МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ОНЛАЙН-ПЛАТФОРМА ДЛЯ НАВЧАННЯ БАТЬКІВ З ПСИХОЕМОЦІЙНОГО РОЗВИТКУ ДІТЕЙ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувачка _____ Інеса ЛЕСНІКОВА

«___»_____ 2026 р.

Керівник д-р техн. наук, професор ___Юрій КОНДРАТЕНКО

«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем
_____ Євген СІДЕНКО
«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувачки

Леснікової Інеси Вікторівни
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Онлайн-платформа для навчання батьків з психоемоційного розвитку дітей.
Керівник роботи: Кондратенко Юрій Пантелійович, професор кафедри ІІС, доктор технічних наук, професор.
Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.
2. Строк представлення кваліфікаційної роботи «15» червня 2026 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: інтелектуальна онлайн-платформа для навчання батьків з психоемоційного розвитку дітей; методи персоналізації навчального контенту; результати тестування користувачів; рекомендаційна система для формування індивідуальної освітньої траєкторії; технології веброзробки Nuxt.js, Vue.js, Node.js, MySQL та Drizzle ORM.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану онлайн-платформ для навчання батьків у сфері психоемоційного розвитку дітей; огляд існуючих методів персоналізації навчання та інтелектуального аналізу результатів тестування; проєктування архітектури онлайн-платформи та структури бази даних; розробка користувацького інтерфейсу платформи; реалізація системи тестування, рекомендацій та аналітики прогресу користувачів; обґрунтування вибору технологій веброзробки та архітектурного підходу SPA + SSR; реалізація серверної логіки взаємодії користувача з платформою.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Юрій КОНДРАТЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Інеса ЛЕСНІКОВА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «21» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Онлайн-платформа для навчання батьків з психоемоційного розвитку дітей.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	Виконано
2	Аналіз предметної області та постановка задачі	25.12.25	30.01.2026	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд останніх публікацій та аналіз існуючих аналогів	31.01.2026	01.03.2026	Виконано
4	Огляд методів та програмно-технічних засобів вирішення поставленої задачі	02.03.2026	01.04.2026	Виконано
5	Проєктування логіки та інтелектуальної складової системи	02.04.2026	16.04.2026	Виконано
6	Реалізація розробки онлайн-платформи	16.04.2026	24.05.2026	Виконано
7	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	Виконано
8	Корегування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	Виконано
9	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	Виконано
10	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	Виконано
11	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	Виконано

Керівник роботи _____ Юрій КОНДРАТЕНКО
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Інеса ЛЕСНІКОВА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«29» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувачки групи 402 ЧНУ ім. Петра Могили

Леснікової Інеси Вікторівни

на тему: **«ОНЛАЙН-ПЛАТФОРМА ДЛЯ НАВЧАННЯ БАТЬКІВ З ПСИХОЕМОЦІЙНОГО РОЗВИТКУ ДІТЕЙ»**

Актуальність роботи заключається у розробці інтелектуальної платформи, здатної гнучко адаптуватися до специфічних потреб кожної сім'ї, що є нагальною необхідністю для ефективного формування здорового психоемоційного стану дітей.

Метою роботи є розробка інтелектуальної онлайн-платформи, що забезпечує оволодіння сучасними методами виховання, вікової психології та емоційної підтримки дітей через індивідуальний підхід.

Об'єктом роботи є процеси інформаційної та методичної підтримки батьків стосовно психоемоційного розвитку та виховання дітей.

Предметом роботи є методи та програмні засоби створення інтелектуальної онлайн-платформи для навчання батьків.

У процесі дослідження використано методи системного аналізу, порівняльного аналізу наявних рішень, об'єктно-орієнтованого проєктування та штучний інтелект для персональних рекомендацій на основі тестування.

Створена платформа вирізняється високим рівнем індивідуалізації навчання. Результати роботи можуть бути використані батьками для самостійного підвищення виховного потенціалу.

Загальний обсяг роботи – 83 сторінки. Кваліфікаційна робота містить 38 рисунків, 9 таблиць і 34 джерел посилання.

Ключові слова: онлайн-платформа, психоемоційний розвиток дитини, свідоме батьківство, емоційний інтелект, інтелектуальне тестування, персоналізація навчання.

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Lesnikova Inesa

«ONLINE PLATFORM FOR EDUCATING PARENTS ON CHILDREN'S PSYCHO-EMOTIONAL DEVELOPMENT»

The relevance of this work lies in the development of an intelligent platform capable of flexibly adapting to the specific needs of each family, which is essential for effectively fostering a healthy psycho-emotional state in children.

The aim of the work is to develop an intelligent online platform that enables parents to master modern methods of child-rearing, age-appropriate psychology, and emotional support for children through an individualized approach.

The object of this study is the processes of providing informational and methodological support to parents regarding children's psycho-emotional development and upbringing.

The subject of this study is the methods and software tools for creating an intelligent online platform for educating parents.

The research utilized methods of systems analysis, comparative analysis of existing solutions, object-oriented design, and artificial intelligence to process the test results.

The created platform is distinguished by a high level of learning personalization. The results of this work can be used by parents to independently enhance their parenting potential.

The overall scope of the work is 83 pages. Thesis contains 38 figures, 9 tables and 34 references in it.

Keywords: online platform, child's psycho-emotional development, conscious parenting, emotional intelligence, intellectual testing, personalized learning.

ЗМІСТ

ВСТУП

У сучасному світі, де швидко розвиваються цифрові технології і зростає увага до психічного здоров’я, питання психоемоційного розвитку дітей стає особливо важливим. У ранньому віці закладаються основні емоційні реакції, комунікаційні навички, рівень довіри до навколишнього світу та здатність спілкуватися з іншими. Важливу роль у цьому процесі відіграють батьки, адже саме сімейне середовище визначає, як формується емоційний стан дитини, її самооцінка та психологічна стійкість.

Однак багато батьків стикаються з труднощами в вихованні дітей через брак перевіреної інформації, відсутність систематизованих знань про вікову психологію та обмежений доступ до сучасних методів емоційної підтримки. Існуючі інформаційні ресурси часто дають загальні поради, не враховуючи конкретні потреби кожної сім’ї, а більшість цифрових сервісів не поєднують навчання, тестування, аналітику та персоналізовані рекомендації. Це створює потребу в розробці розумних онлайн-платформ, які могли б запропонувати комплексний підхід до підтримки батьків.

Актуальність теми кваліфікаційної роботи полягає в тому, що потрібно створити сучасне вебсередовище. Це середовище має допомогти батькам отримувати структуровані знання про психоемоційний розвиток дітей, проходити тести для визначення їх актуальних потреб, аналізувати результати та отримувати персоналізовані рекомендації для подальшого навчання. Використання цифрових технологій у батьківській освіті покращує доступ до психологічної підтримки, сприяє усвідомленому батьківству та допомагає поліпшити емоційний клімат у сім'ї.

У ході дослідження були використані методи системного аналізу, структурного аналізу, декомпозиції, об’єктно-орієнтованого проєктування, а також сучасні підходи до веброзробки, зокрема технології Nuxt.js, Vue.js, Node.js, MySQL та Drizzle ORM.

Структура кваліфікаційної роботи складається з вступу, трьох розділів, висновків, списку джерел та додатків. У першому розділі аналізується предметна область і наявні рішення. У другому розділі описується проєктування архітектури системи, бази даних і логіки роботи платформи. У третьому розділі розглядається процес створення користувацького інтерфейсу, серверної частини та функціональних можливостей онлайн-платформи.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ОБ'ЄКТУ ПРОЄКТУВАННЯ

1.1 Основні поняття у сфері онлайн-навчання батьків

Психоемоційний розвиток – це процес формування емоційної сфери, соціальних зв'язків, прив'язаності, самосприйняття та базових комунікативних навичок [1]. Він включає:

- здатність розпізнавати та виражати свої емоції;
- формування міцного зв'язку з батьками;
- розвиток здатності розуміти почуття інших;
- формування уявлення про себе як особистість;
- навчання контролювати свої емоції та поведінку.

Електронне навчання – це система навчання, яка використовує інтернет для здобуття знань без необхідності фізичної присутності в навчальному закладі. Основні переваги такого навчання для дорослих:

- гнучкість та доступність: можливість навчатися будь-коли та будь-де;
- інтерактивність: використання різноманітних мультимедійних матеріалів (відео, аудіо, тести);
- самостійність: кожен навчається у власному темпі;
- практична спрямованість: акцент на здобутті навичок, які можна одразу застосувати в роботі.

Інтелектуальна платформа – це комплексне програмне або апаратно-програмне рішення, яке використовує штучний інтелект, машинне навчання, аналіз даних та автоматизацію для вирішення складних завдань, оптимізації процесів та прийняття рішень з мінімальним втручанням людини. Її ключові можливості включають:

- автоматизація та управління: створення хмарних інфраструктур, автоматизація IT-процесів;
- інтелектуальна обробка документів: автоматичне сканування, сортування та вилучення інформації з документів;

– аналітика та моніторинг: використання ШІ для аналізу контенту та його розумного редагування;

– безпека та відеомоніторинг: виявлення нетипових ситуацій у реальному часі;

– оптимізація ресурсів: ефективне управління виробничими процесами, наприклад, у сільському господарстві.

Щоб ефективно допомагати батькам у питаннях психоемоційного розвитку їхніх дітей, онлайн-платформа повинна мати певні унікальні риси, які визначають її функціонал:

– доступність у будь-який час та в зручному форматі: батьки часто мають дуже обмежений час;

– індивідуальний підхід завдяки інтелектуальному аналізу: кожна дитина та кожна сім'я унікальні;

– активне залучення та можливість самооцінки: ефективна підтримка вимагає активної участі батьків;

– емоційна підтримка та стабілізація: важлива не тільки інформація, але й допомага батькам з тим, щоб справлятися з їхньою тривогою.

1.2 Огляд останніх публікацій та аналіз існуючих аналогів

Не дивлячись на кількість існуючих рішень у сфері цифрової підтримки створення інтелектуальних онлайн-платформ для батьків з навчання психоемоційного розвитку дітей, більшість з них:

– не використовують комплексного підходу, що має у собі навчання, тестування та рекомендації;

– не враховують індивідуальності кожної дитини, потреби батьків;

– не мають змоги забезпечити адаптивний навчальний процес.

Проаналізувавши усі можливі аналоги можна виокремити: освітні платформи з курсами, психологічні сервіси зі консультаціями та мобільні додатки для розвитку дітей. Але всі вони мають недоліки:

– узагальнена інформація без персоналізації;
– відсутність тестувань для визначення потреб;
– відсутність аналізу результатів;
– можлива прив’язка до психологів;
– можлива відсутність навчальної структури (обумовлено виключним зв’язком зі спеціалістом);
– орієнтація на дітей без батьків;
– можлива відсутність емоційної підтримки батьків.

Враховуючи вище зазначені недоліки, було вирішено провести узагальнений порівняльний аналіз між платформами за ключовими характеристиками. Це допомагає не лише систематизувати, а й визначити рівень відповідності існуючих рішень сучасним вимогам.

Основними критеріями, що були відібрані для порівняння платформ сфери психоемоційного розвитку дітей є:

– структуризація контенту за курсами;
– наявність тестування для визначення потреб користувача;
– персоналізація;
– аналітика та відстеження прогресу;
– рекомендації;
– основний фокус платформи.

За допомогою обраних критеріїв виникає змога комплексно оцінити функціонал платформ та визначити їх переваги з недоліками з точки зору користувача. Результати порівняльного аналізу платформ наведено у табл.1.1.

Таблиця 1.1 – Порівняльний аналіз функцій існуючих онлайн-платформ для батьків

Платформа	Тип платформи	Курс	Тестування	Персоналізація	Аналітика	Рекомендації	Фокус
Coursera	Освітня платформа	+	-	-	-	-	Онлайн-курси
BetterHelp	Психологічна підтримка	-	-	+ (через спеціаліста)	-	+ (від спеціаліста)	Консультації
Bebbo	Додаток для батьків	-	-	+ (частково)	+ (трекінг розвитку)	+	Розвиток дитини
Baby Buddy	Освітній додаток	-	-	-	-	+	Підтримка батьків
The Wonder Weeks	Додаток розвитку	-	-	+	+	+	Поведінка дитини
BabyCenter	Контентна платформа	-	-	+ (базова)	-	+	Поради та статті
LittleLeaps	Додаток трекінгу	-	-	+	+	+	Дані + розвиток
Vroom	Освітній додаток	-	-	-	-	+	Щоденні поради

Проаналізовані платформи показують, що вони:

– створені для загальної інформації, але не цілеспрямованого навчання;

– не мають тестувань для визначення потреб користувача;

– запроваджена часткова або повністю відсутня персоналізація;

– відсутня повноцінна аналітика прогресу користувача.

Проте, можна виокремити освітню платформу Coursera, що пропонує структуровані якісні курси від провідних компаній та університетів. З переваг:

– модульність курсів, що забезпечує послідовне вивчення матеріалу;

– курси створені висококваліфікованими експертами галузі психології та педагогіки;

– інтерактивні елементи навчання у вигляді тестів після модулів, завдань за темою;

– формування практичних навичок виховання;

– сертифікат за проходження курсів.

На рис. 1.1 зображено вигляд платформи Coursera, зокрема сторінка курсу «Everyday Parenting: The ABCs of Child Rearing».

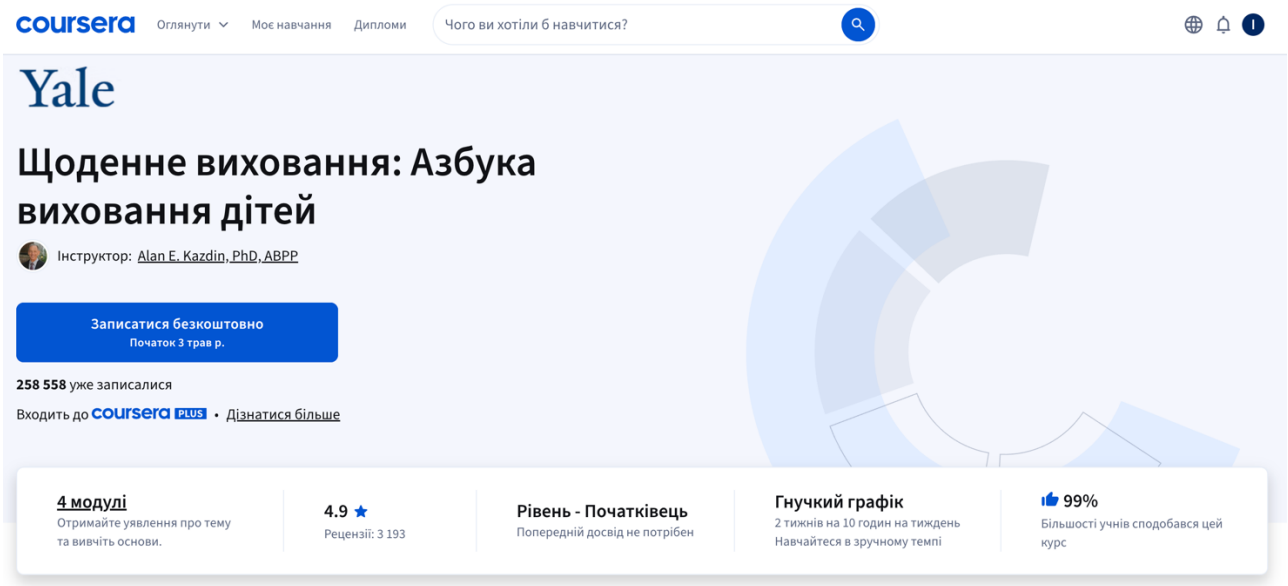

Рисунок 1.1 – Сторінка курсу на онлайн-платформі Coursera [2]

Також цікавим є додаток Bebbo завдяки тому, що він надає поради та інструменти для розвитку дитини. Наприклад:

– трекінг етапів розвитку (показує, якими навичками починає оволодівати дитина у певному віці, і дозволяє відмічати освоєний прогрес);

– щоденник розвитку дитини (можлива фіксація важливих подій, таких як перше слово, перші кроки, вікові кризиси дитини);

– відстеження здоров’я дитини;

– поради за віком дитини;

– нагадування про щеплення, медичні огляди, важливі етапи розвитку, тощо;

– активності для розвитку.

На рис. 1.2 зображено вигляд додатку Bebbo, зокрема головної сторінки та сторінки порад.

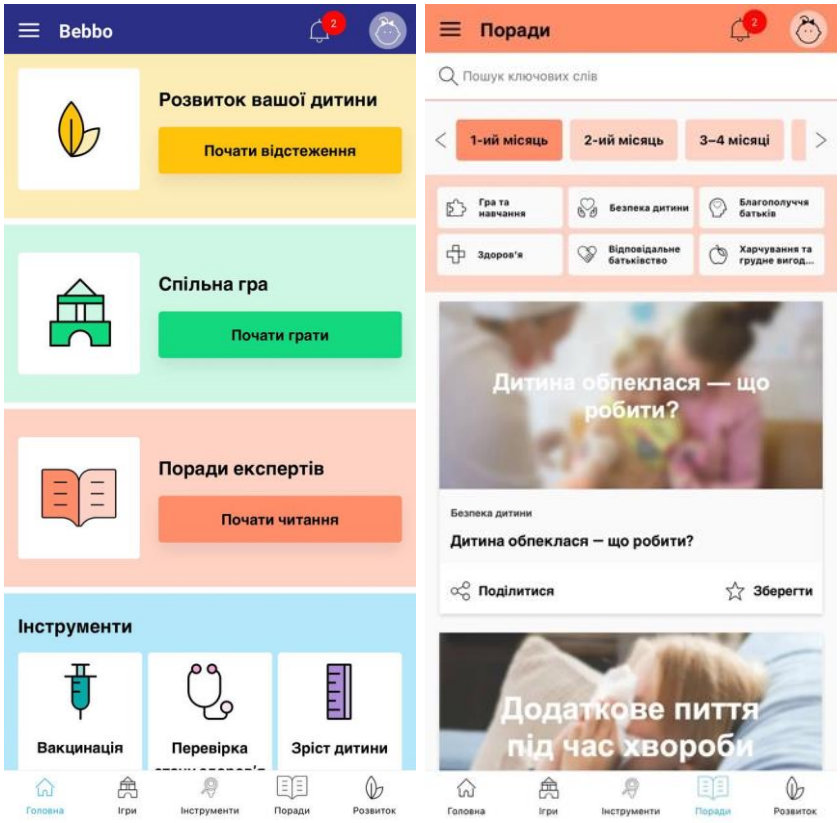

Рисунок 1.2 – Головна сторінка та сторінка порад застосунку Bebbo [3]

Не можна також не звернути уваги на додаток Vroom. Застосунок орієнтований на розвиток дитини за допомогою щоденної взаємодії з батьками, а також використання обґрунтованих підходів до раннього розвитку. Перевагами є:

– щоденна активність для розвитку дитини (пропонується понад 1000 маленьких вправ, які виконуються у звичайних будніх ситуаціях);

– формування навчання через звичайні життєві ситуації, що буде сприяти розвитку мислення та мовлення малюка;

– супроводження кожної активності поясненням того, які саме когнітивні здібності розвиваються у дитини під час процесу;

– персоналізовані поради відносно віку дитини;

Вигляд додатку Vroom продемонстровано нижче на рис. 1.3, зокрема сторінку для сьогоднішнього дня та сторінку прогресу.

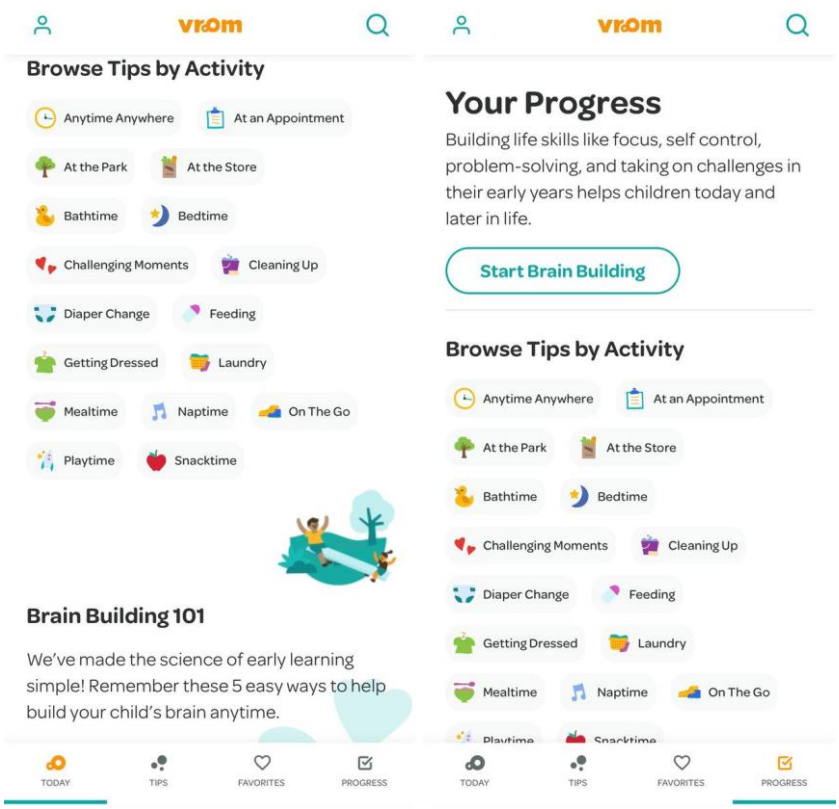

Рисунок 1.3 – Сторінка сьогоднішнього дня та сторінка прогресу застосунку Vroom [4]

Для легшого підбиття підсумку було створено SWOT-аналіз, що допомагає чітко відокремити головні переваги, недоліки, загрози та можливості. Нижче на рис. 1.4 продемонстровано результат цього аналізу.

Рисунок 1.4 – SWOT-аналіз

Після проведення порівняльного аналізу можна зробити висновок, що платформи лише частково покривають потреби батьків для навчання. Жодна платформа не забезпечує повної функціональності, зокрема структуризації, тестувань, аналітики, персоналізації та рекомендацій. Має свої недоліки та загрози. Виявлені обмеження існуючих рішень підтверджують актуальність розробки платформи, що інтегруватиме всі необхідні елементи для комфортного та продуктивного навчання батьків психоемоційного розвитку дітей.

1.3 Огляд методів та програмно-технічних засобів вирішення задач

Для реалізації цього проєкту було обрано передову веб-архітектуру, яка інтегрує клієнтську та серверну складові в єдиний робочий простір. В основі системи лежить фреймворк Nuxt.js, що базується на Vue.js, який забезпечує ефективну розробку сучасних вебзастосунків із підтримкою серверного рендерингу (SSR) та односторінкової архітектури (SPA). Для забезпечення високої продуктивності та швидкої обробки запитів серверна частина розроблена з використанням середовища виконання Node. Взаємодія з базою даних здійснюється через Drizzle ORM, що гарантує типізацію та підвищує безпеку доступу до даних. Як система керування базами даних обрано MySQL, що забезпечує надійне зберігання, структурованість та ефективне опрацювання інформації, що є ключовим для масштабованості платформи.

Nuxt.js – це потужний інструмент, побудований на базі Vue.js, який значно спрощує та прискорює створення вебдодатків. Він надає готові модулі, що містять різноманітний функціонал, який можна легко комбінувати для швидкого збирання проекту. Завдяки Nuxt.js розробникам не доведеться винаходити велосипед для типових завдань. Крім того, фреймворк забезпечує чітку структуру та стандарти оформлення, що полегшує командну роботу та підвищує її ефективність [5-7].

Vue.js – це сучасний JavaScript-фреймворк, спеціально розроблений для створення динамічних інтерфейсів користувача. Він вважається одним з наймолодших, але при цьому ввібрав у себе найкращі риси своїх конкурентів, таких

як Angular та React. Хоча ядро Vue.js не перевантажене великою кількістю вбудованих бібліотек, як це буває у більш «старих» проектів, його головна сила полягає в легкій інтеграції [8].

Drizzle – це сучасний об'єктно-реляційний маппер (Object-Relational Mapping, ORM) для Node.js і TypeScript. Простіше кажучи, Drizzle — це інструмент, що дозволяє працювати з реляційними (PostgreSQL, MySQL, SQL Server, SQLite) та нереляційними (MongoDB) базами даних за допомогою JavaScript або TypeScript без використання SQL (хоча така можливість є). Drizzle дозволяє створювати SQL-запити з перевіркою типів на основі опису схеми бази даних і не має накладних витрат, притаманних багатьом альтернативним інструментам для роботи з БД. Нижче у табл. 1.2 наведено переваги Drizzle порівняно з іншими ORM [9].

Таблиця 1.2 – Переваги Drizzle порівняно з іншими ORM

Інструменти	Мінуси у роботі	Рішення Drizzle ORM
Sequelize	Слабка підтримка TypeScript, застаріла архітектура, дублювання описів схем і типів	У Drizzle типи формуються на основі схеми, а опис структури – у єдиному місці
TypeORM	Багато прихованої поведінки, нестабільна типізація, громіздкий API	У Drizzle все зрозуміло: схема – це код, запити – зрозумілі, поведінка – передбачувана
Prisma	Високий поріг входу, генерація клієнта, великий розмір проекту	Drizzle не потребує генерації, підключається за допомогою двох рядків, не збільшує розмір пакета
Knex	Немає вбудованої типізації, всю логіку потрібно писати вручну	Drizzle надає типи «з коробки», але залишає контроль над структурою та логікою

Drizzle підтримує більшість популярних реляційних баз даних (як локальних, так і хмарних/serverless):

– PostgreSQL;
– MySQL;
– SQLite.

PostgreSQL – це вільна і відкрита система керування базами даних (СУБД) з відкритим вихідним кодом. Є однією з найпопулярніших СУБД, тому її використовують тисячі організацій по всьому світу. Ця система ґрунтується на некомерційній СУБД Postgres, яка почалася як відкритий проект у Каліфорнійському університеті в Берклі. Ініційована 1986 року Майклом Стоунбрейкером, лідером проєкту Ingres, вона була названа «Post Ingres». Згодом вона вийшла за межі університету і стала відомою як PostgreSQL, розвиваючись і вдосконалюючись до теперішнього часу.

SQLite – це зручне і компактне рішення для зберігання та управління даними. Особливо вона зручна в мобільних додатках, де немає необхідності в складних серверах. SQLite дає змогу розробникам ефективно керувати даними, водночас скорочуючи та забезпечуючи їхню цілісність, зберігаючи водночас простоту використання та низькі накладні витрати.

MySQL – це вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [10].

Node.js – це однопоточне кросплатформове середовище виконання з відкритим вихідним кодом і бібліотека, яка використовується для запуску вебдодатків, написаних на JavaScript, поза браузером клієнта. Тобто це програмне

середовище, яке дозволяє запускати програми, написані мовою JavaScript, поза браузером. Історично програми, написані на JavaScript, на відміну від інших мов програмування, можна було запустити лише у браузерах, які мали спеціальний вбудований движок виконання коду цієї мови. Поза браузером JavaScript, можна сказати, не працював. Основний принцип Node.js – модель асинхронного програмування. Це означає, що розробники можуть створювати ефективні та чуйні додатки, спроможні одночасно обробляти безліч запитів. Node.js також відомий своєю легкістю та швидкістю роботи, що дозволяє ефективно використовувати ресурси сервера. Node.js став однією з найпопулярніших платформ для створення веб-серверів, API, мікросервісів та багато іншого [11].

Односторінковий додаток (SPA) – це тип вебдодатку, який завантажується як одна HTML-сторінка. На відміну від традиційних сайтів, де кожен запит призводить до завантаження нової сторінки, в SPA весь контент динамічно оновлюється за допомогою JavaScript. Це створює враження безперебійної роботи, схожої на десктопні програми. Вся логіка додатку, інтерфейс користувача та взаємодія з сервером реалізовані у файлах JavaScript, які завантажуються один раз.

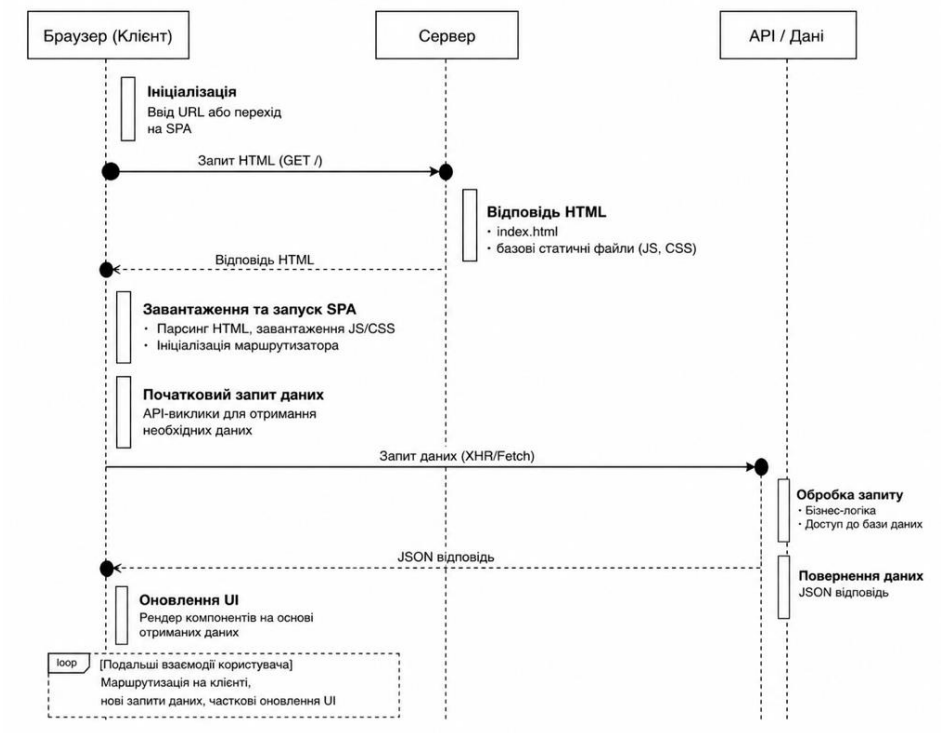

Рисунок 1.5 – Діаграма роботи архітектури Single Page Application

SSR застосунок – це статична генерація сайтів, але з можливістю динамічного та персоналізованого контенту. SSR дозволяє командам оперативно відображати актуальні дані, реагуючи на зміни в режимі реального часу. Користувачі отримують готові сторінки миттєво, без очікування завантаження окремих компонентів. Весь рендеринг відбувається на сервері до відправки в браузер. Хоча динамічне вилучення даних з бази даних або CMS під час навігації по сайту може уповільнити процес, воно гарантує максимальну актуальність контенту [12-14]. Діаграма роботи SSR представлена на рис. 1.6.

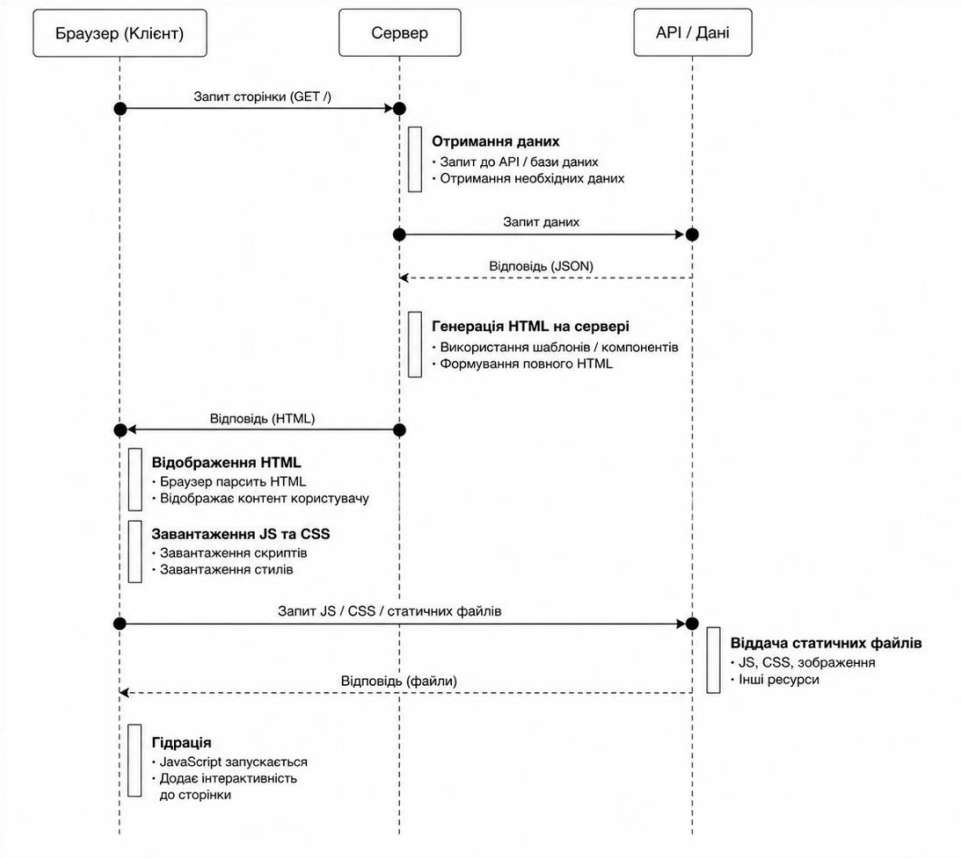

Рисунок 1.6 – Діаграма роботи серверного рендерингу

Розглядалася можливість використання архітектури з чітким розділенням на бекенд і фронтенд, наприклад, з використанням NestJS для серверної частини та Vue.js для клієнтської. Цей підхід пропонує значну гнучкість і повний контроль над серверною логікою, але потребує суттєвих витрат часу на розробку, оскільки необхідно самостійно реалізовувати API, механізми авторизації, валідації та інші базові функції.

Іншим цікавим варіантом є інтеграція Laravel з Inertia.js та Vue.js. Ця комбінація дозволяє створювати додатки, що імітують поведінку SPA, без необхідності розробки окремого API. Такий підхід ідеально підходить для адміністративних панелей та систем, орієнтованих на CRUD-операції, проте може мати обмеження щодо масштабованості та гнучкості при створенні складних клієнтських інтерфейсів.

Підсумовуючи, вибір Nuxt.js видається найбільш збалансованим рішенням, що поєднує швидкість розробки, високу продуктивність та достатню гнучкість. Цей фреймворк дозволяє ефективно реалізувати як клієнтську, так і серверні компоненти додатку, забезпечуючи при цьому відмінний користувацький досвід та відповідність актуальним вимогам веб-розробки.

1.4 Постановка задачі дослідження

Актуальність проблеми та пропоноване рішення: в умовах стрімкої цифровізації освіти та загального зростання стресу в суспільстві батьки стикаються з браком оперативного доступу до якісних виховних методик. Недостатня індивідуалізація в рамках традиційних освітніх систем породжує потребу в інноваційних рішеннях. Розробка інтелектуальної платформи, здатної гнучко адаптуватися до специфічних потреб кожної сім'ї, є нагальною необхідністю для ефективного формування здорового психоемоційного стану дітей.

Об’єктом дослідження є процеси інформаційної та методичної підтримки батьків стосовно психоемоційного розвитку та виховання дітей.

Предметом дослідження є методи та програмні засоби створення інтелектуальної онлайн-платформи для навчання батьків.

Метою роботи є розробка інтелектуальної онлайн-платформи, що забезпечує оволодіння сучасними методами виховання, вікової психології та емоційного підтримання дітей через індивідуальний підхід.

Для досягнення мети необхідно виконати такі **завдання**:

– провести аналіз предметної області психоемоційного розвитку дітей та дослідити сучасні підходи до навчання батьків;

– спроєктувати архітектуру онлайн-платформи, структуру бази даних та механізми взаємодії користувачів із системою;

– розробити вебплатформу з функціоналом реєстрації та авторизації користувачів, проходження тестування, навчальних курсів, системи рекомендацій та особистого кабінету;

– реалізувати серверну частину системи, адміністративну панель, механізми комунікації з користувачами.

Для вирішення поставлених завдань використано методи системного аналізу, порівняльного аналізу наявних рішень, об’єктно-орієнтованого проєктування та алгоритми штучного інтелекту для опрацювання результатів тестування.

Висновки до розділу 1

У першому розділі було проведено аналіз сучасного стану онлайн-освіти для батьків у сфері психоемоційного розвитку дітей. Було визначено ключові поняття, що стосуються дитячої психоемоційної сфери, електронного навчання та інтелектуальних платформ, а також виявлено специфічні потреби батьків у психологічній та освітній підтримці.

Дослідження показало, що існуючі онлайн-платформи та мобільні додатки для батьків мають значні недоліки. Порівняльний аналіз виявив, що жодне з рішень не забезпечує комплексного підходу, який би поєднував навчальні матеріали, тестування, персоналізацію, аналітику та систему рекомендацій.

У розділі також було розглянуто технічні аспекти реалізації інтелектуальної онлайн-платформи. Проаналізовано сучасні технології веброзробки, бази даних та архітектурні підходи.

2 ПРОЄКТУВАННЯ ЛОГІКИ ТА ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Проєктування логіки та інтелектуальної складової системи

Розробка архітектури інтелектуальної платформи здійснюється з використанням методів системного аналізу. Цей підхід дозволяє моделювати програмний продукт як сукупність взаємодіючих компонентів, а саме: користувачів (батьків), освітнього контенту, інтелектуальної системи тестування та централізованої бази даних. Завдяки цьому забезпечується цілісність системи та логічна узгодженість взаємодії між її елементами.

Для детального розглядання вимог та проєктування логіки було використано такі методи системного аналізу:

– метод декомпозиції;

– метод структурного аналізу.

Метод декомпозиції – метод, що дозволяє розбити складні системи на зрозумілі та легко керовані частини. Декомпозиція може бути багаторівнена [15]. Детальніше метод можна розглянути на рис. 2.1.

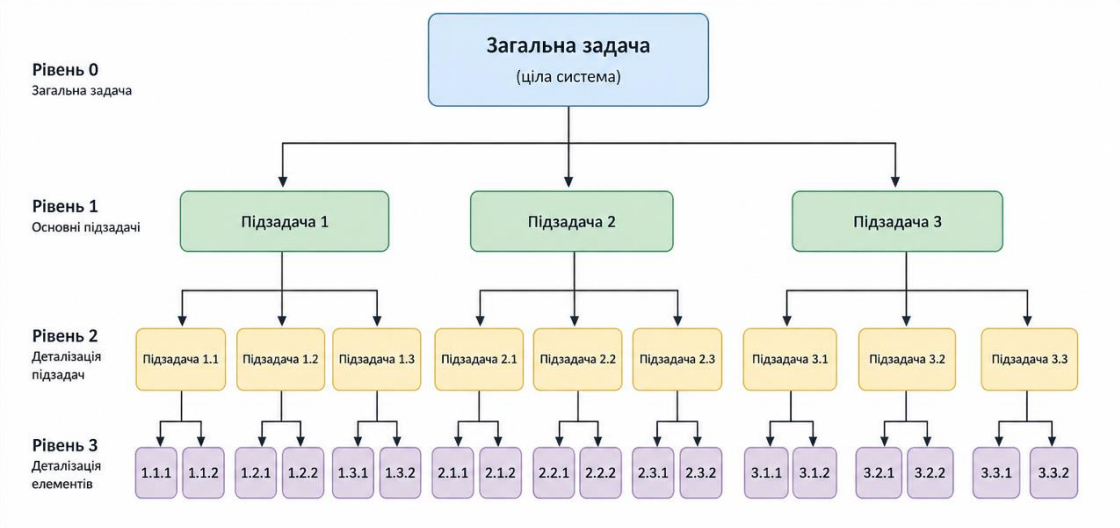

Рисунок 2.1 – Метод декомпозиції

У випадку створення платформи для батьків метод декомпозиції використовується для розбиття загальної мети, тобто створення самої платформи, на підзадачі у вигляді модулів:

– модуль реєстрації та авторизації користувачів;
– модуль тестування (питання, відповіді, результати);
– модуль курсів і навчального контенту;
– модуль збору та збереження статистики користувача;
– модуль формування рекомендацій;
– адміністративна панель для керування контентом.

Метод структурного аналізу полягає у виявленні фундаментальних елементів системи, їхніх характеристик та взаємозалежностей. Детальніше метод можна розглянути на рис. 2.2.

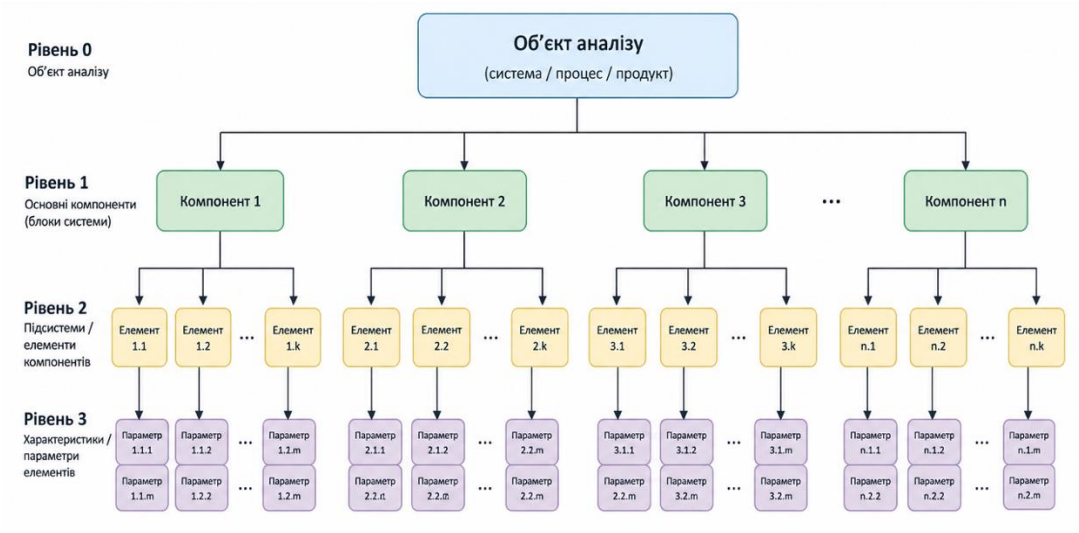

Рисунок 2.2 – Метод структурного аналізу

У контексті розробленої платформи психологічної підтримки батьків, ідентифіковано ключові сутності: користувачі, навчальні курси, тести, відповіді на них, а також допоміжні компоненти, які забезпечують злагоджену роботу та зберігання інформації.

Детальний опис сутностей:

– користувачі: центральним елементом системи є база даних користувачів (таблиця «users»); тут зберігається базова інформація про кожного, включаючи ім'я, контактні дані, фотографію та дату приєднання до платформи; ця таблиця є «ядром», до якого підключені майже всі інші частини системи;

– стан користувача: для відстеження психологічного стану батьків ми створили окрему таблицю «user_stats»; вона фіксує показники рівня стресу та тривожності; оскільки один користувач може мати багато записів про свій стан з часом, ця таблиця пов'язана з таблицею користувачів за принципом «один-до-багатьох», що дозволяє вести історію змін;

– навчання: навчальний контент представлений у вигляді курсів, інформація про які зберігається в таблиці «courses» (назва, опис, зображення); щоб відстежувати, які курси проходять користувачі, як далеко вони просунулися та чи завершили їх, використовується таблиця «user_courses»; вона реалізує зв'язок «багато-до-багатьох», оскільки один користувач може вивчати багато курсів, а один курс можуть проходити багато користувачів;

– тестування: система тестування складається з кількох взаємопов'язаних частин; таблиця «tests» описує самі тести, які можуть бути як незалежними, так і прив'язаними до певного курсу; кожен тест складається з набору запитань, що зберігаються в таблиці «questions»; запитання можуть бути різними за типом: відкритими, з одним правильним варіантом відповіді або з кількома; варіанти відповідей для запитань зберігаються в таблиці «answer_options»; відповіді, які надають користувачі, записуються в таблицю «user_answers»; це дозволяє зберігати як текстові відповіді на відкриті запитання, так і вибір конкретних варіантів;

– підтримка та зворотний зв'язок: для забезпечення можливості звернень користувачів до служби підтримки передбачена таблиця «support_requests»; через неї батьки можуть надсилати запити, які стосуються як конкретних курсів, так і загальних питань;

– інформаційні матеріали: контентна частина платформи представлена статтею, інформація про які зберігається в таблиці «articles»; користувачі можуть

взаємодіяти з цими матеріалами, залишаючи свої думки та запитання у вигляді коментарів, які фіксуються в таблиці «comments»;

– підписки: система також підтримує функціонал підписок, реалізований через таблицю «subscriptions»; це дозволяє відстежувати активність користувачів у зовнішніх сервісах, зокрема в Telegram.

Нижче на рис. 2.3 наведена ER-діаграма до структури бази даних.

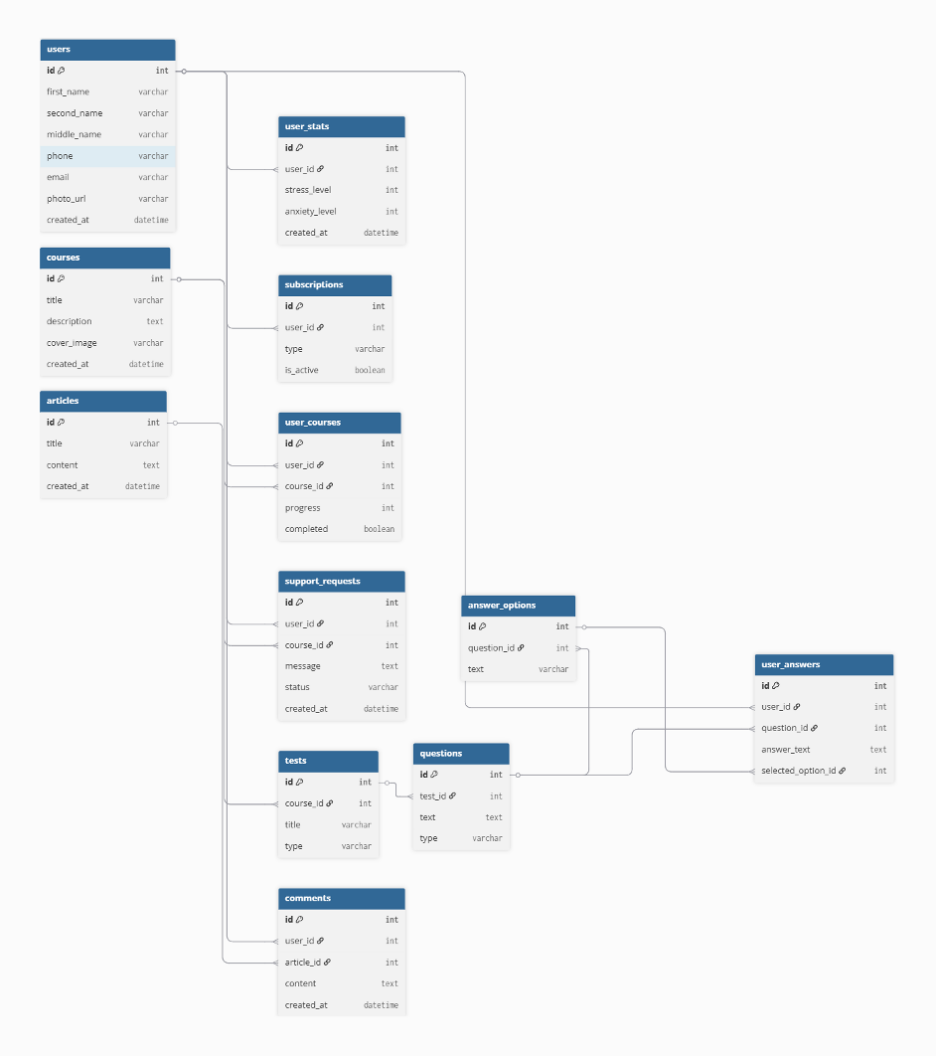

Рисунок 2.3 – ER-діаграма до структури бази даних

Таким чином, розроблена структура бази даних забезпечує надійність, гнучкість та логічну впорядкованість усієї інформації. Вона дозволяє ефективно реалізувати всі ключові функції платформи: навчання, оцінку знань, аналіз психологічного стану та взаємодію користувачів із системою.

Перехід від системного аналізу до програмної реалізації був здійснений за допомогою об'єктно-орієнтованого підходу (ООП). Це дозволило побудувати платформу як систему взаємодіючих об'єктів, що значно підвищує її гнучкість та спрощує масштабування.

У процесі моделювання даних були виділені ключові сутності (класи):

– користувач: центральний елемент, що об'єднує дані про батьків та дітей (вік, особливості розвитку); він також керує авторизацією та зберігає історію взаємодії;

– тест: представляє собою методику оцінювання, що складається з питань, варіантів відповідей та вагових коефіцієнтів для розрахунку результатів;

– результат: зберігає числові та описові дані, отримані після проходження тесту, які є вхідними для інтелектуального модуля;

– рекомендація: динамічно створюється ШІ на основі аналізу «Результату», пропонуючи користувачеві відповідний навчальний контент.

Використання ООП забезпечило модульну архітектуру, де логіка інтелектуального аналізу ізольована від інтерфейсу. Це спрощує тестування окремих компонентів та дозволяє легко розширювати функціонал платформи, додаючи нові тести чи методики, без необхідності зміни базової структури даних.

Розробка платформи базується на ітеративній моделі навчання, що передбачає динамічну адаптацію контенту до потреб користувача. Користувацький сценарій (User Flow) оптимізовано для створення безперешкодного середовища для батьків, мінімізуючи когнітивне навантаження при вивченні психологічних аспектів.

Функціональність платформи реалізована через послідовність ключових етапів, що формують замкнений цикл:

– етап діагностики: система проводить первинне тестування для визначення поточного психоемоційного стану дитини та рівня компетенцій батьків; на основі зібраних даних відбувається фільтрація нерелевантного контенту;

– етап персоналізації: формується індивідуальна освітня траєкторія (рекомендації), що дозволяє користувачеві концентруватися на специфічних викликах (наприклад, вікових кризах або адаптації до навчального закладу);

– етап активної діяльності: користувачі проходять навчальні курси, де теоретичні знання закріплюються за допомогою коротких перевірочних тестів;

– етап моніторингу та рефлексії: після завершення навчального модуля проводиться повторна оцінка, що дозволяє візуалізувати динаміку розвитку користувача та актуалізувати перелік рекомендацій відповідно до нових потреб.

Діаграма User Flow онлайн-платформи представлена на рис. 2.4.

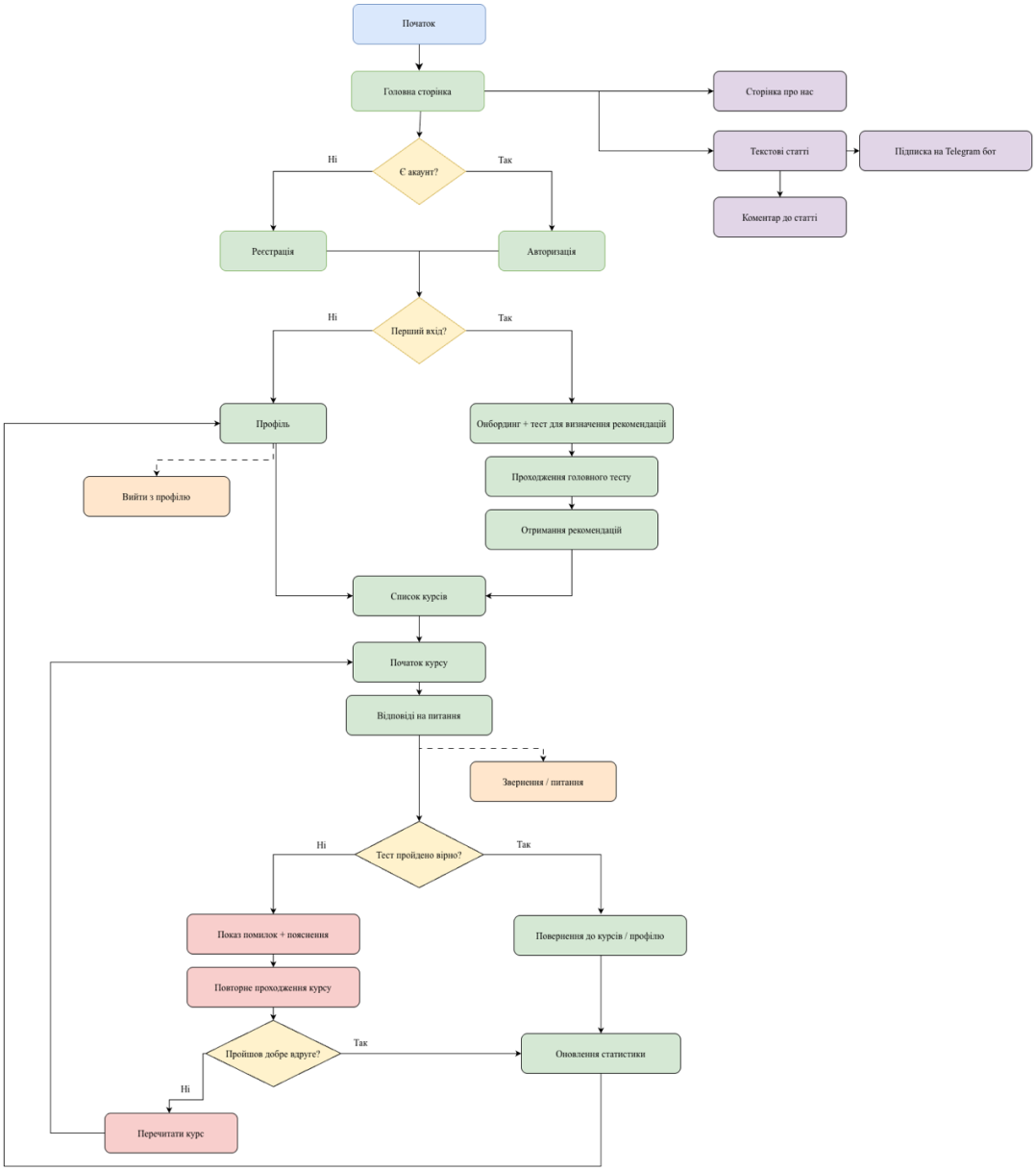

Рисунок 2.4 – Діаграма User Flow онлайн-платформи

Діаграма візуалізує трансформацію процесу: від ініціалізації онбордингу та основного верифікаційного тесту до ітеративного циклу навчання. Особливу увагу приділено механізму обробки некоректних відповідей (блок «Тест пройдено вірно?»): у разі негативного результату, система генерує зворотний зв'язок з детальними роз'ясненнями та ініціює повторне засвоєння контенту.

Такий ітеративний підхід забезпечує адаптивність системи та формує ефективний цикл навчання (Learning Loop), де кожен наступний етап базується на досягненнях попереднього.

2.2 Обґрунтування вибору технології розробки системи

Для розробки клієнтської частини та серверного прошарку інтелектуальної платформи було обрано фреймворк Nuxt.js, побудований на базі Vue.js. Такий вибір зумовлений низкою ключових переваг, що відповідають специфіці системи підтримки батьків у психоемоційному розвитку дітей:

- висока продуктивність та швидкість розробки: платформа забезпечує блискавичну роботу завдяки оптимізованому серверному середовищу Nitro та автоматичному завантаженню компонентів; це означає, що можна швидко впроваджувати функції тестування та взаємодії з користувачами, мінімізуючи складність налаштувань;
- уніфікований підхід до розробки: застосування JavaScript як для фронтенду, так і для бекенду спрощує обслуговування системи; крім того, це дає можливість ефективно використовувати ту саму логіку обробки даних (наприклад, перевірку відповідей у тестах) на обох рівнях; також єдиний стек технологій зменшує кількість помилок при передачі даних між клієнтською та серверною частинами, спрощує підтримку коду та пришвидшує процес розробки нових функціональних модулів;
- готова до використання екосистема: вбудовані модулі для аутентифікації, SEO та роботи з API значно прискорюють створення основних сервісів; це дозволяє

команді зосередитися на розробці унікальних інтелектуальних можливостей платформи.

Використання у проєкті Drizzle ORM та MySQL також мають вагомі переваги як зі технічної, так і з академічної точки зору.

Перевагами Drizzle ORM є такі аспекти:

– типи таблиць і полів генеруються автоматично;

– безпечні SQL запити;

– менше помилок при роботі з базами даних;

– централізоване керування схемою базою даних;

– легша оптимізація запитів;

– відстеження зміни структури бази даних;

– підтримка різних версій проєкту;

– висока продуктивність.

Перевагами MySQL є такі аспекти:

– стабільність;

– безпечне зберігання даних;

– підтримка великих обсягів інформації;

– реляційна структура добре підходить до структури проєкту, де є багато взаємопов’язаних сутностей;

– підтримка складних зв’язків;

– хороша сумісність зі Nuxt/Vue.

Ключовою перевагою проєкту є його гібридна стратегія рендерингу, яка дозволяє підібрати оптимальний спосіб відображення для кожної частини платформи:

– серверний рендеринг (SSR) використовується для публічних сторінок; це забезпечує миттєвий доступ до контенту для відвідувачів та покращує видимість сайту в пошукових системах завдяки ефективному індексуванню;

– односторінковий застосунок (SPA) застосовується для інтерактивних елементів, таких як особистий кабінет та модулі тестування; це створює плавний

досвід користувача, де взаємодія з інтерфейсом (зміна питань, перегляд статистики) відбувається без необхідності перезавантаження сторінки, подібно до роботи мобільного додатку.

На рис. 2.5 відображено діаграми для гібридного підходу SPA + SSR.

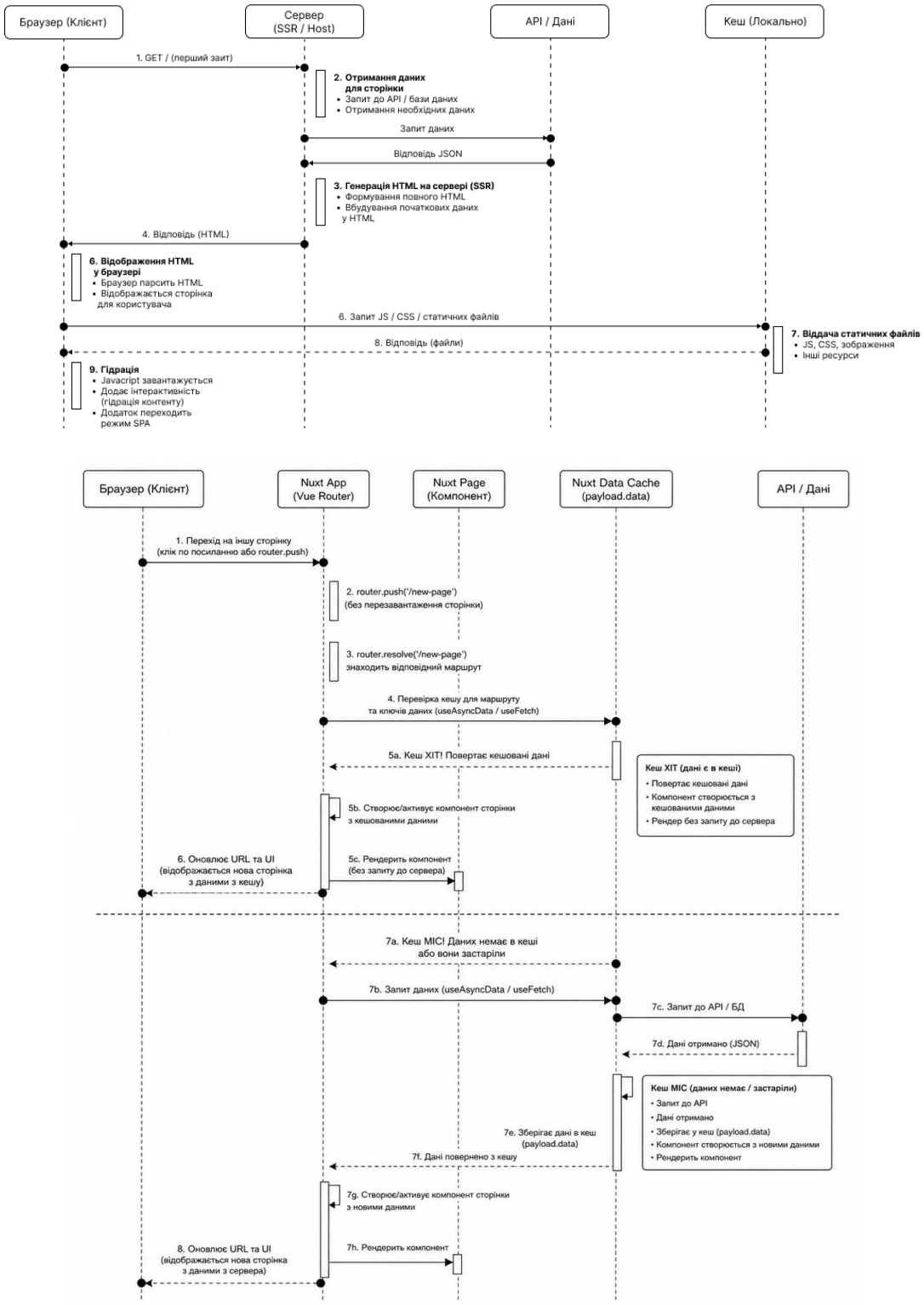

Рисунок 2.5 – Діаграми гібридного підходу SPA + SSR

За допомогою такого підходу платформа має можливість бути легкою для пошукових систем та швидкою для зареєстрованих користувачів.

Адміністративна панель, інтегрована в основний застосунок, функціонує як окремий інтерфейс з обмеженим доступом. Її SPA-архітектура забезпечує зручне керування контентом, включаючи:

- динамічне створення та редагування тестів;
- модерацію контенту та обробку запитів;
- візуалізацію користувацької статистики.

Завдяки впровадженню системи контролю доступу на основі ролей (RBAC), було досягнуто чіткого розмежування повноважень між адміністративним персоналом та батьками користувачів. Це забезпечує високий рівень безпеки, цілісності та конфіденційності інформації, що зберігається на платформі. На рис. 2.6 показано модель RBAC для онлайн-платформи.

Рисунок 2.6 – Модель RBAC для онлайн-платформи

Обрана архітектура, побудована на фреймворку Nuxt.js, формує міцну основу для ефективної роботи інтелектуальних алгоритмів аналізу результатів тестування та подальшого масштабування проєкту.

Висновки до розділу 2

У другому розділі було виконано проєктування архітектури та технологічного стеку для онлайн-платформи, спрямованої на психоемоційний розвиток дітей.

Проведено системний аналіз, що призвів до визначення ключових функціональних модулів (авторизація, тестування, навчальні курси, система рекомендацій, аналітика, адміністративна панель) та їх взаємозв'язків. Застосування методології декомпозиції та структурного аналізу дозволило сформувати логічну структуру системи.

Розроблено схему бази даних та ER-діаграму для представлення зв'язків між сутностями. Особливу увагу приділено проєктуванню інтелектуального ядра платформи, що забезпечує персоналізацію навчання на основі аналізу даних користувача. Сформовано User Flow для оптимізації користувацького досвіду.

Обґрунтовано вибір технологічного стеку, що включає Nuxt (на базі Vue.js), Node.js, Drizzle ORM та MySQL, з урахуванням вимог до продуктивності, типобезпечності та масштабованості. Визначено доцільність використання гібридного підходу SPA + SSR для покращення взаємодії користувача та SEO-оптимізації.

3 РОЗРОБКА ОНЛАЙН-ПЛАТФОРМИ ДЛЯ НАВЧАННЯ БАТЬКІВ

3.1 Розробка користувацького інтерфейсу

Назва платформи «MindNest – простір свідомого батьківства» відтворює ідею безпечного оточення («гнізда») для зростання мислення та емоційного інтелекту дитини. Вона уособлює поєднання психологічної допомоги та навчальних матеріалів для батьків, націлених на формування збалансованого розвитку дитини.

Символ платформи виконаний у стриманому стилі із застосуванням контурного зображення родини у гнізді, що символізує безпечне середовище для розвитку дитини. Плавні лінії та заокруглені форми наголошують на задумі піклування, злагоди та емоційної підтримки. Логотип зазначений нижче на рис. 3.1.

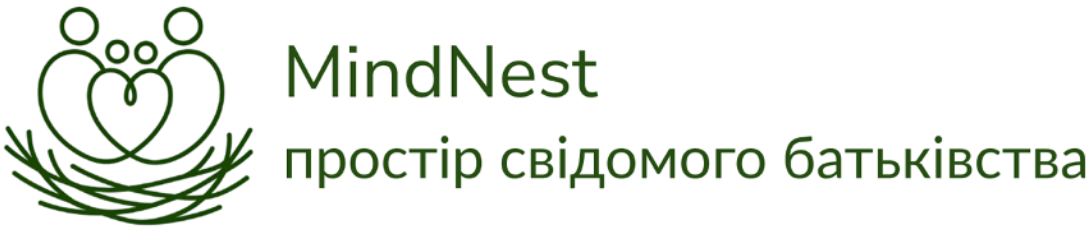

Рисунок 3.1 – Логотип платформи MindNest

Вибір кольорової схеми для інтерфейсу базується на принципах зручності користування (UX) та психології сприйняття кольору. Це робиться для того, щоб зробити взаємодію з системою максимально зрозумілою та ефективною для користувача. На рис 3.2 наведені обрані кольори для сайту.

Рисунок 3.2 – Кольори сайту

Для розробленої онлайн-платформи обрано палітру м'яких пастельних відтінків. Такі кольори створюють позитивний емоційний настрій, допомагають зменшити відчуття тривоги та підвищують довіру користувачів до цифрового середовища. Кольори гармонійно доповнюють один одного, надаючи інтерфейсу легкості та повітряності. Це особливо важливо для сервісів, які спрямовані на психологічну підтримку та навчання.

Значення обраних кольорів:

– зелений: основний колір платформи; він асоціюється з гармонією, стабільністю, зростанням та безпекою; використання зеленого в ключових елементах, таких як кнопки та індикатори прогресу, допомагає користувачам відчувати контроль над навчальним процесом та сприяє спокійному емоційному стану;

– рожевий: використовується як акцентний колір для передачі емоційної складової платформи; він символізує турботу, співчуття та підтримку, що є надзвичайно важливим для батьків, які взаємодіють з дітьми; приглушені відтінки рожевого дозволяють уникнути надмірної емоційності, зберігаючи баланс між теплотою та стриманістю;

– фіолетовий: підкреслює інтелектуальну та психологічну спрямованість платформи; цей колір асоціюється з самоаналізом, внутрішнім розвитком та усвідомленням емоційних процесів, що безпосередньо відповідає меті платформи – розвитку емоційного інтелекту батьків;

– світло-жовтий: створює відчуття тепла, підтримки та оптимізму; його доцільно використовувати в блоках з рекомендаціями, підказками та додатковою інформацією; цей колір привертає увагу, не викликаючи напруження чи перевантаження інтерфейсу;

– блакитний: доповнює палітру, виступаючи маркером довіри, стабільності та ясності; він особливо ефективний для відображення аналітичних даних, результатів тестів та візуалізації прогресу, оскільки підсилює сприйняття інформації як об'єктивної та надійної.

Таким чином, розроблена кольорова палітра не тільки відповідає загальній тематиці платформи, але й виконує важливі функції у взаємодії з користувачем. Вона зменшує когнітивне навантаження, полегшує навігацію та створює сприятливе емоційне середовище для навчання та саморозвитку.

Головна сторінка є основою даної онлайн-платформи. Майже вся логіка зосереджена на даній сторінці. Це забезпечує додатковий комфорт у користуванні платформою користувачам. Сторінка продумана таким чином, щоб поступово ознайомити користувача з її функціями.

У верхній частині розміщене навігаційне меню, яке має у собі базові розділи: курси, блог, інформацію про платформу, а також кнопки для входу та реєстрації. У центрі першого екрану знаходиться основний інформаційний блок із коротким описом можливостей системи, кнопками для проходження тесту та ознайомлення з платформою, а також тематичне зображення сім'ї. Використання сімейної ілюстрації допомагає встановити емоційний зв'язок із користувачем і підкреслити тематику психоемоційного розвитку дітей.

Нижче розташований розділ «Як це працює», який у простій покроковій формі демонструє основні етапи взаємодії з платформою: проходження тестування, отримання рекомендацій, навчання, відстеження прогресу та досягнення результатів. Кожен крок супроводжується відповідною іконкою та коротким поясненням для кращого сприйняття.

Також на головній сторінці є блок особистого кабінету, що демонструє можливості системи в частині аналітики результатів тестів, рекомендацій та моніторингу прогресу користувача. Окрім того, передбачено блок із блогом та інформаційними статтями, які надають користувачам додаткові матеріали та поради з питань дитячої психології та емоційного розвитку.

Інтерфейс розроблено з урахуванням принципів UX/UI дизайну: застосовано зрозумілу навігацію, логічне структурування інформації, великі інтерактивні елементи та мінімум відволікаючих деталей. Особлива увага приділялася

адаптивності інтерфейсу для коректного відображення на різних пристроях. На рис. 3.3 відображено макет головної сторінки сайту.

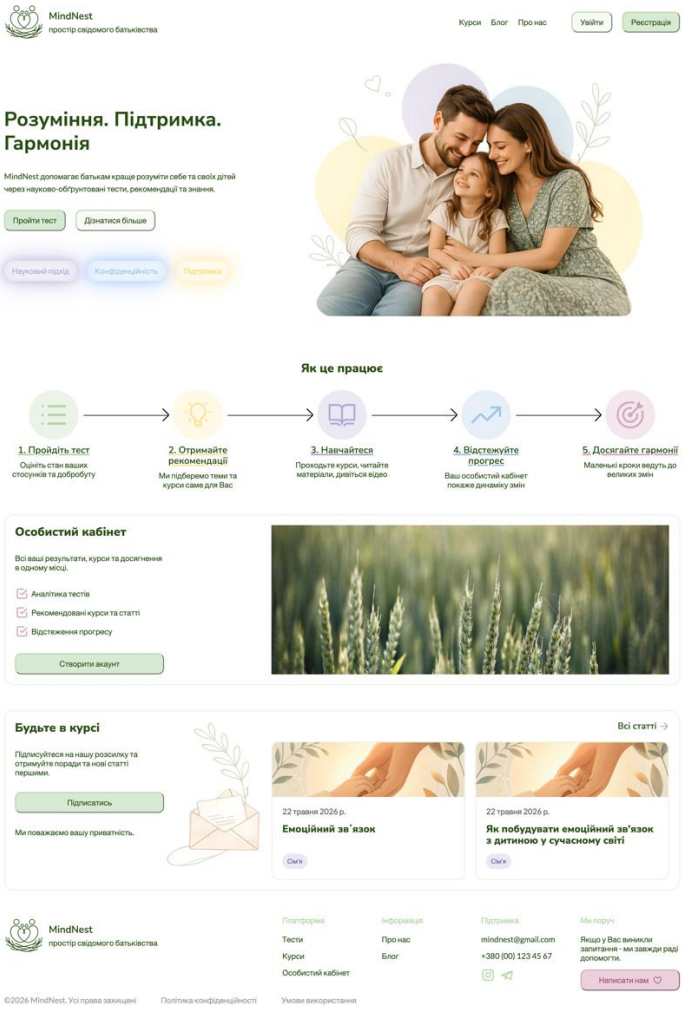

Рисунок 3.3 – Макет головної сторінки

Обов’язково на сайті присутні статичні сторінки, які слугують важливим комунікаційним інструментом. Ці сторінки, зокрема «Про нас», «Умови використання» та «Політика конфіденційності», надають ключову інформацію, що полегшує користування сервісом.

Сторінка «Про нас» – це ознайомлення з платформою. Тут можна дізнатися про її місію, що пропонує сайт, про команду та цінності. Особливо важливо, що цей розділ розкриває соціальну місію платформи, спрямованої на підтримку батьків у розвитку їхніх дітей, та підкреслює її освітню цінність. На рис. 3.4 відображено макет сторінки «Про нас».

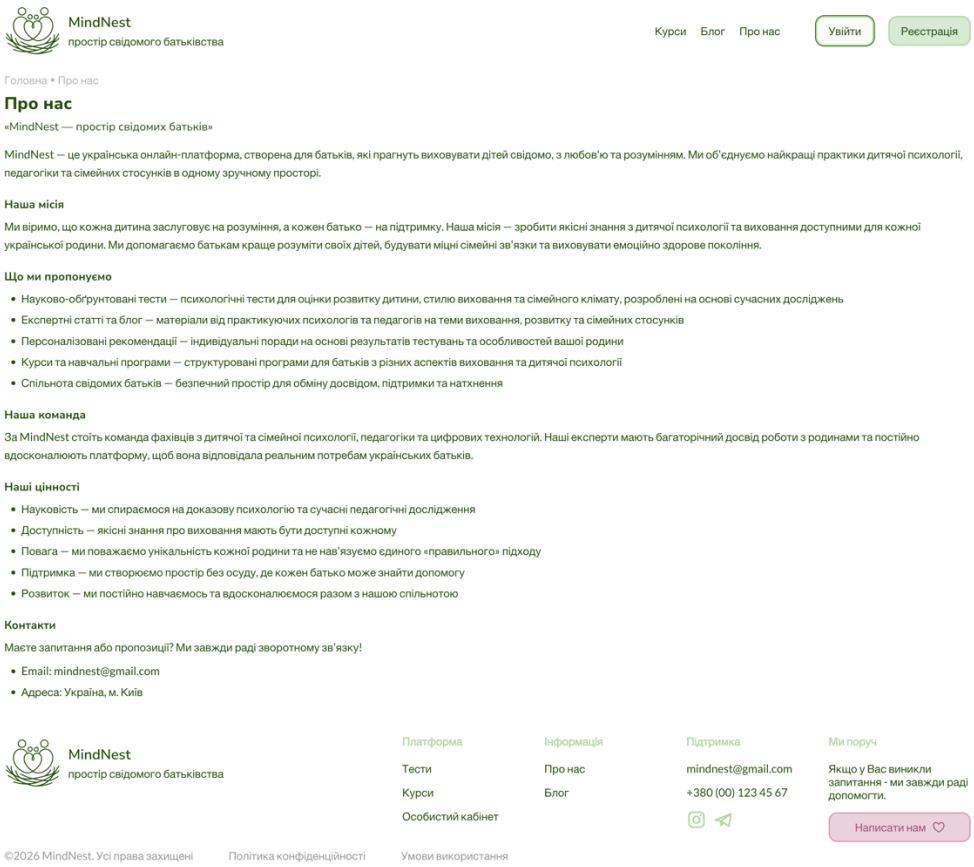

Рисунок 3.4 – Макет сторінки «Про нас»

Сторінка «Умови використання» – це правила сайту. Вони чітко визначають, як можна взаємодіяти з платформою, які користувач має права та обов'язки, і як отримати доступ до навчальних матеріалів, тестів та рекомендацій. Тут також йдеться про захист контенту, щоб навчальні матеріали були в безпеці.

«Політика конфіденційності» – це гарантія безпеки даних. Оскільки платформа збирає інформацію для персоналізації (реєстрація, тести, рекомендації), цей документ пояснює, як дані збираються, використовуються, зберігаються та захищаються особисті дані. Сайт дотримується найвищих стандартів безпеки, щоб дані залишалися конфіденційними.

Таким чином, ці статичні сторінки не тільки інформують вас, але й будують довіру до платформи, роблять її роботу зрозумілішою та забезпечують дотримання всіх необхідних правил та норм.

Для розширення інформаційної підтримки користувачів, платформа тепер має розділ «Блог». Його головне завдання – забезпечити батьків актуальними

матеріалами, порадами та експертними статтями з питань психоемоційного розвитку дітей, виховання, психології та гармонійних сімейних стосунків.

Інтерфейс блогу побудований на принципах адаптивної сітки, що відображає інформаційні картки. Кожна картка візуально виділена тематичним зображенням, датою публікації, назвою та належністю до певних категорій. Це забезпечує інтуїтивно зрозумілу навігацію та швидкий пошук контенту.

Дизайн карток виконаний у єдиній стилістиці, що характеризується м'якою кольоровою гамою, заокругленими елементами та тематичними ілюстраціями для приємного візуального досвіду. Категорії статей позначені кольоровими тегами, що спрощує перегляд матеріалів за різними напрямками.

Для зручності навігації вгорі сторінки розміщено навігаційний ланцюжок (breadcrumbs), який показує поточне місцезнаходження користувача та дозволяє миттєво переходити до інших розділів. При великій кількості публікацій передбачена пагінація для зручного перегляду.

На рис. 3.5 відображений макет сторінки «Блог».

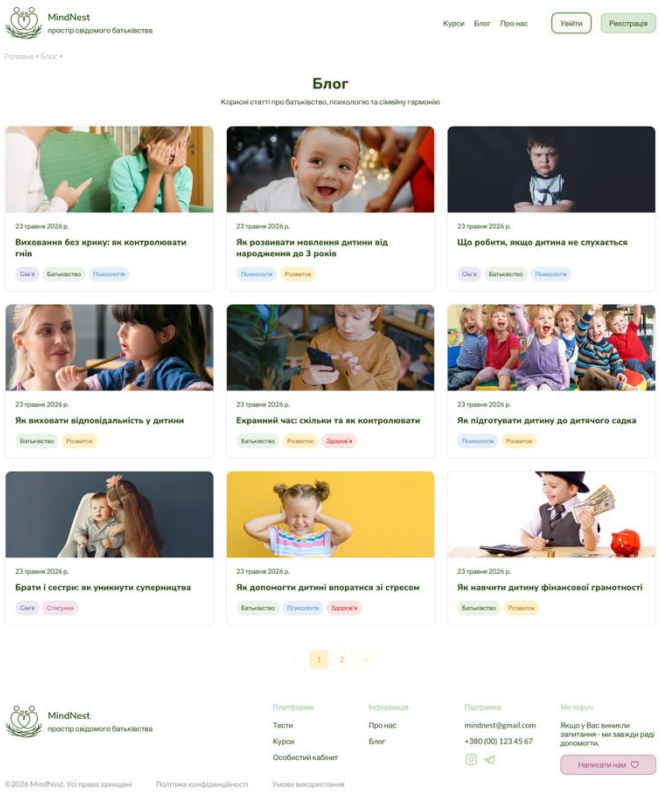

Рисунок 3.5. – Макет сторінки «Блог»

Отже, блог слугує не тільки джерелом інформації, але й активно формує освітнє середовище платформи, пропонуючи користувачам цінні додаткові матеріали поза межами основних курсів та тестів.

Для детального ознайомлення з інформаційними матеріалами на платформі створено окрему сторінку статті. Її головна функція полягає у впорядкованому представленні текстового та мультимедійного контенту у зручному для сприйняття вигляді.

У верхній частині сторінки розташований навігаційний шлях, який дає змогу користувачу швидко повернутися до попередніх розділів платформи. Центральним елементом є велике тематичне зображення, що привертає увагу та допомагає встановити емоційний зв’язок із матеріалом.

Текст статті організовано за допомогою заголовків, підзаголовків, абзаців і списків. Такий формат покращує читабельність і полегшує пошук необхідної інформації. Для логічного розмежування змістових блоків застосовано горизонтальні лінії та достатню кількість вільного простору між секціями.

Особлива увага приділена візуальній структуризації контенту, що дозволяє користувачам швидко орієнтуватися у матеріалі незалежно від його обсягу. Заголовки різних рівнів формують чітку ієрархію інформації та допомагають визначити основні теми й підтеми статті. Марковані та нумеровані списки використовуються для подання рекомендацій, порад та послідовностей дій у зручному для сприйняття вигляді.

Для підвищення комфорту читання обрано достатній міжрядковий інтервал, оптимальний розмір шрифту та контрастне поєднання кольорів тексту й фону. Відступи між абзацами та окремими елементами інтерфейсу сприяють зменшенню візуального навантаження та забезпечують кращу концентрацію уваги на змісті матеріалу.

Додатково статті можуть містити ілюстрації, інформаційні блоки та тематичні рекомендації, які допомагають краще зрозуміти описані методики виховання та психоемоційного розвитку дітей. Завдяки адаптивному дизайну

сторінка коректно відображається як на персональних комп'ютерах, так і на мобільних пристроях, забезпечуючи зручний доступ до навчальних матеріалів у будь-який час.

Крім тексту, сторінка підтримує вставку мультимедійних елементів, зокрема відео з зовнішніх платформ. Це робить процес навчання більш різноманітним, інтерактивним і легшим для сприйняття.

У нижній частині сторінки розміщені тематичні категорії статті, що допомагає користувачам переходити до пов'язаних матеріалів і формує зв'язки між різними інформаційними ресурсами платформи.

На рис. 3.6 відображений макет сторінки «Стаття».

Рисунок 3.6. – Макет сторінки «Стаття»

Отже, сторінка статті забезпечує зручне подання освітнього контенту, відповідає загальній дизайн-системі платформи та сприяє покращенню користувацького досвіду під час ознайомлення з інформацією.

Сторінка особистого профілю є ключовим елементом для забезпечення індивідуального досвіду користувача на платформі. Вона надає повний контроль над персональними даними, дозволяє переглядати результати навчальних активностей (тестів, курсів) та моніторити особистий прогрес.

Структура сторінки базується на логічно організованих інформаційних блоках, що забезпечує інтуїтивно зрозумілу навігацію. Ліва частина сторінки зайнята боковим меню, яке містить основні розділи профілю. Такий підхід гарантує миттєвий доступ до ключових функцій платформи.

Центральна область сторінки відображає основну інформацію про користувача, включаючи ім'я, електронну адресу та короткий опис. Редагування цих даних здійснюється через мінімалістичну та інтуїтивно зрозумілу форму, яка відповідає загальній дизайн-системі платформи.

Усі поля форми мають зрозумілі підписи та підказки для введення даних, що забезпечує зручність використання навіть для користувачів без значного досвіду роботи з вебсервісами. Після внесення змін користувач може зберегти оновлену інформацію, яка автоматично оновлюється в базі даних системи. Для підвищення надійності передбачено перевірку коректності введених даних, зокрема формату електронної пошти та обов’язкових полів.

Окремий блок присвячений візуалізації прогресу навчання. Тут представлено загальний рівень засвоєння матеріалів та детальні результати за окремими напрямками. Прогрес-бари та відсоткові показники забезпечують швидку оцінку поточного стану навчального процесу.

Таким чином, сторінка особистого профілю виступає як централізований хаб для управління персональною інформацією, забезпечує легкий доступ до функціоналу платформи та підтримує персоналізований підхід до навчання та розвитку.

На рис. 3.7 зображено макет профілю.

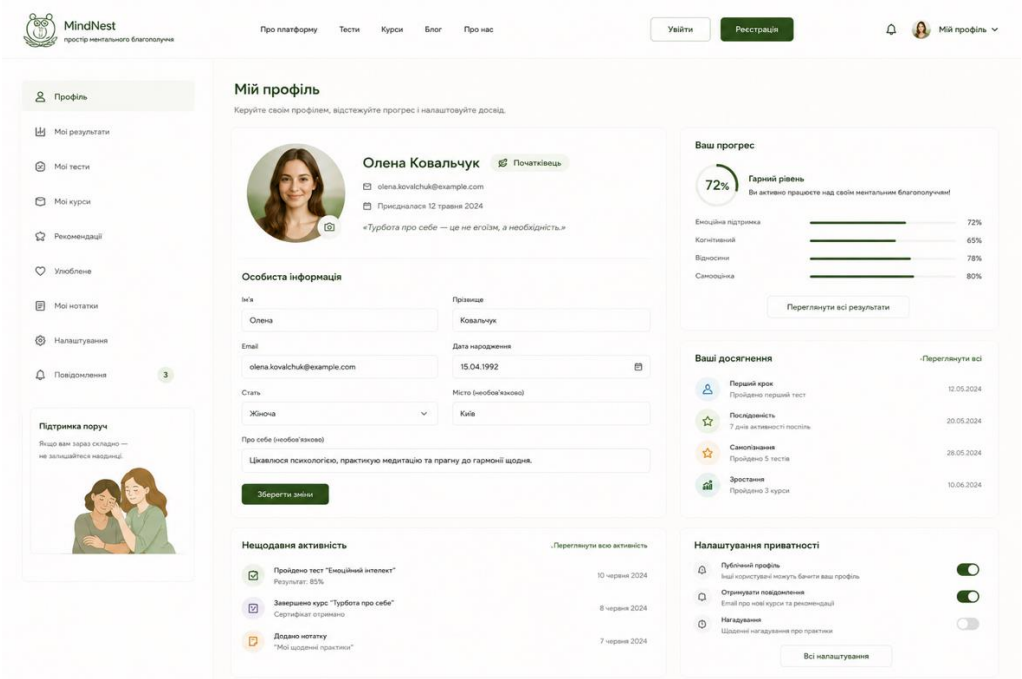

Рисунок 3.7 – Макет сторінки профілю

Створений користувацький інтерфейс забезпечує зручну взаємодію з платформою, підтримує її загальну концепцію та сприяє формуванню позитивного емоційного досвіду під час використання вебзастосунку.

3.2 Проєктування бази даних

Одним із важливих етапів створення онлайн-платформи MindNest стало проєктування структури бази даних, адже саме вона відповідає за збереження, обробку та взаємозв’язок усієї інформації системи. Від правильної організації даних залежить стабільність роботи вебзастосунку, швидкість обробки інформації, безпека збереження даних користувачів та можливість подальшого розширення платформи.

Під час розробки було обрано реляційну модель бази даних, що дає змогу ефективно встановлювати зв’язки між сутностями та уникати дублювання інформації. Як систему управління базами даних використали MySQL, оскільки вона є однією з найпопулярніших і найнадійніших СКБД для вебзастосунків,

забезпечує високу продуктивність, підтримує індексацію, гарантує цілісність даних і має широку сумісність із сучасними інструментами веброзробки.

Для роботи з базою даних у проєкті застосовується ORM-бібліотека Drizzle ORM. Використання ORM-підходу дозволяє описувати структуру таблиць за допомогою TypeScript, що значно полегшує підтримку проєкту, покращує типізацію даних і знижує ризик помилок при виконанні SQL-запитів. Крім того, ORM забезпечує централізоване керування структурою бази даних і спрощує внесення змін у систему в майбутньому. Нижче на рис. 3.8 наведено приклад оформлення файлу зі таблицями для бази даних.

```
import { mysqlTable, varchar, text, datetime, int, boolean, mysqlEnum } from 'drizzle-orm/mysql-core'
import { sql } from 'drizzle-orm'

export const users = mysqlTable('users', {
  id: int( name: 'id').autoincrement().primaryKey(),
  firstName: varchar( name: 'first_name',  config: { length: 255 }).notNull(),
  secondName: varchar( name: 'second_name',  config: { length: 255 }).notNull(),
  middleName: varchar( name: 'middle_name',  config: { length: 255 }),
  phone: varchar( name: 'phone',  config: { length: 50 }),
  email: varchar( name: 'email',  config: { length: 255 }).notNull().unique(),
  password: varchar( name: 'password',  config: { length: 255 }).notNull(),
  photoUrl: varchar( name: 'photo_url',  config: { length: 500 }),
  role: mysqlEnum( name: 'role',  values: ['user', 'admin', 'superadmin']).default( value: 'user'),
  createdAt: datetime( name: 'created_at').default(sql`CURRENT_TIMESTAMP`),
})
```

Рисунок 3.8 – Оформлення майбутньої БД

Проєктування бази даних проводилося з урахуванням функціональних вимог платформи. Було визначено основні сутності системи, серед яких:

– користувач;
– пости;
– теги;
– теги постів;
– статичні сторінки;
– питання.

Між сутностями були встановлені логічні зв’язки, що забезпечують коректну взаємодію компонентів системи та цілісність даних.

Основною складовою бази даних є таблиця «users», яка містить інформацію про зареєстрованих користувачів платформи. Ця таблиця відповідає за реалізацію процесів авторизації, персоналізації та взаємодії користувачів із системою.

У таблиці зберігаються такі дані:

– ім'я користувача;
– прізвище;
– по батькові;
– номер телефону;
– електронна адреса;
– пароль;
– фотографія профілю;
– роль користувача;
– дата створення акаунта.

Поле id виконує роль первинного ключа і автоматично генерується для кожного нового запису. Поле email має унікальне значення, що запобігає реєстрації кількох акаунтів з однаковою електронною поштою.

Для контролю рівня доступу до функцій системи використовується поле role, яке реалізоване як тип enum. У системі передбачено кілька ролей:

– user – звичайний користувач платформи;
– admin – адміністратор контенту;
– superadmin – користувач з повним доступом до системи.

З метою забезпечення безпеки, паролі не зберігаються у відкритому вигляді, а перед записом у базу даних проходять процедуру хешування.

Для забезпечення інформаційної складової платформи була розроблена база даних, що містить таблицю «posts». Ця таблиця слугує сховищем для різноманітного контенту, включаючи статті, рекомендації та навчальні матеріали.

Основні дані, що зберігаються в таблиці «posts»:

– заголовок матеріалу;
– URL-ідентифікатор (slug);

– текстовий вміст;
– зображення-обкладинка;
– статус публікації;
– дати створення та останнього оновлення.

Поле slug відіграє ключову роль у формуванні читабельних URL-адрес, що позитивно впливає на видимість вебзастосунку в пошукових системах.

Поле published надає можливість керувати доступністю контенту. Адміністратори можуть створювати чернетки, редагувати їх та публікувати лише після завершення роботи, забезпечуючи контроль над процесом публікації.

Автоматичне збереження дат створення та оновлення записів дозволяє завжди мати актуальну інформацію про час появи та редагування контенту.

Наявність такого інформаційного розділу є важливим елементом платформи, оскільки він дозволяє надавати користувачам цінну інформацію з таких тем, як психоемоційний розвиток дітей, взаємодія батьків та сімейна психологія.

Для ефективної організації матеріалів за тематичними напрямками була створена таблиця «tags». Вона дозволяє групувати статті за певними категоріями.

Дані, що зберігаються в таблиці «tags»:

– назва тегу;
– унікальний slug;
– колір фону;
– колір тексту.

Використання кольорових тегів у інтерфейсі користувача робить контент більш привабливим візуально та спрощує процес пошуку потрібної інформації.

Оскільки одна стаття може бути віднесена до кількох категорій одночасно, а один тег може застосовуватися до багатьох статей, між таблицями «posts» та «tags» було реалізовано зв'язок типу «багато до багатьох».

Для реалізації цього зв'язку була створена допоміжна таблиця «post_tags». Вона містить посилання на статті та теги за допомогою зовнішніх ключів:

– post_id: ідентифікатор статті;

– tag_id: ідентифікатор тегу.

Використання окремої таблиці для зв'язків забезпечує гнучкість бази даних. Це дозволяє легко змінювати систему категоризації без необхідності дублювати інформацію.

Крім того, для забезпечення цілісності даних застосовується механізм каскадного видалення (onDelete: cascade). Це означає, що при видаленні статті або тегу відповідні записи в таблиці зв'язків «post_tags» будуть автоматично видалені, що запобігає появі «сирітських» записів та підтримує порядок у базі даних.

Ця структура бази даних дозволяє ефективно керувати контентом блогу, забезпечуючи його організованість, доступність та зручність для користувачів платформи. Вона є фундаментом для створення потужного інформаційного ресурсу, що підтримує розвиток користувачів у сферах сімейної психології та особистісного зростання.

Для розміщення інформаційних сторінок нашої платформи використовується спеціальна таблиця під назвою «static_pages». Ця таблиця призначена для зберігання контенту тих сторінок, які змінюються вкрай рідко і мають постійний характер.

До таких сторінок належать, наприклад:

– сторінка «Про нас»;
– сторінка «Політика конфіденційності»;
– сторінка «Умови використання».

У таблиці «static_pages» зберігається така інформація:

– назва сторінки;
– slug;
– текстовий контент;
– дата оновлення;

Виділення статичних сторінок в окрему таблицю дає змогу адміністраторам легко оновлювати інформацію на цих сторінках, не втручаючись у програмний код самої платформи.

Для забезпечення зворотного зв'язку з користувачами створено таблицю «questions». Вона слугує для запису всіх повідомлень та запитань, які надходять від користувачів платформи.

Таблиця «questions» містить такі дані:

– ім'я користувача;

– електронна адреса;

– текст повідомлення;

– статус обробки;

– дата створення.

Поле status реалізовано за допомогою спеціального типу даних (enum), що дозволяє чітко відстежувати, на якому етапі знаходиться звернення. У системі передбачено два основні статуси:

– new: повідомлення ще не було розглянуто;

– resolved: повідомлення було оброблено та вирішено.

Цей функціонал є ключовим для підтримки ефективної комунікації між адміністрацією платформи та її користувачами, що є невід'ємною частиною сучасних вебсистем.

Розроблена структура бази даних має низку суттєвих переваг:

– логічні зв'язки: дані різних сутностей (наприклад, користувачів та їхніх запитів) пов'язані між собою логічно;

– уникнення дублювання: кожна інформація зберігається лише один раз, що запобігає розбіжностям;

– масштабованість: система легко адаптується до зростання обсягів даних та кількості користувачів;

– просте адміністрування: керувати даними та контентом стає значно легше;

– персоналізація: можливість адаптувати контент під потреби конкретних користувачів;

– розширюваність: структура дозволяє легко додавати новий функціонал у майбутньому.

Крім того, використання ORM (Object-Relational Mapping) підходу забезпечує кращу взаємодію між логікою програми та базою даних. Це спрощує підтримку коду та прискорює процес розробки вебзастосунку.

В результаті проєктування бази даних була розроблена реляційна модель, що відзначається структурованістю. Ця модель гарантує оптимальне зберігання, обробку та взаємодію даних для онлайн-платформи MindNest. Сформована структура повністю відповідає функціональним вимогам системи, забезпечує її масштабованість та закладає фундамент для майбутнього розвитку вебплатформи.

3.3 Реалізація логіки взаємодії користувача з платформою через серверну частину

Користувачі можуть легко поставити запитання адміністрації платформи за допомогою зручної форми зворотного зв'язку. Ця форма відкривається у спливаючому вікні, де потрібно вказати своє ім'я, прізвище, електронну пошту та написати саме повідомлення. Після заповнення всіх полів, інформація надсилається на сервер. На рис. 3.9 відображений popup для запитань.

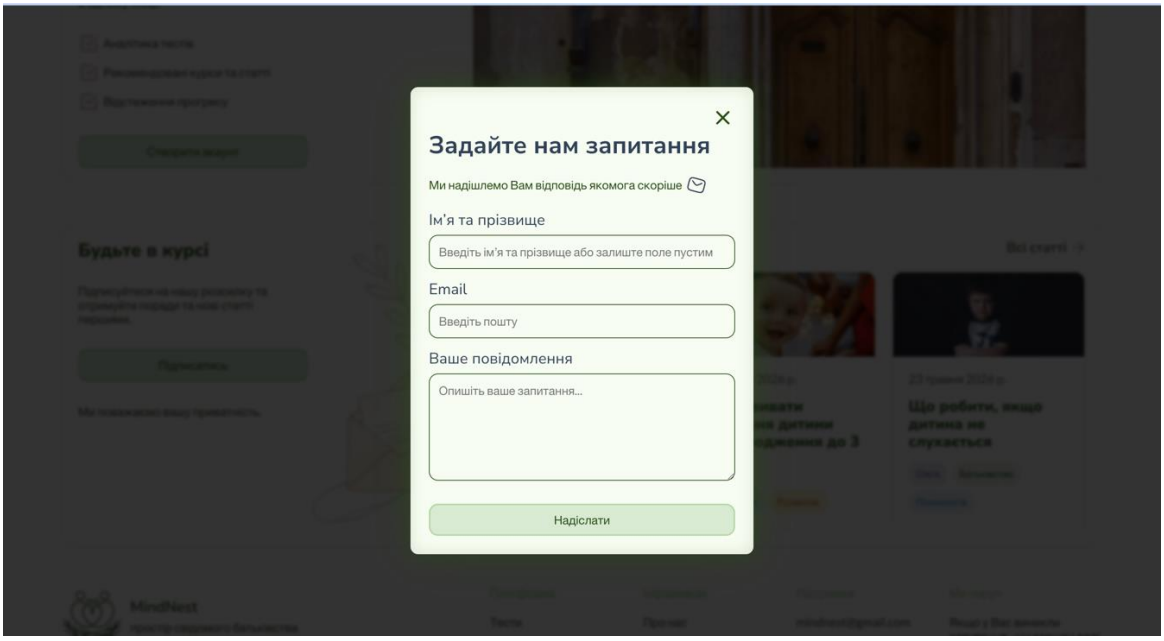

Рисунок 3.9 – Popup для запитань

Щоб уникнути помилок, система спочатку перевіряє введені дані ще до їх надсилання. Вона переконується, що електронна пошта введена правильно, а всі обов'язкові поля заповнені. Якщо виникне помилка, користувач побачить відповідне повідомлення. Якщо все гаразд, з'явиться сповіщення про успішне надсилання запитання. На рис. 3.10 відображено валідацію полів форми.

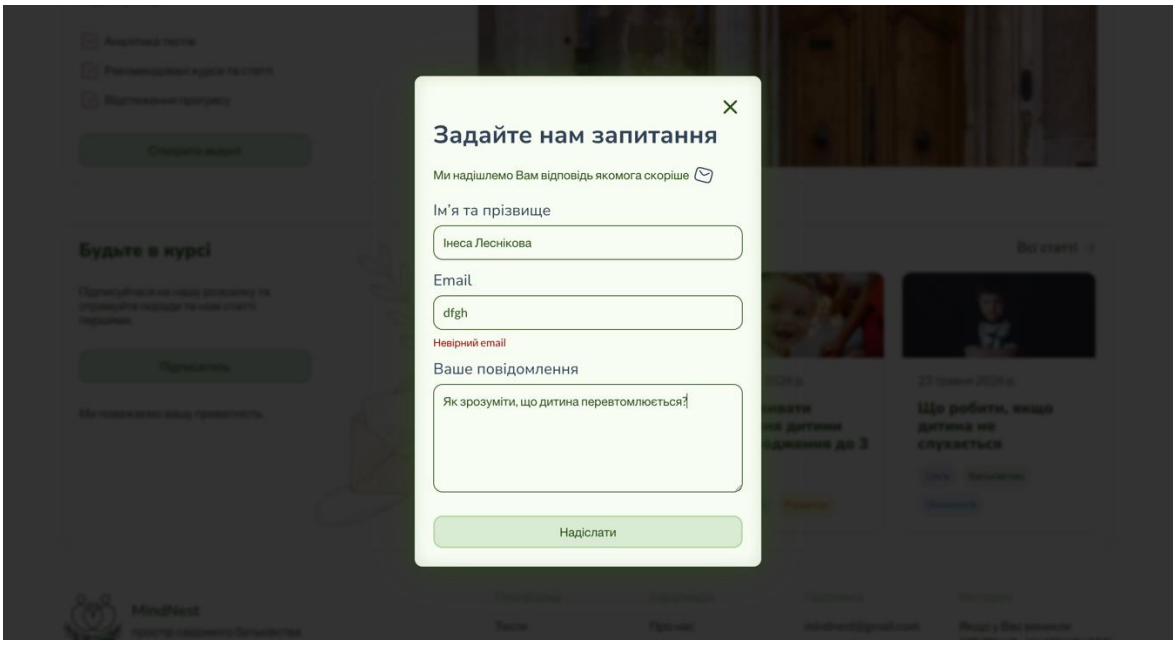

Рисунок 3.10 – Валідація полів форми

Сервер отримує запитання користувача через спеціальний канал зв'язку (API). Там він ще раз перевіряє всі дані та зберігає запитання в базі даних. Кожне запитання містить:

– ім'я та прізвище користувача;
– його електронну пошту;
– текст повідомлення;
– дату, коли було надіслано запитання;
– поточний статус запитання.

Нові запитання одразу отримують статус «Нове», що означає, що адміністратор ще не почав їх розглядати.

Адміністратори мають окремий розділ на платформі, де вони бачать усі запитання від користувачів. Тут відображається вся необхідна інформація: дата

надсилання, контакти користувача, текст запитання та його поточний статус. Для зручності є можливість шукати запитання, переглядати повний текст повідомлення та змінювати статус. Нижче на рис. 3.11 відображено адміністративну панель відділу зі запитаннями.

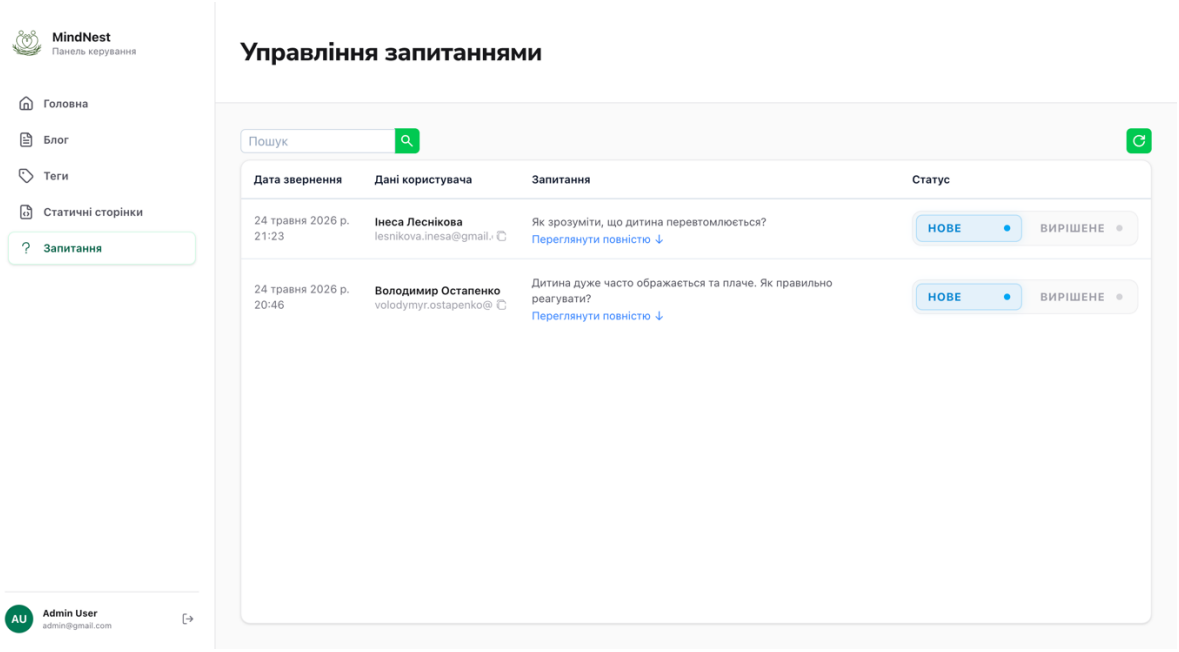

Рисунок 3.11 – Адміністративна панель відділу зі запитаннями

Коли адміністратор відкриває детальну інформацію про запитання, він може написати відповідь прямо в системі. Після натискання кнопки «Відправити», система робить наступне:

– отримує дані про запитання, електронну пошту користувача та текст відповіді;

– надсилає відповідь користувачеві на електронну пошту;

– перевіряє, чи існує запитання в базі даних;

– автоматично змінює статус запитання з «Нове» на «Вирішено».

На рис. 3.12 відображено можливість написати відповідь користувачу прямо у системі.

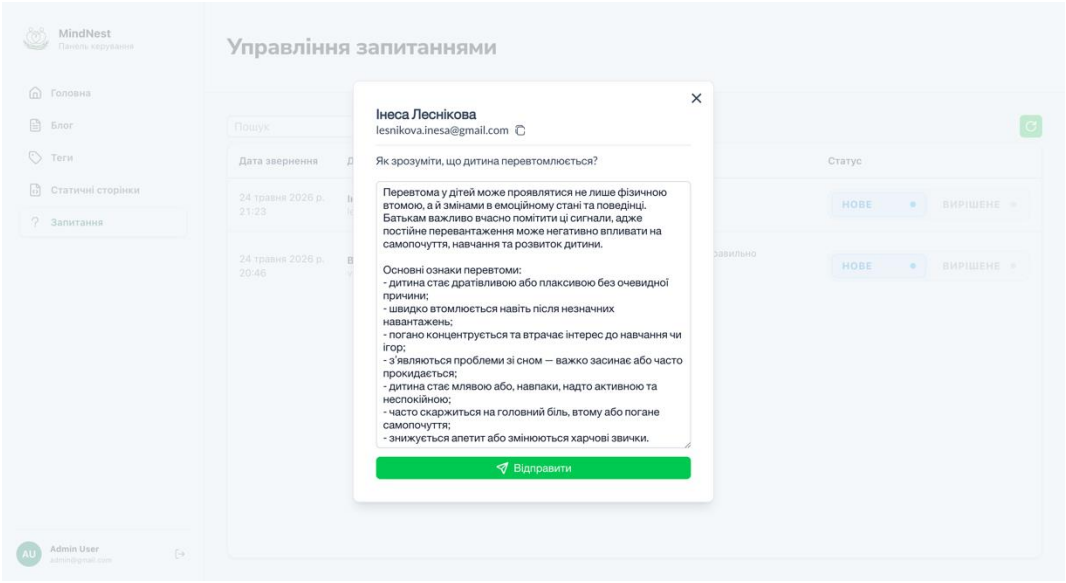

Рисунок 3.12 – Написання відповіді користувачу

Таким чином, після надсилання відповіді, запитання автоматично вважається опрацьованим. Це допомагає адміністраторам відстежувати, які запитання вже вирішені, і уникати повторного розгляду. На рис. 3.13 видно відображення вирішеного запитання.

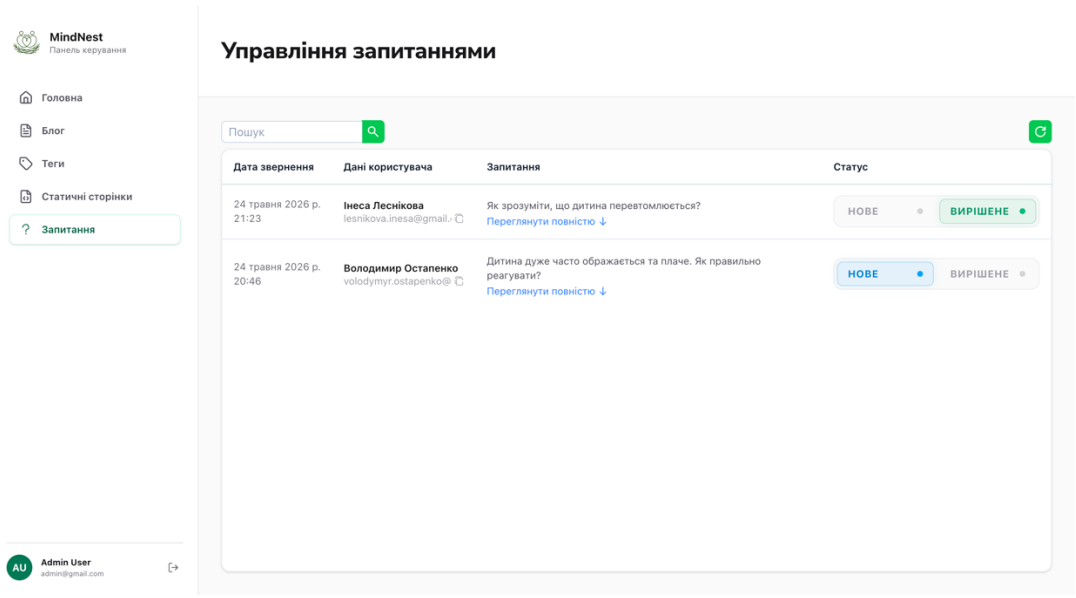

Рисунок 3.13 – Змінений статус повідомлення після відповіді

На рис. 3.14 показано, як користувач отримує зворотній зв'язок.

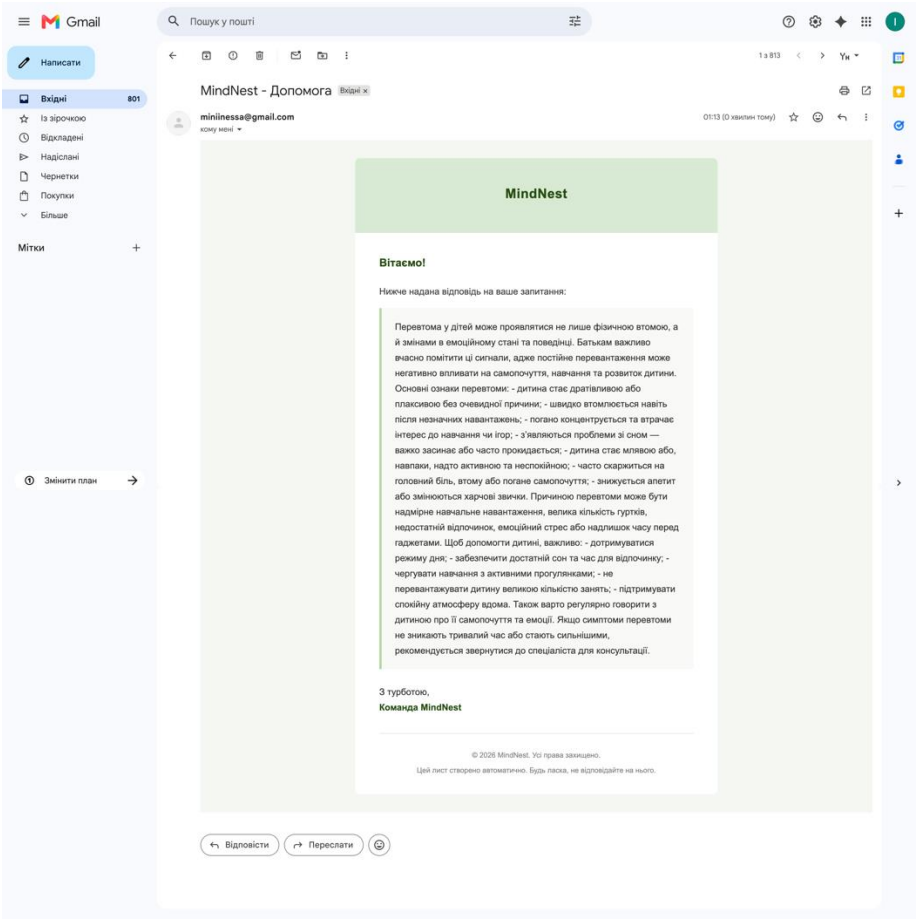

Рисунок 3.14 – Зворотній зв’язок користувачеві на пошті

Також існує можливість вручну змінювати статус запитання через адміністративну панель. Це дає адміністраторам більше контролю над процесом обробки запитів.

Ця система забезпечує плавний процес обробки запитань від користувачів. Вона автоматизує комунікацію, дозволяє адміністраторам ефективно керувати всіма зверненнями через зручний інтерфейс та гарантує, що жодне запитання не залишиться без уваги.

Для забезпечення своєчасного інформування користувачів про актуальний контент та оновлення платформи, було реалізовано функціонал електронної розсилки. Процес підписки ініціюється через модальне вікно, доступне на головній сторінці.

Користувач вводить свою електронну адресу та активує підписку натисканням відповідної кнопки. Перед відправкою даних на сервер, здійснюється

клієнтська валідація введеної інформації за допомогою бібліотеки Yup. Ця перевірка гарантує коректність формату email та наявність обов'язкового значення, надаючи зворотний зв'язок користувачеві без зайвих запитів до сервера.

Успішно валідовані дані надсилаються на сервер через API-ендпоінт /api/subscribe методом POST. Серверна логіка додатково верифікує отриману електронну адресу для забезпечення цілісності та безпеки даних.

Під час обробки запиту серверна частина перевіряє, чи існує в базі даних користувач із вказаною електронною адресою. Для цього використовується ORM Drizzle, що забезпечує зручну взаємодію з базою даних без необхідності написання складних SQL-запитів. Нижче на рис. 3.15 наведено лістинг перевірки існування підписки.

```
const existing = await db.select().from(subscriptions).where(eq(subscriptions.email, email)).limit(1)

if (existing.length > 0) {
  const sub = existing[0]
  if (sub.isActive) {
    return {
      success: false,
      message: 'Ви вже підписані на розсилку'
    }
  }
```

Рисунок 3.15 – Лістинг перевірки існування підписки

Наведений фрагмент коду демонструє механізм перевірки наявності користувача серед підписників. Якщо електронна адреса вже присутня в базі даних, система повертає відповідне повідомлення та не створює новий запис. Такий підхід дозволяє уникнути дублювання даних та забезпечує цілісність інформації.

Крім того, реалізований механізм повторної активації підписки. У випадку, якщо користувач раніше відмовився від розсилки, але вирішив знову отримувати інформаційні повідомлення, система не створює новий запис у базі даних. Натомість оновлюється значення поля активності підписки, що дозволяє зберігати історію взаємодії користувача з платформою та оптимізує структуру даних.

На сервері система перевіряє наявність користувача в таблиці підписників бази даних. Нижче у табл. 3.1 наведено сценарії обробки підписки на розсилку.

Таблиця 3.1 – Сценарії обробки підписки на розсилку

Умова	Процес у системі	Результат для користувача
Email відсутній у базі даних	Створюється новий запис у таблиці subscriptions	Відображається повідомлення про успішну підписку
Email уже існує та підписка активна	Новий запис не створюється	Відображається повідомлення «Ви вже підписані на розсилку»
Email існує, але підписка неактивна	Підписка повторно активується	Відображається повідомлення про поновлення підписки
Введено некоректний email	Запит відхиляється	Відображається повідомлення про помилку валідації
Виникла помилка сервера	Операція скасовується	Відображається повідомлення про помилку підписки

Використання декількох сценаріїв обробки запитів дозволяє забезпечити коректну роботу системи та уникнути створення дубльованих записів у базі даних. Крім того, реалізація механізму повторної активації підписки дає можливість користувачам відновити отримання інформаційних матеріалів без необхідності повторної реєстрації.

Після успішного виконання операції, користувач отримує сповіщення (toast-повідомлення) про результат без перезавантаження сторінки. Модальне вікно автоматично закривається, а форма очищується.

Структура бази даних передбачає зберігання електронної адреси та статусу активності підписки (isActive), що дозволяє ефективно управляти списком розсилки та реалізовувати механізми відновлення підписки без дублювання записів.

Нижче на рис. 3.16 та 3.17 наведені вигляд модулю підписки та повідомлення про вдалу підписку користувача.

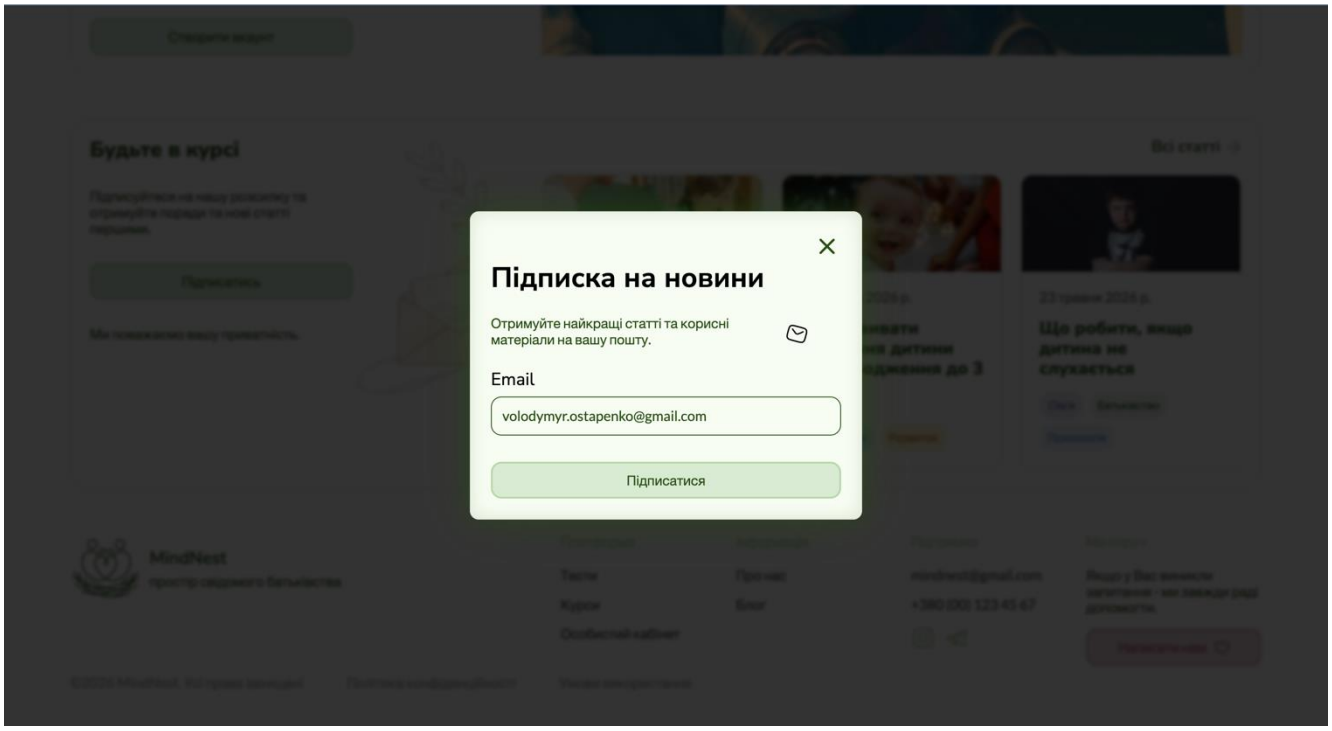

Рисунок 3.16 – Модуль підписки

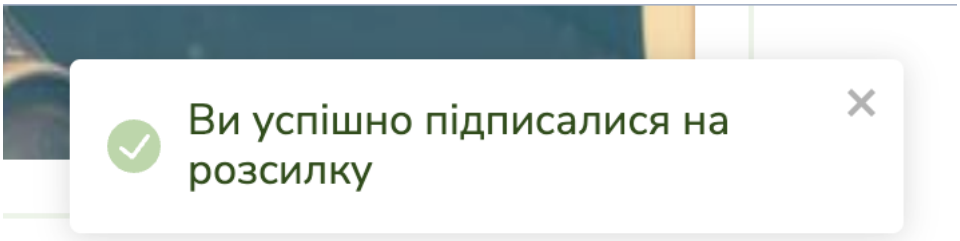

Рисунок 3.17 – Повідомлення про вдалу підписку

Ця реалізація забезпечує зручний процес підписки та підтримує актуальність бази даних для подальших автоматизованих розсилок.

3.4 Реалізація профілю користувача

Особистий кабінет є центральним елементом взаємодії користувача з платформою MindNest. Він забезпечує доступ до персональної інформації, результатів навчання, курсів, рекомендацій та повідомлень системи. Основною метою сторінки є надання користувачу можливості керувати власними даними та відстежувати прогрес проходження освітніх матеріалів.

Сторінка побудована за принципом двоколонкового інтерфейсу. Ліва частина містить навігаційне меню, а права — основний контент профілю.

Для покращення навігації використовується бокова панель, яка надає швидкий доступ до основних розділів особистого кабінету. У табл. 3.2 зазначено основні елементи профілю користувача

Таблиця 3.2 – Основні елементи профілю користувача

Елемент	Призначення
Профіль	Перегляд та редагування власних даних
Мої результати	Перегляд результатів курсів
Мої курси	Перегляд курсів
Мої рекомендації	Персональні рекомендації
Улюблене	Збережені курси
Повідомлення	Системні сповіщення

Навігаційне меню реалізовано за допомогою компонентів маршрутизації Nuxt, що дозволяє виконувати переходи між сторінками без повного перезавантаження вебзастосунку. На рис. 3.18 відображено фрагмент реалізації навігації профілю.

```
<nuxt-link to="/profile/courses" class="profile-form__sidebar--item" exact-active-class="active">
    <Icon name="hugeicons:course" size="28" style="..." />
    Мої курси
</nuxt-link>
```

Рисунок 3.18 - Фрагмент реалізації навігації профілю

Наведений код демонструє реалізацію одного з пунктів меню. При переході на відповідну сторінку автоматично застосовується стилізація активного пункту, що дозволяє користувачу легко визначити своє поточне місцезнаходження в системі.

У центральній частині сторінки відображається основна інформація про користувача. Тут містяться фотографія профілю, ім’я, прізвище, адреса електронної пошти та номер телефону.

Особливістю реалізації є можливість завантаження власної фотографії. Якщо фотографія відсутня, система автоматично формує аватар на основі перших літер імені та прізвища користувача.

У табл. 3.3 відображено поля форми редагування профілю користувача.

Таблиця 3.3 – Поля форми редагування профілю користувача

Поле	Опис
Ім’я	Ім’я користувача
Прізвище	Прізвище користувача
Пошта	Пошта користувача
Роль у батьківстві	Мама, тато, опікун або інше
Вік	Вік дитини у роках
Стать	Хлопчик чи дівчинка

Перед збереженням інформації виконується валідація введених даних, що дозволяє запобігти збереженню некоректних значень та забезпечує цілісність інформації в базі даних.

Завдяки використанню двостороннього зв'язку (v-model) зміни даних миттєво відображаються у стані застосунку та можуть бути передані на сервер для подальшого збереження.

Нижче на рис. 3.19 показано частина форми редагування профілю.

```
<AppInput class="personal"
    title="Ім’я"
    placeholder="Введіть ім’я"
    type="text"
    :error="errors.firstName"
    v-model="firstName"
  />
```

Рисунок 3.19 – Частина форми редагування профілю

Окремий блок профілю призначений для відображення прогресу користувача в навчанні. Тут відображається інформація про рівень проходження курсів, виконані завдання та результати тестування.

Наявність такого функціоналу дозволяє користувачам контролювати власний прогрес та мотивує до подальшого проходження навчальних матеріалів.

Крім того, у профілі передбачено блок підтримки користувача з мотиваційним повідомленням та ілюстрацією. Це відповідає загальній концепції платформи MindNest, яка спрямована не лише на навчання батьків, а й на створення комфортного та психологічно безпечного середовища для отримання нових знань.

Після внесення змін до форми профілю користувач може зберегти оновлену інформацію. Дані передаються на сервер через API-запит та записуються до бази даних. Перед збереженням виконується перевірка коректності введених значень, що дозволяє уникнути появи невалідних даних у системі.

Нижче на рис. 3.20 показано лістинг прикладу оновлення профілю.

```
await $fetch( request: '/api/user/update', opts: {
  method: 'PUT',
  body: values
})
```

Рисунок 3.20 – Лістинг оновлення профілю

Після успішного оновлення система відображає повідомлення про збереження змін без необхідності перезавантаження сторінки.

Оскільки профіль містить персональні дані користувача, доступ до сторінки можливий лише після проходження авторизації. Для перевірки прав доступу використовуються механізми автентифікації та авторизації, реалізовані на серверній стороні застосунку.

Під час звернення до API система перевіряє наявність активної сесії користувача. Якщо користувач не авторизований, доступ до персональних даних блокується та виконується перенаправлення на сторінку входу.

Розроблений профіль користувача забезпечує централізоване управління персональними даними та створює єдину точку взаємодії з платформою. Завдяки використанню компонентного підходу Vue.js інтерфейс легко масштабується та підтримується. Реалізована система валідації форм підвищує якість даних, що зберігаються в базі даних, а асинхронна взаємодія з сервером забезпечує швидкий відгук інтерфейсу та комфортний користувацький досвід.

3.5 Реалізація підключення телеграм-боту

Для підвищення зручності використання платформи MindNest реалізовано інтеграцію з месенджером Telegram. Використання Telegram-бота дозволяє забезпечити швидке інформування користувачів про нові матеріали, відповіді на звернення та інші події без необхідності постійного відвідування вебсайту.

Основним завданням бота є забезпечення додаткового каналу комунікації між платформою та користувачами. Реалізація виконана на серверній частині застосунку з використанням бібліотеки Telegraf для Node.js.

Для роботи бота використовується токен доступу, отриманий через сервіс BotFather. Нижче на рис. 3.21 продемонстровано ініціалізацію телеграм-боту.

```
import { Telegraf } from 'telegraf'

const token = process.env.TELEGRAM_TOKEN

const bot = new Telegraf(token)
```

Рисунок 3.21 – Ініціалізація телеграм-бота

Після запуску серверної частини створюється екземпляр бота, який починає прослуховувати вхідні повідомлення користувачів через механізм Long Polling.

Для забезпечення персоналізованих повідомлень кожен користувач може підключити власний Telegram-акаунт до профілю на платформі.

Ідентифікатор Telegram зберігається у таблиці користувачів бази даних.

У табл. 3.4 зазначені поля інтеграції Telegram

Таблиця 3.4 – Поля інтеграції Telegram

Поле	Призначення	Тип даних
telegram_id	Унікальний ідентифікатор користувача Telegram	VARCHAR
telegram_subscribed	Статус підписки на повідомлення	BOOLEAN

Збереження Telegram ID дозволяє системі надсилати повідомлення конкретному користувачу без необхідності додаткової авторизації в месенджері.

Нижче на рис. 3.22 наведено лістинг коду поля Telegram у схемі бази даних.

```
telegramId: varchar( name: "telegram_id",  config: { length: 500 }),
telegramSubscribed: boolean( name: "telegram_subscribed").default( value: true),
```

Рисунок 3.22 – Поля Telegram у схемі бази даних

Однією з реалізованих можливостей є підтримка авторизації через Telegram. Для цього використовується тимчасовий токен підтвердження, який генерується сервером та передається користувачеві.

Після переходу за спеціальним посиланням користувач надсилає команду боту, а система виконує зіставлення Telegram-акаунта з відповідним записом у базі даних.

Нижче наведено на рис. 3.23 збереження зв'язку між Telegram та користувачем.

```
global.telegramLoginMap = global.telegramLoginMap || new Map()

global.telegramLoginMap.set(token, user.id)
```

Рисунок 3.23 – Збереження зв'язку між Telegram та користувачем

Такий механізм дозволяє безпечно підтверджувати особу користувача без передачі паролів через Telegram.

Для керування повідомленнями реалізовано спеціальні команди Telegram-бота. У табл. 3.5 наведено команди, бо виконує бот.

Таблиця 3.5 – Команди Telegram-бота

Команда	Призначення
/start	Початок роботи з ботом
/subscribe	Увімкнення сповіщень
/unsubscribe	Вимкнення сповіщень
/link	Підключення Telegram до профілю

При виконанні команди підписки система оновлює відповідне поле в базі даних. Нижче на рис. 3.24 наведено код для активації підписки.

```
await db
    .update(users)
    .set({ telegramSubscribed: true })
    .where(eq(users.telegramId, chat.id.toString()))
```

Рисунок 3.24 – Активація підписки

У результаті користувач автоматично додається до списку отримувачів повідомлень.

Однією з ключових функцій Telegram-бота є автоматичне інформування користувачів про появу нових статей на платформі.

Після публікації нового матеріалу система виконує пошук усіх користувачів, які:

– мають підключений Telegram;

– активували підписку на повідомлення.

Далі формується список отримувачів та виконується масове надсилання повідомлень.

Інтеграція Telegram-бота значно розширює функціональні можливості платформи MindNest та забезпечує оперативний канал комунікації з батьками. Реалізований механізм підтримує авторизацію користувачів, прив'язку акаунтів, керування підписками, автоматичну розсилку нових матеріалів та надсилання відповідей на звернення. Завдяки цьому підвищується залученість користувачів до освітнього процесу та спрощується отримання актуальної інформації про нові можливості платформи.

3.6 Реалізація системи тестування

Одним із ключових компонентів платформи MindNest є система тестування, яка дозволяє визначити потреби користувача та сформувати персоналізовані рекомендації щодо навчальних матеріалів. Тестування виступає початковим етапом взаємодії користувача з платформою та забезпечує індивідуальний підхід до навчання.

Після реєстрації користувач проходить загальний тест, що складається з набору тематичних запитань. Основною метою тестування є визначення рівня обізнаності батьків щодо психоемоційного розвитку дитини, труднощів, які виникають під час виховання, та тем, які потребують додаткового опрацювання.

У табл. 3.6 наведено основні сутності системи тестування.

Таблиця 3.6 – Основні сутності системи тестування

Таблиця	Призначення
tests	Інформація про тест
questions	Запитання тесту
answers	Варіанти відповідей
user_answers	Відповіді користувачів
test_results	Результати тестування

Для зберігання тестів використовується декілька взаємопов'язаних таблиць бази даних. Такий підхід дозволяє легко створювати нові тести, редагувати запитання та отримувати результати користувачів.

Кожен тест містить набір запитань, а кожне запитання може мати декілька варіантів відповідей із визначеною кількістю балів.

Після відкриття сторінки тестування клієнтська частина отримує список запитань із серверної частини та формує інтерфейс проходження тесту. Отримані дані автоматично відображаються у вигляді послідовності запитань із варіантами відповідей. Користувач може обрати лише один або декілька варіантів залежно від типу запитання.

Під час проходження тесту всі відповіді тимчасово зберігаються у стані застосунку. Після завершення тестування формується об'єкт результатів, який передається на сервер для подальшої обробки. На серверній стороні виконується перевірка отриманих даних та запис відповідей до бази даних.

Після отримання відповідей система автоматично обчислює загальну кількість набраних балів. Для кожного варіанта відповіді попередньо визначається певна вага, яка враховується під час підрахунку результату.

Нижче на рис. 3.25 зазначено розрахунок та повернення значення пройденого тесту.

```
const score = Math.round( x: (correctCount / totalCount) * 100)
const passed = score >= (test.value.passingScore || 80)

return {
  passed,
  score,
  passingScore: test.value.passingScore || 80,
  mistakes
}
```

Рисунок 3.25 – розрахунок та повернення значення пройденого тесту

Отриманий результат використовується для визначення категорії користувача та подальшого формування рекомендацій.

Окрім загального тестування, у платформі реалізовано міні-тести всередині навчальних курсів. Їх основним завданням є перевірка засвоєння матеріалу після проходження окремого уроку або модуля. На відміну від початкового тестування, результати міні-тестів впливають на показник прогресу проходження курсу.

У табл. 3.7 зазначено логіку оновлення прогресу

Таблиця 3.7 – Логіка оновлення прогресу

Подія	Результат
Проходження уроку	Оновлення прогресу
Успішний міні-тест	Зарахування модуля
Завершення всіх модулів	Завершення курсу

Завдяки цьому користувач може контролювати власний прогрес та оцінювати рівень засвоєння навчального матеріалу.

Реалізована система тестування є основою механізму персоналізації платформи MindNest. Вона дозволяє визначати індивідуальні потреби батьків, автоматично підбирати відповідні навчальні матеріали та контролювати рівень засвоєння знань. Використання тестування на початковому етапі та міні-тестів у процесі навчання забезпечує адаптивний підхід до освітнього процесу та підвищує ефективність використання платформи.

3.7 Реалізація системи рекомендацій

Однією з ключових функціональних можливостей платформи MindNest є система рекомендацій, призначена для персоналізації навчального процесу. Її основним завданням є підбір найбільш релевантних курсів, статей та навчальних матеріалів відповідно до потреб конкретного користувача.

На відміну від традиційних освітніх платформ, де всі користувачі отримують однаковий набір матеріалів, у системі MindNest рекомендації формуються на основі індивідуальних результатів тестування та характеристик профілю користувача.

Після завершення загального тестування система аналізує отримані відповіді та визначає категорії тем, які потребують найбільшої уваги. На основі цих даних автоматично формується перелік рекомендованих курсів і статей.

Після проходження тесту кожна відповідь користувача отримує певну вагу. Накопичені бали дозволяють визначити напрямки, які потребують додаткового вивчення.

Наприклад, якщо користувач демонструє труднощі у питаннях емоційної підтримки дитини, система може рекомендувати курси, пов'язані з розвитком емоційного інтелекту та ефективною комунікацією між батьками та дітьми.

На серверній стороні після отримання результатів тестування виконується аналіз набраних балів та пошук відповідних навчальних матеріалів.

На рис. 3.26 зазначено лістинг коду формування рекомендацій

```
const totalScore = answers.reduce(
    (sum, answer) => sum + answer.score,
    0
)

let category

if (totalScore <= 30) {
    category = 'basic'
} else if (totalScore <= 60) {
    category = 'intermediate'
} else {
    category = 'advanced'
}

const recommendations = await db
    .select()
    .from(courses)
    .where(eq(courses.category, category))
```

Рисунок 3.26 – Формування рекомендацій

Після отримання списку матеріалів система зберігає рекомендації та відображає їх у профілі користувача.

Окрім результатів тестування, у майбутньому система може враховувати додаткові фактори:

– вік дитини;
– роль користувача (мати, батько, опікун);
– історію проходження курсів;
– результати міні-тестів;
– переглянуті статті;
– попередні рекомендації.

Використання декількох джерел даних дозволить підвищити точність підбору навчальних матеріалів.

Подальший розвиток платформи передбачає інтеграцію технологій штучного інтелекту для створення адаптивної системи рекомендацій.

На відміну від правил, заснованих на заздалегідь визначених умовах, алгоритми штучного інтелекту можуть аналізувати великі обсяги даних та самостійно знаходити закономірності в поведінці користувачів.

Потенційно система штучного інтелекту зможе:

– аналізувати результати тестувань;
– враховувати активність користувача на платформі;
– прогнозувати теми, які можуть бути корисними конкретному користувачу;
– формувати індивідуальні навчальні траєкторії;
– автоматично адаптувати рекомендації після кожного завершеного курсу.

Для реалізації таких можливостей можуть використовуватися сучасні моделі машинного навчання та великі мовні моделі (LLM), які здатні аналізувати текстову інформацію та формувати персоналізовані рекомендації на основі поведінкових даних користувачів.

Наприклад, користувач проходить початковий тест та отримує рекомендацію щодо курсу з емоційного розвитку дітей.

Після проходження курсу система аналізує:

– результати міні-тестів;
– тривалість перегляду матеріалів;

– теми, які викликали найбільший інтерес.

На основі цих даних модель штучного інтелекту може запропонувати наступний курс або додаткові статті, які найбільше відповідають потребам користувача.

Таким чином формується безперервний процес адаптивного навчання, у якому рекомендації постійно уточнюються відповідно до прогресу користувача.

Реалізована система рекомендацій забезпечує персоналізацію навчального процесу та дозволяє користувачам отримувати найбільш релевантні освітні матеріали відповідно до результатів тестування. Використання механізму автоматичного підбору курсів підвищує ефективність навчання та сприяє більш усвідомленому засвоєнню знань. Перспективним напрямом розвитку платформи є інтеграція технологій штучного інтелекту, що дозволить створити адаптивну систему рекомендацій, здатну враховувати індивідуальні особливості та поведінку кожного користувача.

Висновки до розділу 3

Було створено та вдосконалено інтерфейс користувача (UI/UX) платформи, використовуючи заспокійливу пастельну палітру. Цей вибір кольорів враховує психологію сприйняття, зменшує навантаження на увагу та знижує рівень тривожності користувачів. Розроблено зручні макети для ключових сторінок: Головної, Блогу, статей, інформаційних розділів та Особистого кабінету.

У процесі розробки було створено сучасний адаптивний інтерфейс із використанням компонентного підходу фреймворку Vue.js та можливостей Nuxt.js. Розроблено сторінки авторизації та реєстрації користувачів, особистий кабінет, систему навігації, сторінки навчальних курсів, рекомендацій, статей та адміністративну панель.

Особливу увагу приділено реалізації системи тестування, яка є основою персоналізації навчального процесу. Створено механізм проходження тестів, обробки відповідей користувачів, автоматичного обчислення результатів та

формування індивідуальних рекомендацій відповідно до отриманих показників. Додатково реалізовано міні-тести всередині навчальних курсів для контролю засвоєння навчального матеріалу та відстеження прогресу користувачів.

У межах реалізації комунікаційних можливостей платформи створено систему зворотного зв'язку, що дозволяє користувачам надсилати запитання адміністраторам через спеціальну форму. Передбачено автоматичне збереження звернень у базі даних, їх опрацювання через адміністративну панель та надсилання відповідей користувачам електронною поштою зі зміною статусу звернення після його виконання.

Також реалізовано систему підписки на інформаційну розсилку, яка забезпечує збір електронних адрес користувачів, перевірку коректності введених даних, уникнення дублювання записів та можливість повторної активації підписки.

Для розширення можливостей комунікації інтегровано Telegram-бота, який дозволяє надсилати повідомлення користувачам, інформувати їх про нові матеріали платформи та використовувати месенджер як додатковий канал взаємодії між системою та користувачами.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено онлайн-платформу MindNest, призначену для навчання батьків з питань психоемоційного розвитку дітей. Створена система спрямована на підвищення рівня обізнаності батьків щодо сучасних підходів до виховання, розвитку емоційного інтелекту дитини та формування безпечного сімейного середовища.

У процесі виконання роботи було проведено аналіз предметної області, досліджено особливості психоемоційного розвитку дітей та визначено основні потреби цільової аудиторії. На основі отриманих результатів сформовано функціональні вимоги до системи та спроєктовано її архітектуру.

Було розроблено структуру бази даних, ER-діаграму та модель взаємодії користувача із системою. Для реалізації проєкту обрано сучасний технологічний стек, що включає Nuxt.js, Vue.js, Node.js, Drizzle ORM та MySQL. Використання гібридного підходу SPA та SSR дозволило забезпечити як високу швидкодію інтерфейсу, так і ефективну індексацію публічного контенту пошуковими системами.

У межах практичної реалізації створено систему реєстрації та авторизації користувачів, особистий кабінет, механізм тестування та формування персоналізованих рекомендацій, навчальні курси з перевіркою знань, систему зворотного зв'язку, підписку на розсилку новин, адміністративну панель для керування контентом, а також інтеграцію з Telegram-ботом для додаткового інформування користувачів.

Особливістю розробленої платформи є використання механізму персоналізації навчання. На основі результатів тестування система автоматично підбирає рекомендовані матеріали та курси, що дозволяє враховувати індивідуальні потреби кожного користувача. Такий підхід підвищує ефективність засвоєння навчального контенту та сприяє більш усвідомленому батьківству.

Розроблена система відповідає поставленим цілям і завданням дипломної роботи, є функціонально завершеним програмним продуктом та може бути

використана як основа для подальшого розвитку сервісів підтримки батьків. Перспективами подальшого розвитку проєкту є впровадження інтелектуальних алгоритмів аналізу поведінки користувачів, використання технологій штучного інтелекту для генерації персоналізованих рекомендацій, розширення функціональності Telegram-бота та створення мобільного застосунку для платформи MindNest.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Психоемоційний розвиток дитини: ключові етапи та роль батьків. URL: https://nutricia.ua/psyhoemoczijnyj-rozvytok-dytyny-vid-narodzhennya-klyuchovi-etapy-ta-rol-batkiv/ (дата звернення: 20.04.2026)

2. Everyday Parenting: The ABCs of Child Rearing : онлайн-курс / Yale University, Coursera. URL: Coursera — Everyday Parenting: The ABCs of Child Rearing (дата звернення: 22.04.2026)

3. Bebbo : мобільний застосунок для підтримки батьківства. URL: Bebbo Parenting App (дата звернення: 22.04.2026)

4. Vroom : мобільний застосунок для раннього розвитку дітей та підтримки батьківства. URL: https://www.vroom.org/ (дата звернення: 22.04.2026)

5. Nuxt.js – офіційна документація. URL: https://nuxt.com/ (дата звернення: 23.04.2026)

6. Nuxt.js – технологія розробки веб-застосунків. URL: https://brander.ua/technologies/nuxtjs (дата звернення: 23.04.2026)

7. Nuxt.js : вебсайт. URL: Brander — Nuxt.js (дата звернення: 23.04.2026)

8. Vue.js — офіційна документація. URL: https://vuejs.org/ (дата звернення: 23.04.2026)

9. Drizzle ORM : вебсайт. URL: npm — drizzle-orm (дата звернення: 28.04.2026)

10. MySQL : вебсайт. URL: Wikipedia — MySQL (дата звернення: 01.05.2026)

11. Що таке Node.js простими словами : вебсайт. URL: DAN.IT Education (дата звернення: 03.05.2026)

12. Обговорення SPA та SSR. URL: https://dou.ua/forums/topic/41585/ (дата звернення: 07.04.2026)

13. SSR чи SPA: що вибрати для вебсайту : вебсайт. URL: Tproger (дата звернення: 10.05.2026)

14. SPA чи SSR застосунки : вебсайт. URL: DOU.ua (дата звернення: 10.05.2026)

15. Метод декомпозиції. URL: https://galaktica.io/blog/dekompozyciya/ (дата звернення: 10.05.2026)

16. MDN Web Docs – ресурс з веб-розробки. URL: https://developer.mozilla.org/ (дата звернення: 10.05.2026)

17. Essentials for Parenting : освітній ресурс для батьків та опікунів / Centers for Disease Control and Prevention. URL: https://www.cdc.gov/parents/essentials/ (дата звернення: 15.05.2026).

18. Triple P – Positive Parenting Program : програма позитивного батьківства. URL: https://www.triplep-parenting.com/ (дата звернення: 24.05.2026). Рекомендована як одна з найпоширеніших доказових програм батьківської освіти.

19. Incredible Years : програма розвитку батьківських навичок та соціально-емоційного розвитку дітей. URL: https://www.incredibleyears.com/ (дата звернення: 24.05.2026).

20. Parent-Child Interaction Therapy (PCIT) : програма підтримки взаємодії батьків і дітей. URL: https://www.pcit.org/ (дата звернення: 24.05.2026).

21. Essentials for Parenting Toddlers and Preschoolers : освітній ресурс для батьків дітей віком від 2 до 4 років / Centers for Disease Control and Prevention. URL: https://www.cdc.gov/parenting-toddlers/about/ (дата звернення: 24.05.2026).

22. Resources for Child Development : інформаційні матеріали щодо розвитку дітей / Centers for Disease Control and Prevention. URL: https://www.cdc.gov/child-development/resources/ (дата звернення: 24.05.2026).

23. Siegel D., Bryson T. The Whole-Brain Child. New York : Delacorte Press, 2011. 192 p.

24. Haverbeke M. Eloquent JavaScript. 3rd ed. San Francisco : No Starch Press, 2018. 472 p.

25. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 704 p.

26. Brown E. Learning Vue.js 2. Birmingham : Packt Publishing, 2018. 420 p.

27. Oliveira S. Nuxt.js Web Development. Birmingham : Packt Publishing, 2018. 310 p.

28. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : Wiley, 2014. 490 p.

29. Freeman A. Pro Node.js for Developers. New York : Apress, 2016. 760 p.

30. Бех І. Д. Особистісно орієнтоване виховання : навч. посіб. Київ : Либідь, 2003. 280 с.

31. Савчин М. В. Вікова психологія : навч. посіб. Київ : Академвидав, 2011. 384 с.

32. Максименко С. Д. Загальна психологія : підручник. Київ : Центр учбової літератури, 2018. 272 с.

33. Поніманська Т. І. Дошкільна педагогіка : підручник. Київ : Академвидав, 2015. 448 с.

34. Bowlby J. Attachment and Loss. New York : Basic Books, 1982. 425 p.

ДОДАТОК А
Схема баз даних

```
import {
    mysqlTable,
    varchar,
    text,
    datetime,
    int,
    boolean,
    mysqlEnum,
} from "drizzle-orm/mysql-core";
import { sql } from "drizzle-orm";

export const users = mysqlTable("users", {
    id: int("id").autoincrement().primaryKey(),
    firstName: varchar("first_name", { length: 255 }).notNull(),
    secondName: varchar("second_name", { length: 255 }).notNull(),
    phone: varchar("phone", { length: 50 }),
    email: varchar("email", { length: 255 }).notNull().unique(),
    password: varchar("password", { length: 255 }).notNull(),
    photoUrl: varchar("photo_url", { length: 500 }),
    telegramId: varchar("telegram_id", { length: 500 }),
    telegramSubscribed: boolean("telegram_subscribed").default(true),
    role: mysqlEnum("role", ["user", "admin", "superadmin"]).default("user"),
    parentingRole: mysqlEnum("parenting_role", ["Мама", "Тато", "Опікун",
"Інше"]),
    childAge: int("child_age"),
    childSex: mysqlEnum("child_sex", ["girl", "boy"]),
    hasSeenOnboarding: boolean("hasSeenOnboarding").default(false),
    resetToken: varchar("reset_token", { length: 255 }),
    resetTokenExpires: datetime("reset_token_expires"),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const posts = mysqlTable("posts", {
    id: int("id").autoincrement().primaryKey(),
    title: varchar("title", { length: 255 }).notNull(),
    slug: varchar("slug", { length: 255 }).notNull().unique(),
    content: text("content").notNull(),
    coverImage: varchar("cover_image", { length: 500 }),
    published: boolean("published").default(false),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
    updatedAt: datetime("updated_at").default(
        sql`CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP`,
    ),
});

export const tags = mysqlTable("tags", {
    id: int("id").autoincrement().primaryKey(),
    title: varchar("title", { length: 255 }).notNull(),
    slug: varchar("slug", { length: 255 }).notNull().unique(),
    color: varchar("color", { length: 7 }).notNull(),
    textColor: varchar("text_color", { length: 7 }).notNull(),
});

export const postTags = mysqlTable("post_tags", {
    postId: int("post_id")
        .notNull()
        .references(() => posts.id, { onDelete: "cascade" }),
    tagId: int("tag_id")
        .notNull()
```

```
        .references(() => tags.id, { onDelete: "cascade" }),
});

export const staticPages = mysqlTable("static_pages", {
    id: int("id").autoincrement().primaryKey(),
    title: varchar("title", { length: 255 }).notNull(),
    slug: varchar("slug", { length: 255 }).notNull().unique(),
    content: text("content").notNull(),
    pageType: mysqlEnum("page_type", ["simple",
"json"]).default("simple").notNull(),
    updatedAt: datetime("updated_at").default(
        sql`CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP`,
    ),
});

export const courses = mysqlTable("courses", {
    id: int("id").autoincrement().primaryKey(),
    title: varchar("title", { length: 255 }).notNull(),
    description: text("description"),
    imageUrl: varchar("image_url", { length: 500 }),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const courseTags = mysqlTable("course_tags", {
    courseId: int("course_id")
        .notNull()
        .references(() => courses.id, { onDelete: "cascade" }),
    tagId: int("tag_id")
        .notNull()
        .references(() => tags.id, { onDelete: "cascade" }),
});

export const blocks = mysqlTable("blocks", {
    id: int("id").autoincrement().primaryKey(),
    courseId: int("course_id")
        .notNull()
        .references(() => courses.id, { onDelete: "cascade" }),
    title: varchar("title", { length: 255 }).notNull(),
    textContent: text("text_content"),
    videoUrl: varchar("video_url", { length: 512 }),
    mainImageUrl: varchar("main_image_url", { length: 512 }),
    sortOrder: int("sort_order").default(0),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const tests = mysqlTable("tests", {
    id: int("id").autoincrement().primaryKey(),
    blockId: int("block_id")
        .notNull()
        .unique()
        .references(() => blocks.id, { onDelete: "cascade" }),
    title: varchar("title", { length: 255 }).notNull(),
    passingScore: int("passing_score").default(80),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const testQuestions = mysqlTable("test_questions", {
    id: int("id").autoincrement().primaryKey(),
    testId: int("test_id")
        .notNull()
        .references(() => tests.id, { onDelete: "cascade" }),
    questionText: text("question_text").notNull(),
```

```
    type: mysqlEnum("type", ["single", "multiple", "text"]).notNull(),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const questionOptions = mysqlTable("question_options", {
    id: int("id").autoincrement().primaryKey(),
    questionId: int("question_id")
        .notNull()
        .references(() => testQuestions.id, { onDelete: "cascade" }),
    optionText: text("option_text").notNull(),
    isCorrect: boolean("is_correct").default(false),
});

export const questions = mysqlTable("questions", {
    id: int("id").autoincrement().primaryKey(),
    userId: int("user_id").references(() => users.id, { onDelete: "set null" }),
    name: varchar("name", { length: 255 }).notNull(),
    email: varchar("email", { length: 255 }).notNull(),
    content: text("content").notNull(),
    status: mysqlEnum("status", ["new", "resolved"]).default("new"),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const subscriptions = mysqlTable("subscriptions", {
    id: int("id").autoincrement().primaryKey(),
    email: varchar("email", { length: 255 }).notNull().unique(),
    isActive: boolean("is_active").default(true),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const notifications = mysqlTable("notifications", {
    id: int("id").autoincrement().primaryKey(),
    userId: int("user_id")
        .notNull()
        .references(() => users.id, { onDelete: "cascade" }),
    title: varchar("title", { length: 255 }).notNull(),
    description: text("description").notNull(),
  seen: boolean("seen").default(false),
    type: mysqlEnum("type", ["blogpost", "support"]),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});

export const settings = mysqlTable("settings", {
    id: int("id").autoincrement().primaryKey(),
    email: varchar("email", { length: 255 }).notNull(),
    phone: varchar("phone", { length: 255 }).notNull(),
    instagram: varchar("instagram", { length: 255 }),
    telegram: varchar("telegram", { length: 255 }),
});

export const favorites = mysqlTable("favorites", {
    id: int("id").autoincrement().primaryKey(),
    userId: int("user_id")
        .notNull()
        .references(() => users.id, { onDelete: "cascade" }),
    courseId: int("course_id")
        .notNull()
        .references(() => courses.id, { onDelete: "cascade" }),
    createdAt: datetime("created_at").default(sql`CURRENT_TIMESTAMP`),
});
```

ДОДАТОК Б
Підключення Телеграм-боту до платформи

```
import { Telegraf } from 'telegraf'
import { db } from '../db/index.js'
import { users } from '../db/schema.js'
import { eq } from 'drizzle-orm'

export default defineNitroPlugin((nitroApp) => {
  const token = process.env.TELEGRAM_TOKEN;

  if (!token) {
    console.warn('Telegram Bot Token is missing. Bot will not start.')
    return
  }

  const bot = new Telegraf(token)

  nitroApp.telegramBot = bot

  console.log('Bot started')

  bot.start(async (ctx) => {
    const userIdStr = ctx.payload
    const chat = ctx.chat

    if (userIdStr) {
      if (userIdStr.startsWith('register_')) {
        const token = userIdStr.replace('register_', '')
        const telegramChatId = chat.id.toString()

        try {
          let [user] = await db.select().from(users).where(eq(users.telegramId,
telegramChatId)).limit(1)

          if (!user) {
            const firstName = chat.first_name || 'Користувач'
            const secondName = chat.last_name || 'Telegram'
            let email = `tg_${telegramChatId}@mindnest.local`

            const [emailExists] = await
db.select().from(users).where(eq(users.email, email)).limit(1)
            if (emailExists) {
              email =
`tg_${telegramChatId}_${Math.random().toString(36).substring(2,
6)}@mindnest.local`
            }

            const bcrypt = await import('bcrypt')
            const hashedPassword = await
bcrypt.default.hash(Math.random().toString(36), 10)

            const [result] = await db.insert(users).values({
              firstName,
              secondName,
              email,
              password: hashedPassword,
              telegramId: telegramChatId,
              telegramSubscribed: true,
              role: 'user'
```

```
            })

            const [newUser] = await db.select().from(users).where(eq(users.id,
result.insertId)).limit(1)
            user = newUser
          }

          global.telegramLoginMap = global.telegramLoginMap || new Map()
          global.telegramLoginMap.set(token, user.id)

          setTimeout(() => {
            if (global.telegramLoginMap) {
              global.telegramLoginMap.delete(token)
            }
          }, 5 * 60 * 1000)

          return ctx.reply(`□ Реєстрація/Вхід для ${user.firstName} успішна!
Поверніться, будь ласка, у вікно браузера для завершення.`)
        } catch (err) {
          console.error('Telegram registration handler error:', err)
          return ctx.reply('❌ Помилка при спробі реєстрації через Telegram.')
        }
      }

      if (userIdStr.startsWith('login_')) {
        const token = userIdStr.replace('login_', '')
        const telegramChatId = chat.id.toString()

        try {
          const [user] = await db.select().from(users).where(eq(users.telegramId,
telegramChatId))
          if (!user) {
            return ctx.reply('❌ Користувача з таким Telegram не знайдено на
MindNest. Будь ласка, спочатку увійдіть звичайним способом та підключіть Telegram
у профілі.')
          }

          global.telegramLoginMap = global.telegramLoginMap || new Map()
          global.telegramLoginMap.set(token, user.id)

          setTimeout(() => {
            if (global.telegramLoginMap) {
              global.telegramLoginMap.delete(token)
            }
          }, 5 * 60 * 1000)

          return ctx.reply(`□ Запит на вхід для користувача ${user.firstName}
отримано! Поверніться, будь ласка, у вікно браузера для завершення входу.`)
        } catch (err) {
          console.error('Telegram login handler error:', err)
          return ctx.reply('❌ Помилка при спробі авторизації через Telegram.')
        }
      }

      const userId = parseInt(userIdStr, 10)
      if (isNaN(userId)) {
        return ctx.reply('Некоректний ідентифікатор користувача.')
      }

      try {
        const [existingTgUser] = await db
          .select()
```

```
          .from(users)
          .where(eq(users.telegramId, chat.id.toString()))
          .limit(1)

        if (existingTgUser) {
          if (existingTgUser.id === userId) {
            return ctx.reply('□ Ваш Telegram вже підключений до цього профілю
MindNest.')
          }
          return ctx.reply(`❌ Цей Telegram акаунт уже підключено до іншого профілю
MindNest (${existingTgUser.email}). Будь ласка, спочатку відключіть його на іншому
акаунті.`)
        }

        const [user] = await db.select().from(users).where(eq(users.id, userId))
        if (!user) {
          return ctx.reply('Користувача не знайдено на сайті MindNest.')
        }

        await db.update(users)
          .set({
            telegramId: chat.id.toString(),
            telegramSubscribed: true
          })
          .where(eq(users.id, userId))

        return ctx.reply(`□ Вітаємо, ${user.firstName}! Ваш акаунт MindNest
успішно підключено до Telegram сповіщень. Ви будете отримувати наші корисні поради
та новини.`)
      } catch (err) {
        console.error('Error linking telegram account:', err)
        return ctx.reply('Помилка при підключенні акаунту. Спробуйте пізніше або
зверніться до підтримки.')
      }
    }

    return ctx.reply('Вітаємо! Я бот MindNest. Щоб підключити сповіщення, будь
ласка, скористайтеся кнопкою у вашому профілі на сайті.')
  })

  bot.command('subscribe', async (ctx) => {
    const chat = ctx.chat
    try {
      const [user] = await db.select().from(users).where(eq(users.telegramId,
chat.id.toString()))
      if (!user) {
        return ctx.reply('Ваш Telegram не підключений до жодного профілю MindNest.
Будь ласка, підключіть його у налаштуваннях профілю на сайті.')
      }

      await db.update(users)
        .set({ telegramSubscribed: true })
        .where(eq(users.telegramId, chat.id.toString()))

      return ctx.reply('□ Ви успішно підписалися на новини та сповіщення від
MindNest!')
    } catch (err) {
      console.error('Error in subscribe command:', err)
      return ctx.reply('Виникла помилка. Спробуйте пізніше.')
    }
  })
```

```
  bot.command('unsubscribe', async (ctx) => {
    const chat = ctx.chat
    try {
      const [user] = await db.select().from(users).where(eq(users.telegramId,
chat.id.toString()))
      if (!user) {
        return ctx.reply('Ваш Telegram не підключений до жодного профілю
MindNest.')
      }

      await db.update(users)
        .set({ telegramSubscribed: false })
        .where(eq(users.telegramId, chat.id.toString()))

      return ctx.reply('□ Ви відписалися від новин та сповіщень. Ви можете
підписатися знову в будь-який момент за допомогою команди /subscribe.')
    } catch (err) {
      console.error('Error in unsubscribe command:', err)
      return ctx.reply('Виникла помилка. Спробуйте пізніше.')
    }
  })

  bot.on('message', (ctx) => {
    if ('text' in ctx.message) {
      const text = ctx.message.text
      ctx.reply(`You said: ${text}`)
    }
  })

  bot.launch()
    .then(() => {
      console.log('Telegram bot started successfully via Long Polling.')
    })
    .catch((err) => {
      console.error('Error running Telegram bot polling:', err)
    })

  nitroApp.hooks.hook('close', () => {
    console.log('Stopping Telegram bot...')
    bot.stop('Nuxt server closed')
  })
})
```

ДОДАТОК В
Перевірка авторизації та прав доступу користувачів

```
export default defineEventHandler(async (event) => {
  const isApiAdmin = event.path.startsWith('/api/admin/')
  const isLogin = event.path.startsWith('/api/admin/login')

  if (isApiAdmin && !isLogin) {
    const session = await getUserSession(event)
    const path = event.path.split('?')[0]

    const isGetCoursesList = event.method === 'GET' && path ===
'/api/admin/courses'
    const isGetCourseDetail = event.method === 'GET' &&
/^\/api\/admin\/courses\/\d+$/.test(path)
    const isGetBlocks = event.method === 'GET' && path === '/api/admin/blocks'
    const isGetBlockDetail = event.method === 'GET' &&
/^\/api\/admin\/blocks\/\d+$/.test(path)
    const isGetTests = event.method === 'GET' && path === '/api/admin/tests'
    const isAllowedForUser = isGetCoursesList || isGetCourseDetail || isGetBlocks
|| isGetBlockDetail || isGetTests

    if (isAllowedForUser) {
      if (!session.user) {
        throw createError({
          statusCode: 401,
          statusMessage: 'Неавторизований доступ'
        })
      }
    } else {
      if (!session.user || (session.user.role !== 'admin' && session.user.role !==
'superadmin')) {
        throw createError({
          statusCode: 401,
          statusMessage: 'Неавторизований доступ'
        })
      }
    }
  }
})
```