. MongoDB Pricing / URL: https://www.mongodb.com/pricing. Дата звернення: 16.05.2024.

17. Clerk Pricing / URL: https://clerk.com/pricing. Дата звернення: 16.05.2024.

18. Agile Vs. Waterfall: Which Project Management Methodology Is Best For You? / URL: https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/. Дата звернення: 20.05.2024.

19. 45+ JavaScript Statistics to Accelerate your Business Growth / URL: https://www.esparkinfo.com/blog/javascript-statistics.html/. Дата звернення: 21.05.2024.

20. The top programming languages / URL: https://octoverse.github.com/2022/top-programming-languages/. Дата звернення: 21.05.2024.

21. The TypeScript Tax / URL: https://medium.com/javascript-scene/the-typescript-tax-132ff4cb175b. Дата звернення: 21.05.2024.

22. Zheng Gao, Christian Bird, Earl T. Barr. To Type or Not to Type: Quantifying Detectable Bugs in JavaScript. London: University College London, 2020. 4-9 p.

23. Rotimi-Williams Bello, Soncy Justin Tobi. Software bugs: detection, analysis and fixing. SSRN Electronic Journal. 2023. Vol. 189, No 2. P. 5-9.

24. The deepest reason why modern JavaScript framework exist / URL: https://medium.com/dailyjs/the-deepest-reason-why-modern-javascript-frameworks-exist-933b86ebc445. Дата звернення: 24.05.2024.

25. Стан JS 2022 / URL: https://2022.stateofjs.com/ua-UA/. Дата звернення: 24.05.2024.

26. Official React Documentation / URL: https://react.dev/. Дата звернення: 24.05.2024.

27. Next.js GitHub Repository / URL: https://github.com/vercel/next.js. Дата звернення: 24.05.2024.

28. Remix GitHub Repository / URL: https://github.com/remix-run/remix. Дата звернення: 24.05.2024.

29. Comparing relational and document databases / URL: https://www.prisma.io/dataguide/types/relational-vs-document-databases#how-relational-databases-and-document-databases-organize-data. Дата звернення: 27.05.2024.

30. Official Statista Page / URL: https://www.statista.com/aboutus/. Дата звернення: 28.05.2024.

31. Ranking of the most popular database management systems worldwide, as of June 2024 / URL: https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/. Дата звернення: 27.05.2024.

32. Top 10 Best NoSQL Databases for 2024 / URL: https://em360tech.com/top-10/best-nosql-databases/. Дата звернення: 29.05.2024.

33. Official MongoDB Page / URL: https://www.mongodb.com/. Дата звернення: 29.05.2024.

34. MongoDB with NextJS: A Developer’s Quickstart Guide / URL: https://nextjsstarter.com/blog/mongodb-with-nextjs-a-developers-quickstart-guide/. Дата звернення: 29.05.2024.

35. Database as a Service (DBaaS) Explained / URL: https://www.mongodb.com/resources/basics/databases/database-as-a-service/. Дата звернення: 29.05.2024.

36. Як за допомогою тестів пришвидшити реліз / URL: https://dou.ua/lenta/articles/how-testing-speed-up-release/. Дата звернення: 31.05.2024.

37. Next.js Testing / URL: https://nextjs.org/docs/app/building-your-application/testing. Дата звернення: 01.06.2024.

38. Comparing Next.js testing tools and strategies / URL: https://blog.logrocket.com/comparing-next-js-testing-tools-strategies/. Дата звернення: 31.05.2024.

39. Introducing Vitest: the super fast testing framework / URL: https://codemaker2016.medium.com/introducing-vitest-the-super-fast-testing-framework-c4a86b431f8d/. Дата звернення: 31.05.2024.

40. Vercel as a hosting platform: When It's the best choice and when to look elsewhere / URL: https://focusreactive.com/when-to-host-on-vercel-and-when-not/. Дата звернення: 03.06.2024.

41. Exploring Architecture Patterns for Next.js Applications / URL: https://pavlo-lompas.medium.com/exploring-architecture-patterns-for-next-js-applications-f5550562c63b/. Дата звернення: 04.06.2024.

42. What Are Webhooks And How Do They Work / URL: https://hookdeck.com/webhooks/guides/what-are-webhooks-how-they-work/. Дата звернення: 05.06.2024.

43. Next.js Project Structure / URL: https://nextjs.org/docs/getting-started/project-structure/. Дата звернення: 06.06.2024.

44. Clerk Next.js Quickstart / URL: https://clerk.com/docs/quickstarts/nextjs/. Дата звернення: 07.06.2024.

45. Introduction to Next.js Server Actions / URL: https://makerkit.dev/blog/tutorials/nextjs-server-actions/. Дата звернення: 07.06.2024.

46. Prismaliser Application / URL: https://prismaliser.app/. Дата звернення: 10.06.2024.

47. TinyMCE Trusted WYSIWYG editor / URL: https://www.tiny.cloud/. Дата звернення: 11.06.2024.

48. ASUS TUF Gaming 90NR02D1-M11380 / URL: https://asus.website/catalog/notebooks/FX505DT-BQ443_90NR02D1-M11380/. Дата звернення: 12.06.2024.

49. Samsung F24T350FHI / URL: https://hard.rozetka.com.ua/ua/samsung_lf24t350fhixc/p259976291/. Дата звернення: 12.06.2024.

50. Samsung Galaxy M33 5G / URL: https://www.samsung.com/levant/smartphones/galaxy-m/galaxy-m33-5g-brown-128gb-sm-m336bznimea/. Дата звернення: 12.06.2024.

51. Apple iPhone 13 / URL: https://hotline.ua/ua/mobile-mobilnye-telefony-i-smartfony/apple-iphone-13-256gb-midnight-mlmu3/. Дата звернення: 12.06.2024.

52. ASUS TUF Gaming M3 / URL: https://www.asus.com/accessories/mice-and-mouse-pads/tuf-gaming/tuf-gaming-m3/. Дата звернення: 12.06.2024.

53. TECURS KB510 / URL: https://www.amazon.de/-/en/TECURS-Mechanical-Gaming-Keyboard-Illuminated/dp/B0BG27GMJZ/. Дата звернення: 12.06.2024.

54. DOU. Статистика зарплат програмістів, тестувальників і PM в Україні / URL: https://jobs.dou.ua/salaries/?period=2023-12&position=Junior%20SE&technology=TypeScript&company=4&experience=0-2&education=3/. Дата звернення: 04.06.2024.

55. Національний банк України. Офіційний курс гривні щодо іноземних валют / URL: https://bank.gov.ua/ua/markets/exchangerates/. Дата звернення: 04.06.2024.

56. Робота студентів: все, що потрібно знати / URL: https://trudovi.org.ua/analytics/robota-studentiv-vse-shcho-potribno-znaty/. Дата звернення: 04.06.2024.

57. Мінфін. Тарифи на електроенергію / URL: https://index.minfin.com.ua/ua/tariff/electric/2023-06-01/. Дата звернення: 04.06.2024.

58. Volia. ТАРИФИ НА ІНТЕРНЕТ-ПАКЕТИ ПОСЛУГИ ДОСТУПУ ДО ІНТЕРНЕТУ ТАРИФНОЇ ЛІНІЙКИ «GENERAL» / URL: https://ex.volia.com/user/files/documents/G-internet-tariffs.pdf/. Дата звернення: 04.06.2024.

59. Вагонова О.Г., Романюк Н.Н. Факультет менеджменту, кафедра прикладної економіки: методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи». Дніпро: НТУ «ДП», 2007. 5-8 с.

## ДОДАТОК А

## ЛІСТИНГ ПРОГРАМИ

**layout.tsx** // Головний модуль програми

```
// Імпортування необхідних модулів
import { ThemeProvider } from '@/context/ThemeProvider';
import { Children } from '@/types';
import { ukUA } from '@clerk/localizations';
import { ClerkProvider } from '@clerk/nextjs';
import type { Metadata } from 'next';
import { Inter, Roboto_Condensed } from 'next/font/google';
import '../styles/prism.css';
import './globals.css';
// Налаштування шрифту Inter
const inter = Inter({
    subsets: ['latin', 'cyrillic'],
    weight: ['100', '200', '300', '400', '500', '600', '700', '800', '900'],
    variable: '--font-inter',
});
// Налаштування шрифту Roboto Condensed
const robotoCondensed = Roboto_Condensed({
    subsets: ['latin', 'cyrillic'],
    weight: ['300', '400', '500', '600', '700'],
    variable: '--font-robotoCondensed',
});
// Визначення метаданих для сторінки
export const metadata: Metadata = {
    title: 'ITHelps',
    description:
        'IT-Спільнота для взаємодопомоги та обміну знаннями. Отримуй допомогу, ділись своїми знаннями та співпрацюй із розробниками з усього світу.  Занурся в такі теми, як веб-розробка, розробка мобільних додатків, алгоритми, структури даних і багато іншого.',
    icons: {
        icon: '/assets/images/site-logo.svg',
    },
};
export default function RootLayout({ children }: Children) {
    return (
        <html lang="ua">
            {/* Визначення мови сторінки */}
            <body className={`${inter.variable} ${robotoCondensed.variable}`}>
                {/* Застосування шрифтів глобально */}
                <ClerkProvider
                    localization={ukUA} // Налаштування української локалізації для платформи Clerk
                    appearance={{
                        elements: {
                            formButtonPrimary: 'primary-gradient', // Класи для стилізації основної кнопки у формах
```

```
                                        footerActionLink: 'primary-text-gradient
hover:text-primary-500', // Класи для стилізації посилань у футері
                                    },
                                }}
                            >
                                <ThemeProvider>
                                    {/* Обгортання майбутніх компонентів-нащадків у
ThemeProvider для управління темою */}
                                    {children}
                                </ThemeProvider>
                            </ClerkProvider>
                        </body>
                    </html>
                );
            }
```

**Theme.tsx** //файл із реалізацією компонента «Theme», задля забезпечення вибору колірної гами веб-платформи

```
'use client';
// Імпортування необхідних модулів
import {
    Menubar,
    MenubarContent,
    MenubarItem,
    MenubarMenu,
    MenubarTrigger,
} from '@/components/ui/menubar';
import { themes } from '@/constants';
import { useTheme } from '@/context/ThemeProvider';
import Image from 'next/image';
import React from 'react';
import { themeStyled } from './Theme.styles';
const Theme = (): React.JSX.Element => {
    // Отримання поточного режиму теми та функції для її зміни
    const { mode, setMode } = useTheme();
    return (
        <Menubar className={themeStyled.menuBar}>
            <MenubarMenu>
                <MenubarTrigger className={themeStyled.menuBarTrigger}>
                    {/* Умова для світлої теми, інакше буде обрана темна */}
                    {mode === 'light' ? (
                        <Image
                            src="/assets/icons/sun.svg"
                            alt="іконка світлої теми"
                            width={20}
                            height={20}
                            className={themeStyled.triggerImage}
                        />
                    ) : (
                        <Image
                            src="/assets/icons/moon.svg"
```

```
                                        alt="іконка темної теми"
                                        width={20}
                                        height={20}
                                        className={themeStyled.triggerImage}
                                    />
                                )}
                            </MenubarTrigger>
                            {/* Вміст меню зі стилізацією */}
                            <MenubarContent className={themeStyled.menuBarContent}>
                                {themes.map(item => (
                                    <MenubarItem
                                        key={item.value}
                                        className={themeStyled.menuBarItem}
                                        onClick={() => {
                                            setMode(item.value);
                                            if (item.value !== 'system')
localStorage.theme = item.value;
                                            else localStorage.removeItem('theme');
                                        }}
                                    >
                                        <Image
                                            src={item.icon}
                                            alt={item.value}
                                            width={16}
                                            height={16}
                                            className={`${mode === item.value
&& themeStyled.triggerImage}`}
                                        />
                                        <p
                                            className={`${themeStyled.paragraph}
${
                                                mode === item.value
                                                    ? themeStyled.lightValue
                                                    : themeStyled.darkValue
                                            }`}
                                        >
                                            {/* Мітка для елементу теми */}
                                            {item.label}
                                        </p>
                                    </MenubarItem>
                                ))}
                            </MenubarContent>
                        </MenubarMenu>
                    </Menubar>
                );
            };
            // стандартне експортування компонента «Theme»
            export default Theme;
```

**Theme.styles.ts** // файл зі стилями компонента «Theme»

```
            // Імпортування необхідних модулів
```

```
import { Dynamic } from '@/types';
type StyledKey =
    | 'menuBar'
    | 'menuBarTrigger'
    | 'triggerImage'
    | 'menuBarContent'
    | 'menuBarItem'
    | 'paragraph'
    | 'lightValue'
    | 'darkValue';
export const themeStyled: Dynamic<StyledKey, string> = {
    menuBar: 'relative border-none bg-transparent shadow-none',
    menuBarTrigger:
      'focus:bg-light-900 data-[state=open]:bg-light-900 dark:focus:bg-dark-200 dark:data-
[state=open]:bg-dark-200',
    triggerImage: 'active-theme',
    menuBarContent:
      'absolute right-[-3rem] mt-3 min-w-[120px] rounded border bg-light-900 py-2
dark:border-dark-400 dark:bg-dark-300',
    menuBarItem: 'flex items-center gap-4 px-2.5 py-2 dark:focus:bg-dark-400',
    paragraph: 'body-semibold text-light-500',
    lightValue: 'text-primary-500',
    darkValue: 'text-dark100_light900',
};
```

**Theme.test.tsx** // файл із тестовими сценаріями для компонента Theme

```
import { themes } from '@/constants';
import { ThemeProvider, useTheme } from '@/context/ThemeProvider';
import { Children } from '@/types';
import '@testing-library/jest-dom';
import { fireEvent, render, renderHook, screen } from '@testing-library/react';
import Image from 'next/image';
import { beforeAll, describe, expect, vi } from 'vitest';
import Theme from './Theme';
const MockTheme = (): React.JSX.Element => {
    const { mode, setMode } = useTheme();
    return (
      <div>
            {themes.map(item => (
                  <button
                        key={item.value}
                        onClick={() => {
                              setMode(item.value);
                              if (item.value !== 'system') {
                                    localStorage.theme = item.value;
                              } else {
                                    localStorage.removeItem('theme');
                              }
                        }}
                  >
                        {item.value}
```

```
                    </button>
                ))}

                <Image
                    width={20}
                    height={20}
                    src={themes[0].icon}
                    alt={`${mode} theme icon`}
                />
            </div>
        );
};
describe('Компонент Theme', () => {
    beforeAll(() => {
        Object.defineProperty(window, 'matchMedia', {
            writable: true,
            value: vi.fn().mockImplementation(_ => ({
                matches: false,
            })),
        });
    });
    it('овинен відображатися в межах ThemeProvider', () => {
        render(
            <ThemeProvider>
                <Theme />
            </ThemeProvider>
        );
    });
    it('повинен бути у світлому режимі за замовчуванням', () => {
        const wrapper = ({ children }: Children) => (
            <ThemeProvider>{children}</ThemeProvider>
        );
        const { result } = renderHook(() => useTheme(), { wrapper });
        expect(result.current.mode).toBe('light');
    });
    it('повинен змінювати свій режим зі світлого на темний', () => {
        render(
            <ThemeProvider>
                <MockTheme />
            </ThemeProvider>
        );
        const resultElement = screen.getByRole('img');
        expect(resultElement.getAttribute('alt')).toBe('light theme icon');
        const darkThemeButton = screen.getByText('dark');
        fireEvent.click(darkThemeButton);
        expect(resultElement.getAttribute('alt')).toBe('dark theme icon');
    });
    it('системний режим повинен відповідати бажаному режиму користувача', () => {
        render(
            <ThemeProvider>
                <MockTheme />
```

```
                </ThemeProvider>
            );
            const resultElement = screen.getByRole('img');
            expect(resultElement.getAttribute('alt')).toBe('dark theme icon');
            const systemThemeButton = screen.getByText('system');
            fireEvent.click(systemThemeButton);
            expect(resultElement.getAttribute('alt')).toBe('light theme icon');
        });
    });
```

**Metric.interface.ts** // файл із інтерфейсом властивостей компонента Metric

```
export interface MetricProps {
    imgUrl: string; // URL зображення
    alt: string; // Альтернативний текст для зображення
    value: string | number; // Значення, яке буде відображено в абзаці.
    title: string; // Заголовок, який буде відображено в <span> всередині абзацу
    href?: string; // Опціональний URL для посилання, якщо задано, додається додатковий клас до зображення
    textStyles?: string; // Опціональні додаткові класи для стилізації абзацу
    isAuthor?: boolean; // Опціональний прапорець, якщо true, додається додатковий клас до <span>
}
```

**ContentMetrics.tsx** // файл із реалізацією компонента «ContentMetrics» для відображення найважливішої інформації про учасника

```
// Імпортування необхідних модулей
import Image from 'next/image';
import { MetricProps } from '../Metric.interface';
import { contentMetricsStyled } from './ContentMetrics.styles';
export const ContentMetrics = (props: MetricProps): React.JSX.Element => (
    <>
        {/* Компонент Image для відображення зображення */}
        <Image
            src={props.imgUrl}
            width={16}
            height={16}
            alt={props.alt}
            className={`${contentMetricsStyled.image} ${props.href ? contentMetricsStyled.imageWithHref : ''}`}
        />
        <p className={`${props.textStyles} ${contentMetricsStyled.paragraph}`}>
            {/* Значення тексту, передане від батьківського компонента */}
            {props.value}
            <span
                className={`${contentMetricsStyled.paragraphSpan} ${props.isAuthor ? contentMetricsStyled.paragraphSpanIsAuthor : ''}`}
            >
                {/* Заголовок, переданий від батьківського компонента */}
                {props.title}
            </span>
        </p>
```

```
        </>
    );
```

**ContentMetrics.styles.tsx** // файл зі стилями компонента «ContentMetrics»

```
    // Імпортування необхідних модулей
    import { Dynamic } from '@/types';
    type StyledKey =
        | 'image'
        | 'imageWithHref'
        | 'paragraph'
        | 'paragraphSpan'
        | 'paragraphSpanIsAuthor';
    export const contentMetricsStyled: Dynamic<StyledKey, string> = {
        image: 'object-contain',
        imageWithHref: 'rounded-full',
        paragraph: 'flex items-center gap-1',
        paragraphSpan: 'small-regular line-clamp-1',
        paragraphSpanIsAuthor: 'max-sm:hidden',
    };
```

**ContentMetrics.test.tsx** // файл із тестовими сценаріями для компонента «ContentMetrics»

```
    // Імпортування необхідних модулей
    import '@testing-library/jest-dom';
    import { render, screen } from '@testing-library/react';
    import { MetricProps } from '../Metric.interface';
    import { ContentMetrics } from './ContentMetrics';
    describe('Компонент ContentMetrics', () => {
        const defaultProps: MetricProps = {
            imgUrl: '/path/to/image.png',
            alt: 'Test Image',
            value: '123',
            title: 'Test Title',
            textStyles: 'text-style-class',
            href: '',
            isAuthor: false,
        };
        it('відображає зображення з правильними src, шириною, висотою та alt', () => {
            render(<ContentMetrics {...defaultProps} />);
            const image = screen.getByRole('img', { name: /test image/i });
            expect(image).toBeInTheDocument();
            expect(image).toHaveAttribute('src', defaultProps.imgUrl);
            expect(image).toHaveAttribute('width', '16');
            expect(image).toHaveAttribute('height', '16');
            expect(image).toHaveAttribute('alt', defaultProps.alt);
        });
        it('відображає абзац з правильними текстом і назвами класів', () => {
            render(<ContentMetrics {...defaultProps} />);
            const paragraph = screen.getByText(defaultProps.value);
            expect(paragraph).toBeInTheDocument();
            expect(paragraph).toHaveClass(defaultProps.textStyles as string);
```

```
            expect(paragraph).toHaveClass('flex items-center gap-1');
        });
        it('відображає span з правильними заголовком і назвами класів', () => {
            render(<ContentMetrics {...defaultProps} />);
            const span = screen.getByText(defaultProps.title);
            expect(span).toBeInTheDocument();
            expect(span).toHaveClass('small-regular line-clamp-1');
        });
        it('застосовує додатковий клас до зображення, якщо надано атрибут href', () => {
            const propsWithHref = { ...defaultProps, href: 'https://example.com' };
            render(<ContentMetrics {...propsWithHref} />);
            const image = screen.getByRole('img', { name: /test image/i });
            expect(image).toHaveClass('rounded-full');
        });
        it('застосовує додатковий клас до span, якщо isAuthor = true', () => {
            const propsWithAuthor = { ...defaultProps, isAuthor: true };
            render(<ContentMetrics {...propsWithAuthor} />);
            const span = screen.getByText(defaultProps.title);
            expect(span).toHaveClass('max-sm:hidden');
        });
    });
```

**Metric.tsx** // файл із реалізацією компонента «Metric», обгортки компонента «ContentMetrics»

```
    // Імпортування необхідних модулей
    import Link from 'next/link';
    import { ContentMetrics } from './ContentMetrics/ContentMetrics';
    import { MetricProps } from './Metric.interface';
    import { metricStyled } from './Metric.styles';
    export const Metric = (props: MetricProps): React.JSX.Element =>
        // Перевірка щодо наявності значення href у властивостях
        props.href ? (
            // Якщо href присутній, компонент Link використовується для створення посилання
            <Link href={props.href} className={metricStyled.linkBody}>
                <ContentMetrics {...props} />
            </Link>
        ) : (
            // Якщо href відсутній, використовується div для обгортки вмісту
            <div className={metricStyled.divBody}>
                <ContentMetrics {...props} />
            </div>
        );
```

**Metric.styles.ts** // файл зі стилями компонента «Metric»

```
    // Імпортування необхідних модулей
    import { Dynamic } from '@/types';
    type StyledKey = 'linkBody' | 'divBody';
    export const metricStyled: Dynamic<StyledKey, string> = {
        linkBody: 'flex-center gap-1',
        divBody: 'flex-center flex-wrap gap-1',
```

```
};
```

**Pagination.interface.ts** // файл із інтерфейсом властивостей компонента «Pagination»

```
export interface PaginationProps {
    pageNumber: number; // Номер поточної сторінки
    isNext: boolean; // Прапорець, який вказує на наявність наступної сторінки
}
```

**Pagination.tsx** // файл із реалізацією компонента пагінації

```
'use client';
// Імпортування необхідних модулів
import { Button } from '@/components/ui/button';
import { setURLQuery } from '@/lib/utils';
import { AppRouterInstance } from 'next/dist/shared/lib/app-router-context.shared-runtime';
import {
    ReadonlyURLSearchParams,
    useRouter,
    useSearchParams,
} from 'next/navigation';
import { PaginationProps } from './Pagination.interface';
import { paginationStyled } from './Pagination.styles';
// Функція для обробки навігації між сторінками
const navigateHandler = (
    pageNumber: PaginationProps['pageNumber'],
    queryParameters: ReadonlyURLSearchParams,
    routeHandler: AppRouterInstance,
    route: string
): void => {
    const followingPageNumber =
        route === 'prev' ? pageNumber - 1 : pageNumber + 1;
    const refreshedUrl = setURLQuery({
        parameters: queryParameters.toString(),
        property: 'page',
        data: followingPageNumber.toString(),
    });
    routeHandler.push(refreshedUrl);
};
export const Pagination = (props: PaginationProps): React.JSX.Element => {
    const routeHandler = useRouter();
    const queryParameters = useSearchParams();
    return (
        // Основний контейнер компонента зі стилями
        <div className={paginationStyled.body}>
                {/* Кнопка "Попередня" */}
                <Button
                        disabled={props.pageNumber === 1}
                        onClick={() =>
                                navigateHandler(
                                        props.pageNumber,
                                        queryParameters,
                                        routeHandler,
```

```
                                        'prev'
                                    )
                                }
                                className={paginationStyled.prevButton}
                            >
                                <p className={paginationStyled.prevButtonText}>Попередня</p>
                            </Button>
                            {/* Відображення поточного номера сторінки */}
                            <div className={paginationStyled.center}>
                                <p
className={paginationStyled.centerPageNumber}>{props.pageNumber}</p>
                            </div>
                            {/* Кнопка "Наступна" */}
                            <Button
                                disabled={!props.isNext}
                                onClick={() =>
                                    navigateHandler(
                                        props.pageNumber,
                                        queryParameters,
                                        routeHandler,
                                        'next'
                                    )
                                }
                                className={paginationStyled.nextButton}
                            >
                                <p className={paginationStyled.nextButtonText}>Наступна</p>
                            </Button>
                        </div>
                    );
                };
```

**Pagination.styles.ts** // файл зі стилями компонента «Pagination»

```
                // Імпортування необхідних модулів
                import { Dynamic } from '@/types';
                type StyledKey =
                    | 'body'
                    | 'prevButton'
                    | 'prevButtonText'
                    | 'center'
                    | 'centerPageNumber'
                    | 'nextButton'
                    | 'nextButtonText';
                export const paginationStyled: Dynamic<StyledKey, string> = {
                    body: 'flex w-full items-center justify-center gap-2',
                    prevButton:
                        'light-border-2 btn flex min-h-[36px] items-center justify-center gap-2 border',
                    prevButtonText: 'body-medium text-dark200_light800',
                    center:
                        'flex items-center justify-center rounded-md bg-primary-500 px-3.5 py-2',
                    centerPageNumber: 'body-semibold text-light-900',
                    nextButton:
```

```
        'light-border-2 btn flex min-h-[36px] items-center justify-center gap-2 border',
    nextButtonText: 'body-medium text-dark200_light800',
};
```

**Pagination.test.tsx** // файл із тестовими сценаріями для компонента «Pagination»

```
// Імпортування необхідних модулів
import '@testing-library/jest-dom';
import { fireEvent, render, screen } from '@testing-library/react';
import { useRouter, useSearchParams } from 'next/navigation';
import { Mock, vi } from 'vitest';
import { Pagination } from './Pagination';
import { PaginationProps } from './Pagination.interface';
// Імітація хуків useRouter і useSearchParams
vi.mock('next/navigation', () => ({
    useRouter: vi.fn(),
    useSearchParams: vi.fn(),
}));
const mockPush = vi.fn();
const mockSearchParams = new URLSearchParams('?page=1');
const testProps: PaginationProps = {
    pageNumber: 1,
    isNext: true,
};
describe('Компонент Pagination', () => {
    beforeEach(() => {
        vi.clearAllMocks();
        (useRouter as Mock).mockReturnValue({
                push: mockPush,
        });
        (useSearchParams as Mock).mockReturnValue(mockSearchParams);
    });
    test('відображається коректно з наданим номером сторінки', () => {
        render(<Pagination {...testProps} />);
        expect(screen.getByText('Попередня')).toBeInTheDocument();
        expect(screen.getByText('Наступна')).toBeInTheDocument();
        expect(
                screen.getByText(testProps.pageNumber.toString())
        ).toBeInTheDocument();
    });
    test('вимикає кнопку з текстом "Попередня", якщо номер сторінки = 1', () => {
        render(<Pagination {...testProps} />);
        expect(screen.getByText('Попередня').closest('button')).toBeDisabled();
    });
    test('вмикає кнопку з текстом "Попередня", якщо номер сторінки > 1', () => {
        render(<Pagination {...testProps} pageNumber={2} />);
        expect(screen.getByText('Попередня').closest('button')).not.toBeDisabled();
    });
    test('вимикає кнопку з текстом "Наступна", коли isNext дорівнює false', () => {
        render(<Pagination {...testProps} isNext={false} />);
        expect(screen.getByText('Наступна').closest('button')).toBeDisabled();
    });
```

```
        test('викликає функцію navigateHandler з правильними аргументами за натискання на кнопку з текстом "Попередня"', () => {
            render(<Pagination {...testProps} pageNumber={2} />);
            fireEvent.click(screen.getByText('Попередня'));
            expect(mockPush).toHaveBeenCalledWith(expect.stringContaining('page=1'));
        });
        test('викликає navigateHandler з правильними аргументами за натискання на кнопку з текстом "Наступна"', () => {
            render(<Pagination {...testProps} />);
            fireEvent.click(screen.getByText('Наступна'));
            expect(mockPush).toHaveBeenCalledWith(expect.stringContaining('page=2'));
        });
        test('відповідає знімку', () => {
            const { asFragment } = render(<Pagination {...testProps} />);
            expect(asFragment()).toMatchSnapshot();
        });
    });
```

**Filter.interface.ts** // файл із інтерфейсом властивостей компонента «Filter»

```
    // Описує об'єкт фільтра з ім'ям та значенням
    type Filter = {
        name: string; // Назва фільтра
        value: string; // Значення фільтра
    };
    export interface FilterProps {
        filters: Filter[]; // Масив фільтрів, кожен елемент якого є об'єктом типу Filter
        otherClasses?: string; // Додаткові класи для декорування елементів у компоненті
        containerClasses?: string; // Класи контейнера для декорування основного блоку компонента
    }
```

**Filter.tsx** // файл із реалізацією компонента фільтрації

```
    'use client';
    // Імпортування необхідних модулів
    import {
        Select,
        SelectContent,
        SelectGroup,
        SelectItem,
        SelectTrigger,
        SelectValue,
    } from '@/components/ui/select';
    import { setURLQuery } from '@/lib/utils';
    import { AppRouterInstance } from 'next/dist/shared/lib/app-router-context.shared-runtime';
    import {
        ReadonlyURLSearchParams,
        useRouter,
        useSearchParams,
    } from 'next/navigation';
    import { FilterProps } from './Filter.interface';
    import { filterStyled } from './Filter.styles';
```

```
// Функція для оновлення параметрів запиту URL
const updateQueryParams = (
    queryParameters: ReadonlyURLSearchParams,
    routeHandler: AppRouterInstance,
    value: string
): void => {
    // Формування нового URL з оновленими параметрами запиту
    const refreshedUrl = setURLQuery({
        parameters: queryParameters.toString(),
        property: 'filter',
        data: value,
    });
    // Переходить до нового URL без прокрутки сторінки
    routeHandler.push(refreshedUrl, { scroll: false });
};
export const Filter = (props: FilterProps): React.JSX.Element => {
    const queryParameters = useSearchParams();
    const routeHandler = useRouter();
    const queryFilter = queryParameters.get('filter');
    return (
        // Основний контейнер компонента зі стилями та додатковими класами
        <div className={`${filterStyled.body} ${props.containerClasses}`}>
            <Select
                // Оновлення параметрів URL при зміні значення вибору
                onValueChange={value =>
                    updateQueryParams(queryParameters, routeHandler, value)
                }
                // Встановлення значення за замовчуванням з параметра "filter"
або undefined
                defaultValue={queryFilter || undefined}
            >
                <SelectTrigger
                    className={`${props.otherClasses}
${filterStyled.selectTrigger}`}
                >
                    <div className={filterStyled.selectTriggerBody}>
                        \{/* Плейсхолдер для вибору фільтра */}
                        <SelectValue placeholder="Обери Фільтр" />
                    </div>
                </SelectTrigger>
                <SelectContent className={filterStyled.selectContent}>
                    <SelectGroup>
                        {/* Відображення елементів вибору фільтрів з
властивостей компонента */}
                        {props.filters.map(item => (
                            <SelectItem
                                key={item.value}
                                value={item.value}
                                className={filterStyled.selectItem}
                            >
                                {item.name}
```

```
                                        </SelectItem>
                                    ))}
                                </SelectGroup>
                            </SelectContent>
                        </Select>
                    </div>
                );
            };
```

**Filter.styles.ts //** файл зі стилями компонента «Filter»

```
            // Імпортування необхідних модулів
            import { Dynamic } from '@/types';
            type StyledKey =
                | 'body'
                | 'selectTrigger'
                | 'selectTriggerBody'
                | 'selectContent'
                | 'selectItem';
            export const filterStyled: Dynamic<StyledKey, string> = {
                body: 'relative',
                selectTrigger:
                    'body-regular light-border background-light800_dark300 text-dark500_light700
border px-5 py-2.5',
                selectTriggerBody: 'line-clamp-1 flex-1 text-left',
                selectContent:
                    'text-dark500_light700 small-regular border-none bg-light-900 dark:bg-dark-300',
                selectItem: 'cursor-pointer focus:bg-light-800 dark:focus:bg-dark-400',
            };
```

**Filter.test.tsx //** файл із тестовими сценаріями для компонента «Filter»

```
            // Імпортування необхідних модулів
            import '@testing-library/jest-dom';
            import { fireEvent, render, screen } from '@testing-library/react';
            import { useSearchParams } from 'next/navigation';
            import { useRouter } from 'next/router';
            import { Mock, vi } from 'vitest';
            import { Filter } from './Filter';
            import { FilterProps } from './Filter.interface';
            // Імітація хуків useRouter і useSearchParams
            vi.mock('next/navigation', () => ({
                useRouter: vi.fn(),
                useSearchParams: vi.fn(),
            }));
            // Імітація даних для тестових сценаріїв
            const mockFilters = [
                { value: 'all', name: 'Всі' },
                { value: 'active', name: 'Активні' },
                { value: 'completed', name: 'Завершені' },
            ];
            const mockProps: FilterProps = {
                filters: mockFilters,
```

```
        containerClasses: 'custom-container',
        otherClasses: 'custom-trigger',
    };
    describe('Компонент Filter', () => {
        let pushMock: ReturnType<typeof vi.fn>;
        let searchParamsMock: URLSearchParams;
        beforeEach(() => {
            pushMock = vi.fn();
            searchParamsMock = new URLSearchParams('?filter=active');
            (useRouter as Mock).mockReturnValue({
                    push: pushMock,
            });
            (useSearchParams as Mock).mockReturnValue(searchParamsMock);
        });
        afterEach(() => {
            vi.clearAllMocks();
        });
        test('відображається коректно зі вказаними фільтрами', () => {
            render(<Filter {...mockProps} />);
            expect(screen.getByPlaceholderText('Обери Фільтр')).toBeInTheDocument();
            mockFilters.forEach(filter => {
                    expect(screen.getByText(filter.name)).toBeInTheDocument();
            });
        });
        test('встановлює значення за замовчуванням згідно з параметром URL-запиту', ()
=> {
            render(<Filter {...mockProps} />);
            expect(screen.getByText('Активні')).toBeInTheDocument();
        });
        test('оновлює параметри запиту з правильними аргументами за зміни значення', ()
=> {
            render(<Filter {...mockProps} />);
            fireEvent.click(screen.getByPlaceholderText('Обери Фільтр'));
            fireEvent.click(screen.getByText('Завершені'));
            expect(pushMock).toHaveBeenCalledWith(
                    expect.stringContaining('filter=completed'),
                    { scroll: false }
            );
        });
        test('застосовує передані класи контейнера та тригера', () => {
            render(<Filter {...mockProps} />);
            const container = screen.getByRole('combobox').parentElement;
            expect(container).toHaveClass('custom-container');
            const trigger = screen.getByPlaceholderText('Обери Фільтр').parentElement;
            expect(trigger).toHaveClass('custom-trigger');
        });
    });
```

**RenderTag.interface.ts** // файл із інтерфейсом властивостей компонента «RenderTag»

```
export interface RenderTagProps {
```

```
    _id: number | string; // ідентифікатор тегу
    name: string; // назва тегу
    totalQuestions?: number; // загальна кількість питань, пов'язаних із тегом,
    showCount?: boolean; // прапорець, що визначає, чи відображати кількість питань
}
```

**RenderTag.tsx** // файл із реалізацією компонента для рендерингу тегу

```
// Імпортування необхідних модулів
import { Badge } from '@/components/ui/badge';
import Link from 'next/link';
import { RenderTagProps } from './RenderTag.interface';
import { renderTagStyled } from './RenderTag.styles';
export const RenderTag = (props: RenderTagProps): React.JSX.Element => (
    // Використання компонента Link для створення посилання на тег з переданим
ідентифікатором
    <Link href={`/tags/${props._id}`} className={renderTagStyled.link}>
        {/* Використання компонента Badge для відображення імені тегу */}
        <Badge className={renderTagStyled.nameBadge}>{props.name}</Badge>

        {/* Перевірка умови, якщо лічильник є істинним, відображається кількість
питань */}
        {props.showCount && (
            <p className={renderTagStyled.showCountParagraph}>
                {props.totalQuestions}
            </p>
        )}
    </Link>
);
```

**RenderTag.styles.ts** // файл зі стилями компонента «RenderTag»

```
// Імпортування необхідних модулів
import { Dynamic } from '@/types';
type StyledKey = 'link' | 'nameBadge' | 'showCountParagraph';
export const renderTagStyled: Dynamic<StyledKey, string> = {
    link: 'flex justify-between gap-2',
    nameBadge:
        'subtle-medium background-light800_dark300 text-light400_light500 rounded-md
border-none px-4 py-2 uppercase',
    showCountParagraph: 'small-medium text-dark500_light700',
};
```

**RenderTag.test.tsx** // файл із тестовими сценаріями для компонента «RenderTag»

```
// Імпортування необхідних модулів
import '@testing-library/jest-dom';
import { render, screen } from '@testing-library/react';
import { RenderTag } from './RenderTag';
import { RenderTagProps } from './RenderTag.interface';
import { renderTagStyled } from './RenderTag.styles';
describe('RenderTag component', () => {
    const testProps: RenderTagProps = {
        _id: '123',
```

```
            name: 'JavaScript',
            showCount: true,
            totalQuestions: 42,
        };
        test("коректно відображається з обов'язковими властивостями", () => {
            render(<RenderTag {...testProps} />);
            const linkElement = screen.getByRole('link');
            expect(linkElement).toHaveAttribute('href', '/tags/123');
            const badgeElement = screen.getByText('JavaScript');
            expect(badgeElement).toBeInTheDocument();
        });
        test('відображає загальну кількість питань, коли лічильник дорівнює true', () => {
            render(<RenderTag {...testProps} />);
            const countElement = screen.getByText('42');
            expect(countElement).toBeInTheDocument();
        });
        test('не відображає загальну кількість питань, коли лічильник дорівнює false', () =>
{
            const propsWithoutCount: RenderTagProps = {
                    ...testProps,
                    showCount: false,
            };
            render(<RenderTag {...propsWithoutCount} />);
            const countElement = screen.queryByText('42');
            expect(countElement).not.toBeInTheDocument();
        });
        test('посилання має правильний клас', () => {
            render(<RenderTag {...testProps} />);
            const linkElement = screen.getByRole('link');
            expect(linkElement).toHaveClass(renderTagStyled.link);
        });
        test('значок має правильний клас', () => {
            render(<RenderTag {...testProps} />);
            const badgeElement = screen.getByText('JavaScript');
            expect(badgeElement).toHaveClass(renderTagStyled.nameBadge);
        });
        test('абзац має правильний клас, коли лічильник дорівнює true', () => {
            render(<RenderTag {...testProps} />);
            const countElement = screen.getByText('42');
            expect(countElement).toHaveClass(renderTagStyled.showCountParagraph);
        });
    });
```

**NoResult.interface.ts //** файл із інтерфейсом властивостей компонента «NoResult», який буде відображений користувачам, якщо результати пошуку відсутні.

```
    export interface NoResultProps {
        title: string; // Заголовок, який відображається за відсутності результатів
        description: string; // Опис, який відображається за відсутності результатів
        link: string; // Посилання на іншу сторінку
        linkTitle: string; // Текст посилання
    }
```

**NoResult.tsx //** файл із реалізацією компонента «NoResult»

```
// Імпортування модулів
import Image from 'next/image';
import Link from 'next/link';
import { Button } from '../../ui/button';
import { NoResultProps } from './NoResult.interface';
import { noResultComponentStyled } from './NoResult.styles';
export const NoResult = (props: NoResultProps): React.JSX.Element => (
    <div className={noResultComponentStyled.body}>
        {/* Зображення для темної теми */}
        <Image
                classNameе={noResultComponentStyled.darkThemeImage}
                width={270}
                height={200}
                src="/assets/images/light-illustration.png"
                alt="ілюстрація відсутності результатів"
        />
        {/* Зображення для світлої теми */}
        <Image
                classNameе={noResultComponentStyled.lightThemeImage}
                width={270}
                height={200}
                src="/assets/images/dark-illustration.png"
                alt="ілюстрація відсутності результатів"
        />
        {/* Заголовок з повідомленням про відсутність результатів */}
        <h2 classNameе={noResultComponentStyled.title}>{props.title}</h2>
        {/* Опис з повідомленням про відсутність результатів */}
        <p classNameе={noResultComponentStyled.description}>{props.description}</p>
        {/* Посилання на іншу сторінку з кнопкою */}
        <Link href={props.link}>
                <Button classNameе={noResultComponentStyled.link}>
                        {props.linkTitle}
                </Button>
        </Link>
    </div>
);
```

**NoResult.styles.tsx //** файл зі стилями компонента «NoResult»

```
// Імпортування модулів
import { Dynamic } from '@/types';
type StyledKey =
    | 'body'
    | 'darkThemeImage'
    | 'lightThemeImage'
    | 'title'
    | 'description'
    | 'link';
export const noResultComponentStyled: Dynamic<StyledKey, string> = {
    body: 'mt-10 flex w-full flex-col items-center justify-center',
```

```
    darkThemeImage: 'block object-contain dark:hidden',
    lightThemeImage: 'hidden object-contain dark:flex',
    title: 'h2-bold text-dark200_light900 mt-8',
    description: 'body-regular text-dark500_light700 my-3.5 max-w-md text-center',
    link: 'paragraph-medium mt-5 min-h-[46px] rounded-lg bg-primary-500 px-4 py-3 text-light-900 hover:bg-primary-500 dark:bg-primary-500 dark:text-light-900',
};
```

**NoResult.test.tsx** // файл із тестовими сценаріями для компонента «NoResult»

```
// Імпортування модулів
import { render, screen } from '@testing-library/react';
import { describe, expect, it } from 'vitest';
import NoResult from './NoResult';
import { NoResultProps } from './NoResult.interface';
describe('NoResult Компонент', () => {
    const testProps: NoResultProps = {
        title: 'Немає питань для відображення',
        description:
            'Не будьмо мовчунами! 🚀 Запитай щось та розпочни обговорення. Твоє питання може стати чимось важливим, чого навчаться інші. Долучайся! 💡',
        link: '/ask-question',
        linkTitle: 'Запитай',
    };
    it('коректно рендериться', () => {
        render(<NoResult {...testProps} />);
    });
    it('рендерить заголовок правильно', () => {
        render(<NoResult {...testProps} />);
        const titleElement = screen.getByText(testProps.title);
        expect(titleElement).toBeInTheDocument();
    });
    it('рендерить опис правильно', () => {
        render(<NoResult {...testProps} />);
        const descriptionElement = screen.getByText(testProps.description);
        expect(descriptionElement).toBeInTheDocument();
    });
    it('рендерить посилання правильно', () => {
        render(<NoResult {...testProps} />);
        const linkElement = screen.getByRole('link');
        expect(linkElement).toBeInTheDocument();
        expect(linkElement.getAttribute('href')).toBe(testProps.link);
    });
});
```

**utils.ts** // файл із основними утилітами проєкту

```
// Імпортування необхідних модулів
import { TIME_INTERVALS } from '@/constants';
import { Nullable } from '@/types';
import { ClassArray, clsx } from 'clsx';
import qs, { ParsedQuery } from 'query-string';
import { twMerge } from 'tailwind-merge';
```

```
// Визначення типу для дільників чисел з суфіксом
type Division = {
    divisor: number;
    suffix: string;
};
// Інтерфейс для визначення параметрів запиту
interface QueryParameter {
    parameters: string;
    property: string;
    data: Nullable<string>;
}
// Інтерфейс для параметрів видалення з URL запиту
interface DeleteURLQueryParameter {
    parameters: string;
    propertiesToDelete: string[];
}
// Масив дільників для форматування великих чисел із суфіксами
const DIVISIONS: Division[] = [
    { divisor: 1000000, suffix: 'M' },
    { divisor: 1000, suffix: 'K' },
    { divisor: 1, suffix: '' },
];
// Функція для об’єднання класів за допомогою
export const mergeClassNames = (...classNames: ClassArray): string =>
    twMerge(clsx(classNames));
// Функція для форматування числа з відповідним суфіксом (K, M тощо)
export const formatNumberWithSuffix = (initialNumber: number): string => {
    for (const { divisor, suffix } of DIVISIONS)
        if (initialNumber >= divisor) {
            const formattedNumber = initialNumber / divisor;

            return `${
                divisor === 1 || formattedNumber % 1 === 0
                    ? Math.floor(formattedNumber)
                    : formattedNumber.toFixed(1)
            }${suffix}`;
        }

    return initialNumber.toString();
};
// Функція для отримання форматованої дати становлення учасником
export const getBecomeParticipantDate = (date: Date): string => {
    const fullMonthName = date.toLocaleString('uk-UA', { month: 'long' });
    const monthNameCapitalized =
        fullMonthName.charAt(0).toUpperCase() + fullMonthName.slice(1);
    const fullYear = date.getFullYear();

    return `${monthNameCapitalized} ${fullYear}`;
};
// Внутрішня функція для перетворення об'єкта URL на рядок за допомогою query-
string
```

```
const getQsStringifiedURLObject = (URL: ParsedQuery<string>): string =>
    qs.stringifyUrl(
        {
            url: window.location.pathname,
            query: URL,
        },
        {
            skipNull: true,
        }
    );
// Функція для встановлення параметру запиту в URL
export const setURLQuery = ({
    parameters,
    property,
    data,
}: QueryParameter): string => {
    const actualURL = qs.parse(parameters);
    actualURL[property] = data;
    return getQsStringifiedURLObject(actualURL);
};
// Функція для видалення вказаних властивостей з URL запиту
export const deletePropertiesFromURLQuery = ({
    parameters,
    propertiesToDelete,
}: DeleteURLQueryParameter): string => {
    const actualURL = qs.parse(parameters);
    propertiesToDelete.forEach((property) => delete actualURL[property]);
    return getQsStringifiedURLObject(actualURL);
};
```

**ДОДАТОК Б**

**ВІДГУК**

**керівника економічного розділу**

**на кваліфікаційну роботу бакалавра**

**на тему:**

**«Розробка інтерактивної Q&A платформи для ІТ-спільноти з використанням Next.js»**

**студента групи 122-20-2 Потебенька Єгора Дмитровича**

**Керівник економічного розділу**

**доц. каф. ПЕП та ПУ, к.е.н** **Л.В. Касьяненко**

# ДОДАТОК В

# ДІАГРАМИ ДІЯЛЬНОСТІ КОРИСТУВАЧІВ У СИСТЕМІ

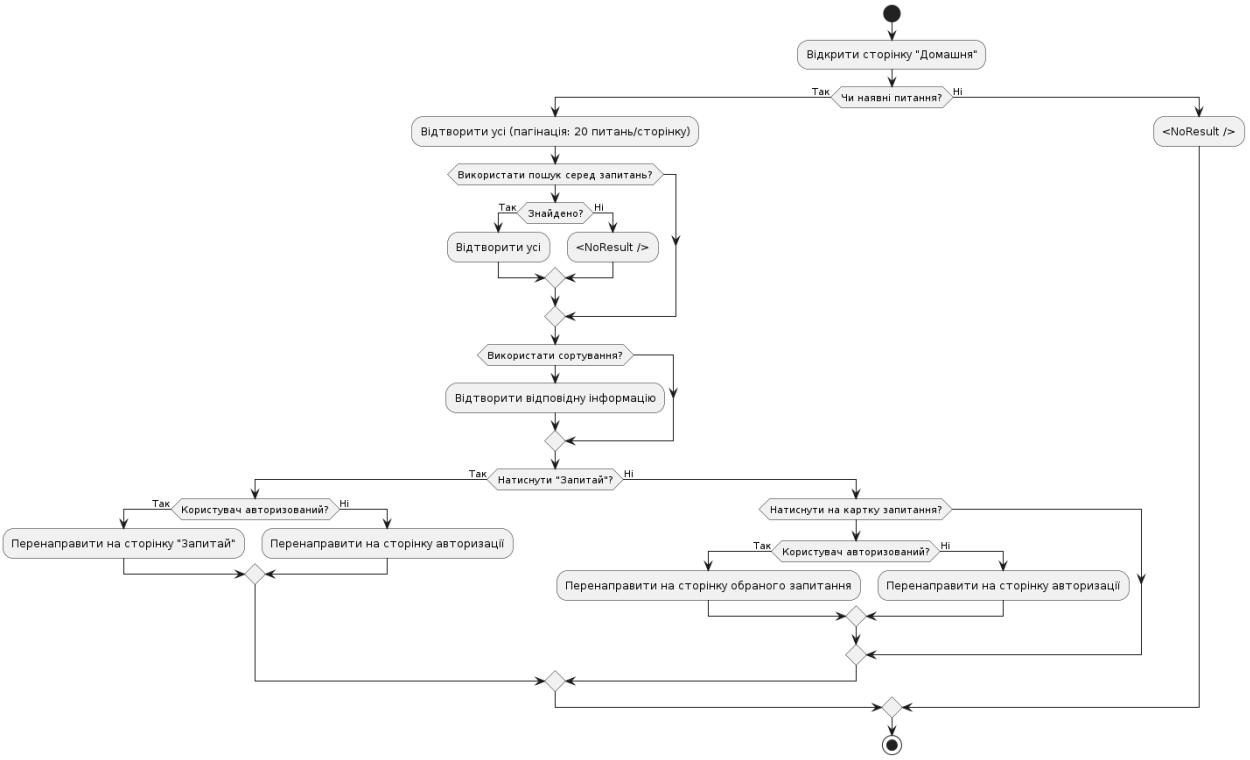


Рис. В.1. Діаграма діяльності на домашній сторінці

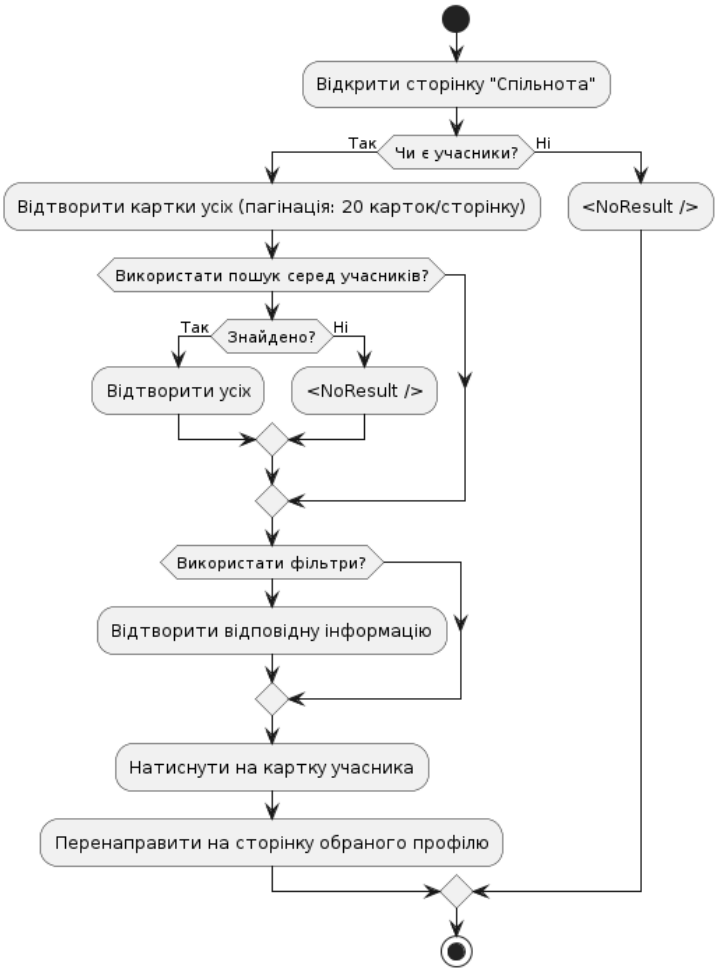


Рис. В.2. Діаграма діяльності на сторінці спільноти

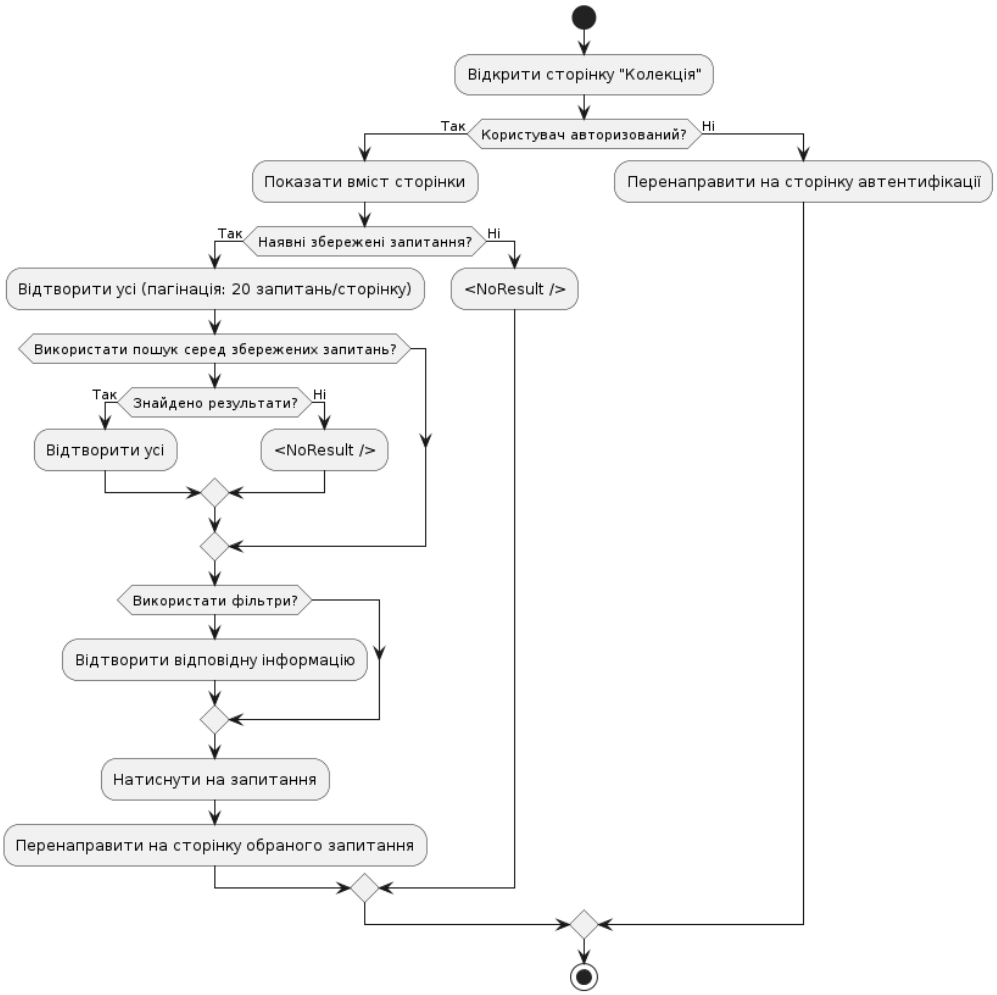


Рис. В.3. Діаграма діяльності на сторінці збережених запитань

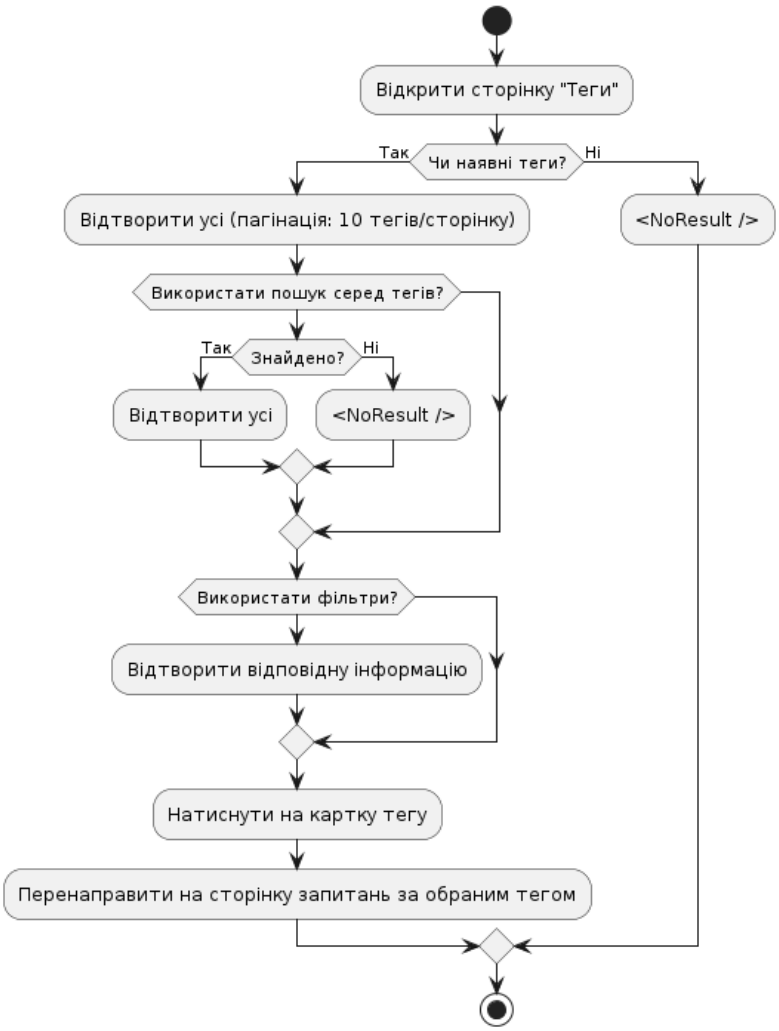


Рис. В.4. Діаграма діяльності на сторінці тегів

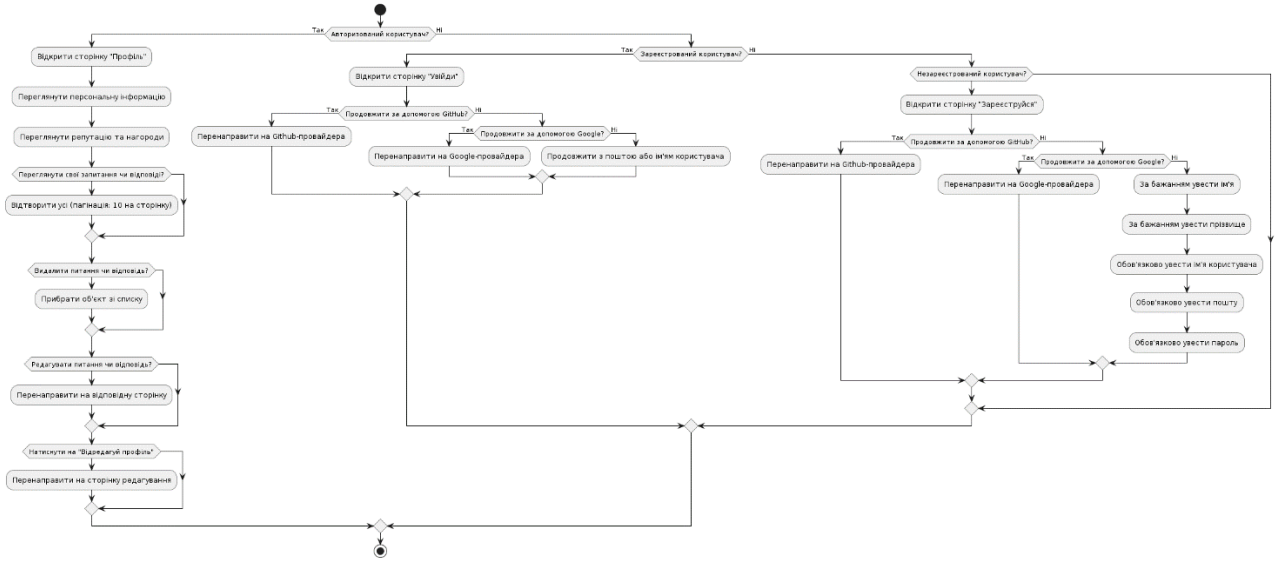


Рис. В.5. Діаграма діяльності на сторінках профілю, реєстрації та автентифікації

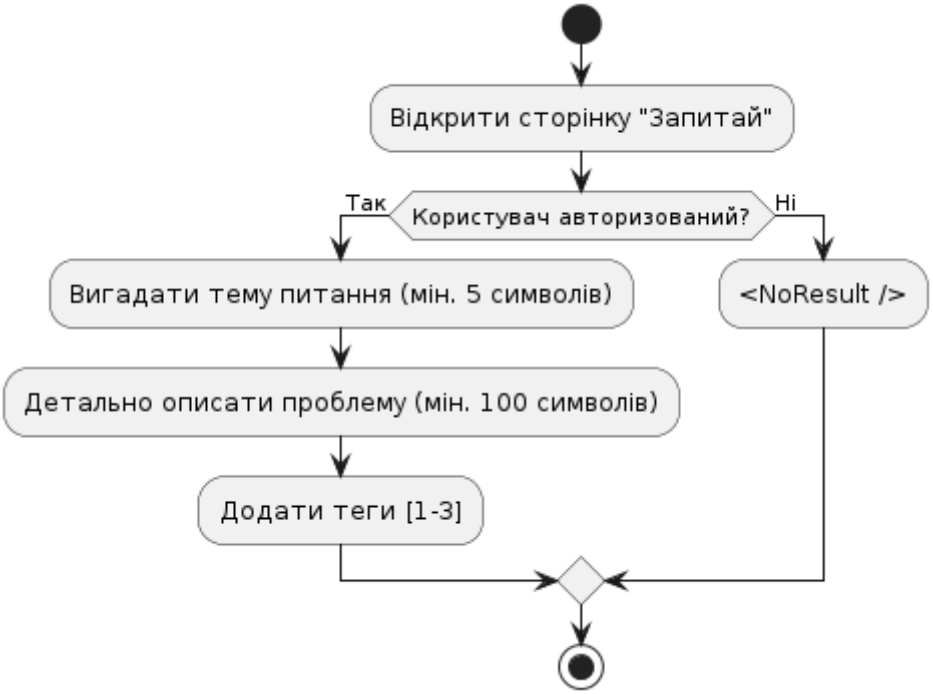


Рис. В.6. Діаграма діяльності на сторінці створення запитання

**ДОДАТОК Г**

# МОДЕЛІ ДОКУМЕНТІВ БАЗИ ДАНИХ

Таблиця Г.1

**Табличне представлення моделі документу «User»**

| Ключ | Опис | |
|---|---|---|
| | Тип даних | Призначення |
| id | String | Унікальний ID документа |
| clerkId | String | Унікальний ID облікового запису Clerk |
| name (опціонально) | String | Повне ім’я учасника |
| username | String | Унікальне ім’я користувача |
| email | String | Унікальна електронна адреса |
| bio (опціонально) | String | Додаткова інформація про учасника |
| picture | String | Аватар |
| location (опціонально) | String | Локація |
| portfolioWebsite (опціонально) | String | Посилання на веб-сайт із портфоліо |
| reputation | Int | Бали репутації (за замовчуванням – 0) |
| questions | Question[] | Масив посилань на поставлені запитання |
| answers | Answer[] | Масив посилань на надані відповіді |
| interactions | Interaction[] | Масив посилань на активність |
| joinedAt | DateTime | Дата й час реєстрації в системі |

Таблиця Г.2

**Табличне представлення моделі документу «Question»**

| Ключ | Опис | |
|---|---|---|
| | Тип даних | Призначення |
| id | String | Унікальний ID документа |
| title | String | Заголовок |
| content | String | Вміст |
| views | Int | Кількість переглядів (за замовчуванням – 0) |
| upvotes | Int | Кількість уподобань (за замовчуванням – 0) |
| downvotes | Int | Кількість голосів «проти» (за замовчуванням – 0) |
| author | User | Посилання на автора |
| answers | Answer[] | Масив посилань на надані відповіді |
| interactions | Interaction[] | Масив посилань на пов’язану активність |
| tags | Tag[] | Масив посилань на додані теги |
| createdAt | DateTime | Дата й час створення запису в БД |

Таблиця Г.3

**Табличне представлення моделі документу «Answer»**

| Ключ | Опис | |
|---|---|---|
| | Тип даних | Пояснення |
| id | String | Унікальний ID документа |
| content | String | HTML-розмітка вмісту |
| upvotes | Int | Кількість уподобань (за замовчуванням – 0) |
| downvotes | Int | Кількість голосів «проти» (за замовчуванням – 0) |
| author | User | Посилання на автора |
| question | Question | Посилання на відповідне запитання |
| interactions | Interaction[] | Масив посилань на пов’язану активність |
| createdAt | DateTime | Дата й час створення запису в БД |

Таблиця Г.4

**Табличне представлення моделі документу «Interaction»**

| Ключ | Опис | |
|---|---|---|
| | Тип даних | Призначення |
| id | String | Унікальний ID документа |
| action | String | Один із типів дії на платформі: створення запитання, перегляд запитання, надання відповіді, уподобання запитання/відповіді, голосування «проти» запитання/відповіді, збереження запитання в колекції |
| user | User | Посилання на відповідного користувача |
| question (опціонально) | Question | Посилання на відповідне запитання |
| answer (опціонально) | Answer | Посилання на пов’язану відповідь |
| createdAt | Date | Дата й час створення запису в БД |

Таблиця Г.5

**Табличне представлення моделі документу «Tag»**

| Ключ | Опис | |
|---|---|---|
| | Тип даних | Призначення |
| id | String | Унікальний ID документа |
| name | String | Унікальна назва тегу |
| questions | Question[] | Масив посилань на відповідні запитання |
| createdAt | DateTime | Дата й час створення запису в БД |

**ДОДАТОК Д**

# ПАРАМЕТРИ ВИКОРИСТАНИХ ТЕХНІЧНИХ ЗАСОБІВ

Таблиця Д.1

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ НОУТБУКУ «ASUS TUF GAMING 90NR02D1-M11380»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Процесор | AMD Ryzen™ 5 3550H із частотою 2,1 ГГц (2 МБ кеш-пам’яті, до 3,7 ГГц) |
| 2 | Загальний об’єм пам’яті | 8GB DDR4-2666 SO-DIMM |
| 3 | Відеокарта | NVIDIA GeForce GTX 1650 |
| 4 | Відеопам’ять | 4GB GDDR5 |
| 5 | Розмір дисплея | 15,6 дюймів |
| 6 | Роздільна здатність дисплея | FHD (1920 x 1080) 16:9 |
| 7 | Дисплей | Рівень IPS |
| 8 | Технологія багатоканальної пам’яті | 256GB M.2 NVMe™ PCIe® 3.0 SSD |
| 9 | Операційна система | Windows 10 Домашня |

Таблиця Д.2

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ МОНІТОРУ «SAMSUNG F24T350FHI»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Діагональ дисплея | 23.8" |
| 2 | Максимальна роздільна здатність дисплея | 1920x1080 (FullHD) |
| 3 | Частота оновлення | 75 Гц |
| 4 | Тип матриці | IPS |
| 5 | Час реакції матриці | 5 мс |
| 6 | Яскравість дисплея | 250 кд/м² |
| 7 | Контрастність дисплея | 1000:1 |
| 8 | Вертикальний / горизонтальний кути огляду | 178° / 178° |

Таблиця Д.3

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ СМАРТФОНУ «SAMSUNG GALAXY M33 5G»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Тип процесора | 8-ядерний |
| 2 | Тактова частота процесора | 2,4 ГГц, 2 ГГц |
| 3 | Оперативна пам’ять | 6 Гб |
| 4 | Внутрішня пам’ять | 128 Гб |
| 5 | Розмір дисплея | 6,6" |
| 6 | Роздільна здатність дисплея | 1080 x 2408 (FHD+) |
| 7 | Ємність акумулятора | 5000 мА/год |
| 8 | Операційна система | Android 14 |

Таблиця Д.4

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ СМАРТФОНУ «APPLE IPHONE 13»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Операційна система | iOS 15 |
| 2 | Процесор | Apple A15 Bionic |
| 3 | Оперативна пам’ять | 4 Гб |
| 4 | Внутрішня пам’ять | 256 Гб |
| 5 | Діагональ дисплея | 6,1" |
| 6 | Роздільна здатність дисплея | 1170x2532 |
| 7 | Щільність розміщення пікселів на один дюйм | 460 |
| 8 | Частота оновлення екрану | 60 Гц |
| 9 | Ємність акумулятора | 3240 мА/год |

Таблиця Д.5

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ МИШІ «ASUS TUF GAMING M3»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Сенсор | PAW3325 |
| 2 | Максимальна роздільна здатність | 7000 DPI |
| 3 | Максимальна швидкість | 100 IPS |
| 4 | Максимальне прискорення | 20 G |
| 5 | Частота звіту USB | 1000 Гц |

Таблиця Д.6

**ТЕХНІЧНІ ХАРАКТЕРИСТИКИ КЛАВІАТУРИ «TECURS KB510»**

| № п/п | Характеристика | |
|---|---|---|
| | Назва | Значення |
| 1 | Тип клавіатури | Механічна |
| 2 | Тип акумулятора | Літій-полімерний |
| 3 | Ємність акумулятора | 2500 мА/год |

**ДОДАТОК Е**

# ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

| **Ім’я файлу** | **Опис** |
|---|---|
| Пояснювальні документи | |
| КваліфікаційнаРобота_Потебенько.docx | Пояснювальна записка до кваліфікаційної роботи. Документ Word. |
| КваліфікаційнаРобота_Потебенько.pdf | Пояснювальна записка до кваліфікаційної роботи у форматі PDF. |
| Програма | |
| ITHelps.zip | Архів. Містить оптимізовану версію з відкомпільованим кодом програми. |
| Презентація | |
| Презентація_Потебенько.pptx | Презентація кваліфікаційної роботи. |