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

_____Юрій КОНДРАТЕНКО

«____»_____ 20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ ТЕНІСНОГО КЛУБУ НА ОСНОВІ ФРЕЙМВОРКА REACT

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Роман АНДРІЮК

«___»_____ 20__ р.

Керівник ст. викладач кафедри КІ _____ Катерина ОБУХОВА

«___»_____ 20__ р.

Миколаїв – 2025

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

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Андріюка Романа Анатолійовича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Вебзастосунок для тенісного клубу на основі фреймворка React».

Керівник роботи: Обухова Катерина Олександрівна ст. викладач кафедри КІ .

(прізвище, ім'я, по батькові, посада, науковий ступінь, вчене звання)

Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2025 р. № 321.

2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні:

У результаті роботи планується створити зручний сайт для тенісного клубу, де користувачі зможуть реєструватися, переглядати свій розклад, записуватись на турніри та бронювати корти. На початку були визначені ролі користувачів (учень,

тренер, батько, аматор) та їхні основні потреби для реалізації відповідного функціоналу.

4. Перелік питань, що підлягають розробці:

- провести огляд предметної сфери та проаналізувати аналогічні вебсайти;
- сформулювати вимоги до вебсайту тенісного клубу;
- розробити структуру сайту з урахуванням різних ролей для користувачів;
- створити дизайн, який буде адаптований до різних типів пристроїв;
- реалізувати вебзастосунок та його повний функціонал (напр., логіку реєстрації, бронювання, турнірів тощо);
- протестувати вебзастосунок на різних категоріях користувачів.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Катерина ОБУХОВА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Роман АНДРІЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «20» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Вебзастосунок для тенісного клубу на основі фреймворка React.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.10.2024	20.10.2024	виконано
2	Аналіз предметної області та постановка завдання	21.10.2024	30.11.2025	виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема вебзастосунок для тенісного клубу на основі фреймворка React	31.01.2025	01.02.2025	виконано
4	Огляд існуючих вебсайтів тенісних клубів	02.02.2025	21.02.2025	виконано
5	Розробка клієнтської частини вебзастосунку з використанням React	22.02.2025	15.03.2025	виконано
6	Створення серверної частини вебзастосунку	15.03.2025	15.04.2025	виконано
7	Тестування функціоналу та виправлення помилок	15.04.2025	15.05.2025	виконано
	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	виконано
7	Корегування роботи за результатами попереднього захисту	17.05.2025	29.05.2025	виконано
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	виконано
9	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	17.06.2025	виконано

Керівник роботи _____ Катерина ОБУХОВА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Роман АНДРІЮК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану «30» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 402 ЧНУ ім. Петра Могили

Андріюка Романа Анатолійовича

на тему: **«ВЕБЗАСТОСУНОК ДЛЯ ТЕНІСНОГО КЛУБУ НА ОСНОВІ ФРЕЙМВОРКА REACT»**

Актуальність даної роботи полягає у необхідності впровадження сучасних цифрових рішень для оптимізації взаємодії між тенісним клубом та його клієнтами, що сприятиме підвищенню ефективності обслуговування та розвитку клубу.

Об'єктом роботи є процес організації та автоматизації взаємодії користувачів із вебсистемою тенісного клубу.

Предмет роботи охоплює структуру, функціонал і засоби реалізації вебзастосунку для управління діяльністю тенісного клубу.

Мета роботи – розробити зручний, ефективний та високопродуктивний вебзастосунок для тенісного клубу, що відповідає сучасним вимогам цифрового обслуговування спортивних установ та забезпечує автоматизацію основних процесів взаємодії з користувачами.

У роботі використано методи аналізу, об'єктно-орієнтованого моделювання, проєктування, програмної реалізації, модульного тестування.

Дана робота складається з чотирьох розділів. У першому розділі проведено огляд предметної області, аналогів і постановку завдання. Другий розділ присвячено структуруванню інтерфейсу, дизайну, ролям користувачів і проєктуванню бази даних. У третьому розділі побудовано UML-діаграми, що відображають логіку та архітектуру системи. Четвертий розділ містить реалізацію клієнтської та серверної частин, інтеграцію модулів і результати тестування.

Розроблено вебзастосунок, побудований на технологіях React, Node.js, Express та MongoDB. Реалізовано повну рольову модель з авторизацією, персоналізованим

контентом, реєстрацією на події, бронюванням і редагуванням контенту. Система пройшла тестування з реальними користувачами. Вебзастосунок рекомендовано до адаптації та впровадження в роботу реального тенісного клубу.

Загальний обсяг роботи – 103 сторінки. Кваліфікаційна робота містить 2 додатки, 59 рисунків, 1 таблицю і 29 джерела посилання.

Ключові слова: вебзастосунок, тенісний клуб, інформаційна система, React, MongoDB.

ABSTRACT

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

Andriiuk Roman

"WEB APPLICATION FOR A TENNIS CLUB BASED ON THE REACT FRAMEWORK"

The relevance of this study lies in the need to implement modern digital solutions to optimize interaction between a tennis club and its clients, which contributes to improving service efficiency and supporting the club's development.

The object of the study is the process of organizing and automating user interaction with the tennis club's web system.

The subject of the study includes the structure, functionality, and implementation tools of a web application for managing the activities of a tennis club.

The purpose of the study is to develop a user-friendly, efficient, and high-performance web application for a tennis club that meets the modern requirements of digital service in sports institutions and enables automation of key user interaction processes.

The study applies methods of analysis, object-oriented modeling, system design, software implementation, and modular testing.

This qualification paper consists of four chapters. The first chapter provides an overview of the subject area, similar systems, and problem definition. The second chapter focuses on interface structuring, design, user roles, and database design. The third chapter presents UML diagrams reflecting the system's logic and architecture. The fourth chapter contains the implementation of both the client and server parts, module integration, and testing results.

A web application has been developed using React, Node.js, Express, and MongoDB technologies. A full role-based access model has been implemented, including authorization,

personalized content, event registration, court booking, and content editing. The system has been tested with real users. The web application is recommended for adaptation and deployment in a real tennis club.

Total volume of the work – 103 pages. The qualification paper includes 2 appendices, 59 figures, 1 table, and 29 references.

Keywords: web application, tennis club, information system, React, MongoDB.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІІС – інтелектуальні інформаційні системи

КБР – кваліфікаційна бакалаврська робота

УТЦ – український тенісний центр

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

UML – Unified Modeling Language

ВСТУП

В наші дні Інтернет став важливою частиною сучасного життя, а вебзастосунки – головним інструментом в руках людини для надання інформаційних, комунікаційних та сервісних послуг. У цифровому середовищі присутність вебсайту є основоположним елементом для ефективної взаємодії з користувачами та надання якісного сервісу. Наявність власної інформаційної системи дозволяє організаціям не лише представити себе в онлайн-просторі, а й спростити внутрішні процеси – від реєстрації користувачів до обробки запитів та керування контентом.

Особливо це актуально для спортивних клубів, які потребують зручного інструменту для взаємодії з клієнтами – від онлайн-бронювання до оголошень про заходи. Розробка інформаційної системи для тенісного клубу дозволяє автоматизувати значну частину рутинних процесів, підвищити зручність для користувачів і забезпечити сучасну форму представлення клубу в онлайн-середовищі. Вебзастосунок, створений у межах цієї роботи, підтримує рольову модель доступу, включає особисті кабінети для різних категорій користувачів, динамічні вкладки, адаптивний дизайн і гнучку систему управління розкладом та бронюваннями.

Об’єкт роботи – це процес організації та автоматизації взаємодії користувачів із вебсистемою тенісного клубу.

Предмет роботи – структура, функціонал і засоби реалізації вебзастосунку для управління діяльністю тенісного клубу.

Мета роботи – розробити зручний, ефективний та високопродуктивний вебзастосунок для тенісного клубу, що відповідає сучасним вимогам цифрового обслуговування спортивних установ та забезпечує автоматизацію основних процесів взаємодії з користувачами.

Завдання дослідження полягає у виконанні комплексу дій, необхідних для досягнення поставленої мети роботи, зокрема:

- провести огляд предметної сфери та проаналізувати аналогічні вебсайти;

– сформулювати вимоги до вебсайту тенісного клубу;

– розробити структуру сайту з урахуванням різних ролей для користувачів;

– створити дизайн, який буде адаптований до різних типів пристроїв;

– реалізувати вебсайт та його повний функціонал (напр., логіку реєстрації, бронювання, турнірів тощо);

– протестувати вебзастосунок на різних категоріях користувачів.

Методи дослідження: аналіз, проєктування, моделювання, об'єктно-орієнтоване програмування, тестування.

Інструменти, використані при розробці системи: React, Node.js, Express, MongoDB, HTML/CSS.

Практичне значення: розроблений вебзастосунок буде використаний в майбутньому як основна інформаційна система для реального тенісного клубу. Після незначної адаптації до потреб конкретного закладу сайт дозволить автоматизувати процеси реєстрації, управління розкладом, організації турнірів та бронювання кортів, що значно покращить зручність обслуговування клієнтів і внутрішню координацію в клубі.

Апробація: матеріали кваліфікаційної бакалаврської роботи (КБР) було представлено на XXVII Всеукраїнській науково-практичній конференції «Могилянські читання – 2024: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (підсекція «Інтелектуальні інформаційні системи», м. Миколаїв, 6–10 листопада 2024 р.) [11].

1 АНАЛІЗ ОСОБЛИВОСТЕЙ ФУНКЦІОНУВАННЯ ВЕБСИСТЕМИ ТЕНІСНОГО КЛУБУ

1.1 Актуальність обраної теми

В умовах динамічного розвитку та трансформації цифрових технологій їх інтеграція в різні сфери є невід’ємною складовою сучасного життя. Наразі присутність в онлайн-просторі є необхідною умовою ефективного роботи будь-якої компанії чи організації. Вебсайт виконує роль візитівки установи, засобу комунікації з клієнтами, основного джерела актуальної інформації та інструменту залучення нової аудиторії.

Спортивні клуби, зокрема тенісні, мають особливі вимоги, що потребують сучасного та функціонального вебзастосунку. Основною метою такого вебсайту має бути оптимізація процесу взаємодії між клубом та його чинними членами або потенційними клієнтами з акцентом на доступність і зручність користування. Наприклад, на вебсайті тенісного клубу має бути інформація про тренерів, тарифи, розклад, а також стрічка новин і правила користування кортами.

Незважаючи на те, що зараз проходить активне впровадження цифрових рішень у різні сфери, чимала кількість тенісних клубів досі покладаються лише на сторінки в соціальних мережах або телефонний зв'язок. Подібний підхід у більшості випадків є неефективним, оскільки знижує швидкість передачі та отримання інформації. Повноцінний вебсайт, навпаки, дає можливість не тільки надавати актуальні дані про різні заходи та тренування, а також виступає потужним маркетинговим інструментом, який підвищує рівень довіри до клубу, розширює аудиторію та формує позитивний імідж.

У сучасних умовах швидкий доступ до динамічного контенту вже давно є нормою, особливо актуальним є використання інноваційних інструментів розробки, таких як React. Ця технологія дає можливість розробляти вебзастосунки з гнучким інтерфейсом, в якому оновлення відбуваються без перезавантаження сторінки. Ця

функція є особливо зручною під час роботи з розкладами та новинами. React забезпечує високу швидкість роботи, зручність та легкість підтримки, що сьогодні є дуже важливими факторами для довгострокового розвитку вебсайту тенісного клубу.

Окрім технічних плюсів, React має велику спільноту розробників і широкий набір готових базових рішень, що скорочує час розробки деяких компонентів у кілька разів. Завдяки цьому можна швидко створити зручний та функціональний інтерфейс, який відповідатиме всім потребам різних користувачів. Використання такого гнучкого інструмента в основі вебзастосунку гарантує його актуальність, стабільність, ефективність та можливість спокійного розширення функціоналу в майбутньому.

Підсумовуючи, розробка сучасного вебсайту для тенісного клубу є важливим кроком. Це дозволяє не лише покращити організацію роботи клубу, а й забезпечити найвищий рівень обслуговування, що відповідає очікуванням сучасних користувачів. Особливу увагу слід приділити технічній реалізації проєкту, зокрема вибору оптимального інструменту розробки. Використання React як основи для створення інтерфейсу гарантує ефективність та стабільність вебзастосунку. Отже, актуальність даної теми зумовлена не лише практичними потребами організації ефективної роботи тенісного клубу, а й необхідністю використання сучасних технологій для забезпечення конкурентоспроможності на ринку послуг.

1.2 Огляд інструментів для розробки фронтенду вебсайтів

Сьогодні майже кожен веброзробник при створенні вебсайтів застосовує три основні технології: HTML, CSS та JavaScript. Ці три компоненти утворюють комбінацію, яка забезпечує формування структури, візуального оформлення та динамічної поведінки вебсторінок.

HTML (англ. HyperText Markup Language) являє собою базовий елемент усього вебпростору. Вона визначає структуру та ієрархію наповнення вебсайту використовуючи спеціальні елементи, що позначаються тегами. HTML надає каркас, на основі якого створюється візуальна складова вебінтерфейсу. Наприклад, елементи

<h1> і <p> позначають відповідно заголовки та абзаци, а <body> вміщує основний вміст документа. Теги мають відкриваючі та закриваючі форми, що дозволяє чітко розмежовувати логічні блоки контенту. HTML забезпечує лише базову структуру сторінки без стилізації чи динамічної взаємодії.

Для оформлення зовнішнього вигляду сторінки використовується CSS (англ. Cascading Style Sheets) – це так звана мова стилів, яка надає можливість змінювати колір, шрифт, розмір, відступ, а також реалізовувати адаптивність інтерфейсу. Застосування CSS дає змогу централізовано керувати виглядом усіх елементів вебресурсу, що значно підвищує ефективність роботи над дизайном. Кожному HTML-елементу можуть бути прив’язані певні стилістичні інструкції, які формують його остаточний вигляд на екрані користувача [4, 7].

Мова програмування JavaScript забезпечує інтерактивність вебдодатків. Вперше представлений у 1995 році, цей інструмент швидко завоював визнання завдяки ефективній взаємодії з користувачем у режимі реального часу. JavaScript дозволяє реалізовувати динамічні ефекти, обробку подій, анімацію, інтеграцію з базами даних, а також розробку повноцінних вебдодатків. З розвитком вебтехнологій JavaScript став незамінним інструментом як на клієнтському, так і на серверному боці, значно розширивши можливості створення багатофункціональних і масштабованих систем.

У комплексі HTML, CSS і JavaScript формують ядро фронтенд-розробки та забезпечують створення сучасних, інтерактивних і візуально привабливих вебзастосунків. Проте зі зростанням складності інтерфейсів та потреби в оптимізації їх продуктивності виникла потреба у більш гнучких інструментах, що дозволяють ефективно керувати оновленнями інтерфейсу та повторним використанням коду. Одним із таких рішень є бібліотека React.

React – це відкритий JavaScript-засіб для розробки інтерфейсів користувача, який побудований на компонентному підході з використанням JSX-синтаксису. Розроблений командою Facebook (тепер Meta) у 2011 році, він був представлений як open-source проєкт у 2013 році. Бібліотека спрямована на вирішення проблем

часткового оновлення вмісту вебсторінок, що особливо актуально для односторінкових застосунків.

У React реалізовано механізм віртуального DOM, завдяки якому оновлення інтерфейсу відбувається більш ефективно за рахунок порівняння поточного стану з новою версією документа. Основною ідеєю є одностороння передача даних, завдяки чому компоненти залишаються незалежними та повторно використовуваними. Оскільки React є бібліотекою, для реалізації управління станом, маршрутизації та взаємодії з API часто використовуються додаткові рішення, наприклад, Redux.

React знаходить застосування у широкому спектрі вебзастосунків: від соціальних мереж до корпоративних платформ. Серед популярних прикладів – Facebook, Instagram, YouTube, TikTok, Netflix, Asana, Reddit, Airbnb, Pinterest, UberEats, BBC, Messenger, Discord, Crisp Chat, Shopify, Snapchat, Spotify та інші.

Завдяки своїй простоті, високій продуктивності та можливості легкого комбінування з іншими бібліотеками, React виступає ефективним засобом для розробки сучасних вебінтерфейсів. Його компоненти, включаючи функціональні та класові, дозволяють створювати масштабовані та динамічні застосунки . З появою нових технологічних підходів, таких як React Hooks та архітектура Fiber, React продовжує залишатися фаворитом серед розробників, що прагнуть до інноваційних рішень у сфері фронтенд-розробки [1, 2, 6].

1.3 Аналіз застосунків-аналогів

Перед початком розробки вебсайту потрібно зробити огляд аналогів, проаналізувати їхні плюси та мінуси, щоб уникнути подібних помилок у проєкті.

Користувачі, які вперше заходять на вебсайт, майже ніколи не оцінюють його з професійної точки зору. Їх аналіз базується зазвичай на власних враженнях, і в більшості випадків для оцінювання використовуються такі епітети, як «гарний», «хороший», «зрозумілий», «заплутаний», «незручний» тощо. Однак для детальнішого аналізу суб’єктивного враження буде недостатньо. Для оцінювання потрібно

використовувати об’єктивні параметри, які враховують структуру, дизайн, функціональність і наповнення вебресурсу.

Одним із таких параметрів є показник відмов (bounce rate), який дає інформацію про частку користувачів, що закривають вебсайт після перегляду головної сторінки. Для інформаційних і презентаційних вебсайтів, до яких належить і вебзастосунок тенісного клубу, задовільним вважається bounce rate у межах 20–45 %[8]. Для найуспішніших вебсайтів середній показник близький до 36 %.

Швидкість завантаження сторінки є особливо критичним фактором, що безпосередньо впливає на перше враження користувача про вебсайт. Приблизно 47 % користувачів очікують, що сторінка відкриється менш ніж за 2 секунди, а 40 % залишають вебсайт, якщо завантаження триває понад 3 секунди. Кожна додаткова секунда може зменшити рівень задоволеності на 16 % [9].

Ще одним не менш важливим параметром є середня тривалість сеансу (average session duration), яка демонструє, скільки часу користувачі взаємодіють із вебсайтом. За даними аналітичного сервісу Contentsquare, приблизний середній час перебування на вебсайті для різних сфер становить від 1 хвилини 30 секунд до 3 хвилин. Наприклад, для вебресурсів, що надають освітні або клубні послуги, цей показник становить 2 хвилин 52 секунд, що свідчить про достатній інтерес користувачів до контенту [10].

Таким чином, впроваджуючи вебзастосунок тенісного клубу на основі фреймворку React, доцільно орієнтуватися на зазначені показники як на інструменти постійного вдосконалення проєкту. Вони можуть бути використані для подальшої оптимізації структури вебсайту, логіки навігації та якості вмісту, що, у свою чергу, позитивно вплине на залучення нових відвідувачів і підвищення рівня їхньої задоволеності.

Щоб правильно проаналізувати вебсайт потрібно почати з розуміння його призначення. Він повинен ефективно виконувати поставлене завдання і бути корисним для цільової аудиторії. Наприклад, якщо це вебсайт для тенісного клубу, то

для відвідувачів важливою буде можливість швидко знаходити інформацію про розклад тренувань.

Переглядаючи аналоги, дуже важливо звертати увагу на доцільність всіх елементів: кожна сторінка, кожен розділ мають відповідати своєму чіткому призначенню, а інформація має бути зрозумілою, доцільною та актуальною.

Також одним із найголовніших аспектів є логіка побудови вебсайту. Цей аспект відповідає за зручність орієнтування в меню та швидкість знаходження потрібної інформації користувачем. Окремої уваги вартує художнє оформлення вебзастосунку: кольорова гама, типографіка, візуальний баланс елементів – все це має вплив на загальну оцінку вебсайту.

Аналіз аналогічних вебсайтів тенісних клубів дає змогу виокремити найбільш ефективні рішення та сформувати обґрунтовані висновки для розробки власного проєкту. Це сприяє створенню ресурсу, який буде не тільки привабливим зовні, а й зручним та ефективним для користувача [12].

На сьогодні кількість вебсайтів тенісних клубів є відносно невеликою, і не всі з них є зручними у користуванні. З метою аналізу пропонується розглянути декілька прикладів. На рис. 1.1 представлено головну сторінку вебсайту «MARINA Tennis Club».

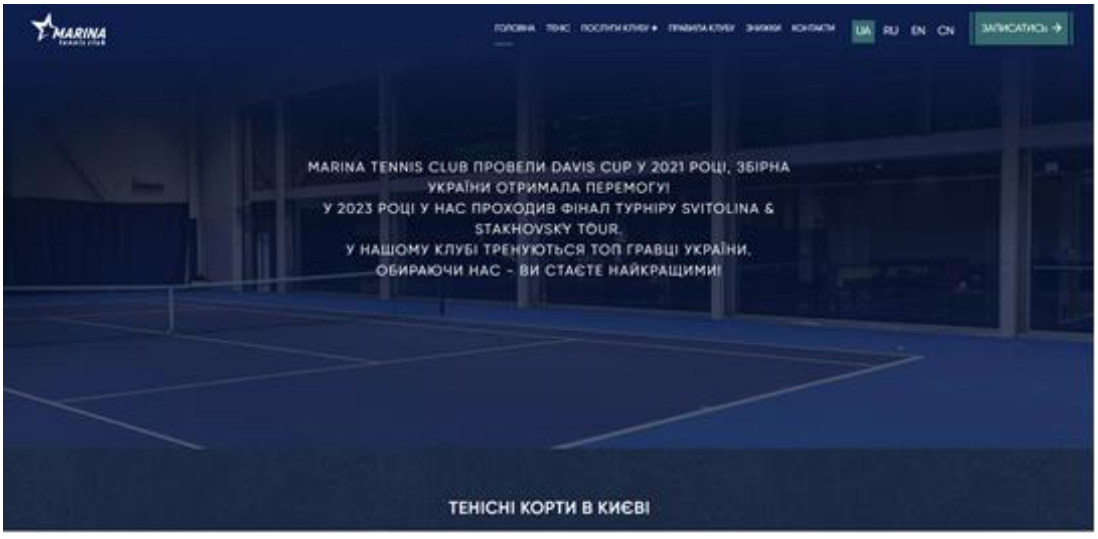


Рисунок 1.1 – Головна сторінка «MARINA tennis club»

На головній сторінці подано короткі відомості про клуб; навігація реалізована у вигляді горизонтального меню з можливістю вибору мови інтерфейсу та функцією запису на тренування. Водночас, на вебсайті відсутній функціонал для створення особистого кабінету користувача, а також немає можливості перегляду доступності кортів і здійснення онлайн-бронювання [25].

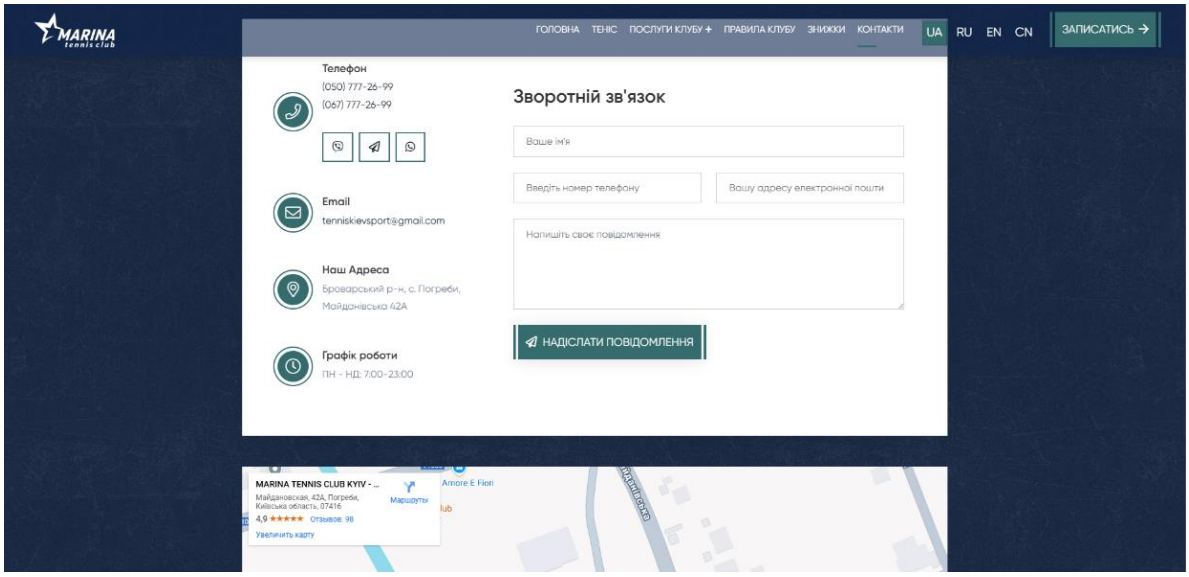


Рисунок 1.2 – Сторінка контактної інформації «MARINA tennis club»

На рис. 1.2 наведено вдало реалізовану секцію вебсайту, що включає інтерактивну карту для зручнішого визначення місцезнаходження клубу та форму зворотного зв’язку.

Далі розглянемо вебзастосунок українського тенісного центру (УТЦ) «Viccourt», головну сторінку якого зображено на рис. 1.3.

Рисунок 1.3 – Головна сторінка вебсайту УТЦ «Viccourt»

Перш за все, увагу привертає дизайн, який не відповідає сучасним стандартам візуального оформлення. Основною проблемою є те, що інтерфейс ресурсу перенасичений яскравими кольорами, які не гармонують між собою, а центральний банер виглядає надто мультяшним і не відповідає загальному стилю тенісного клубу. Це знижує сприйняття бренду як авторитетного та сучасного. Попри наявність основного функціоналу, відсутність особистого кабінету користувача та інших подібних інтерактивних елементів знижує загальну привабливість вебсайту, ускладнюючи процес залучення нових відвідувачів і не сприяючи формуванню довіри до ресурсу [26].

Останнім прикладом є вебсайт тенісного клубу «GRAYS». На рис 1.4 наведено головну сторінку ресурсу, а на рис 1.5 – форму реєстрації особистого кабінету користувача.

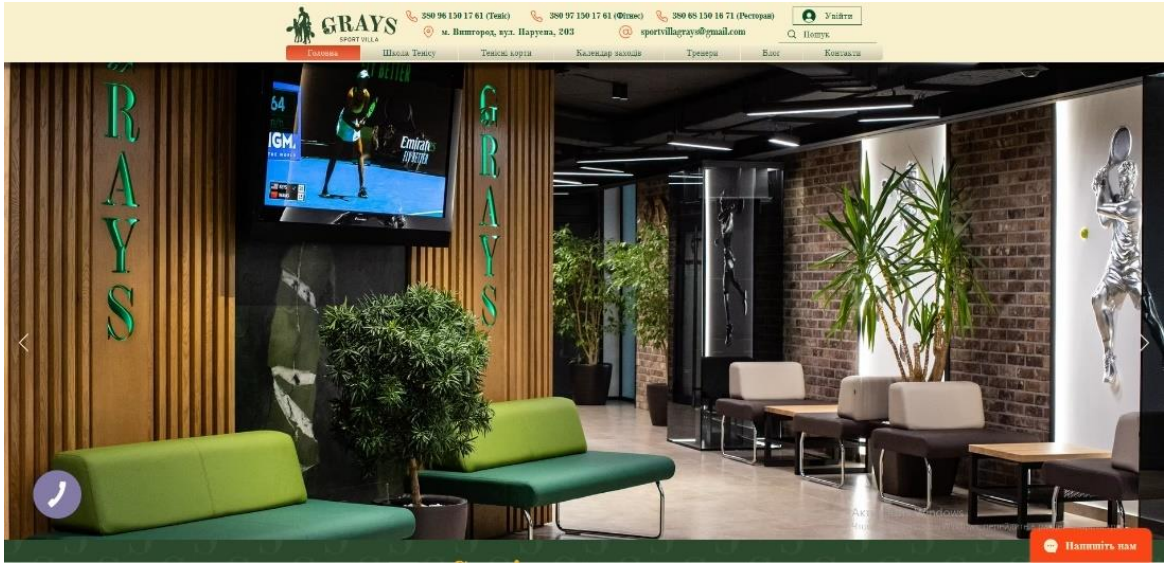


Рисунок 1.4 – Головна сторінка вебсайту тенісного клубу «GRAYS»

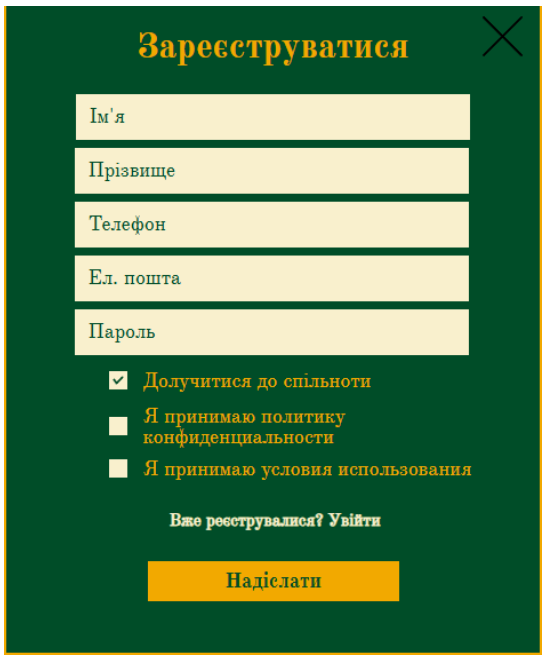


Рисунок 1.5 – Форма реєстрації кабінету користувача

Вебсайт «GRAYS» має зручну навігацію та чітке меню, що забезпечує комфортне користування, особливо актуальне в умовах сьогодення. Важливою перевагою є наявність функції створення особистого кабінету та авторизації, що

свідчить про орієнтованість клубу на сучасні сервіси для клієнтів. Це дозволяє користувачам управляти власними записами, переглядати розклад, а також здійснювати онлайн-оплату послуг. Вся інформація легко доступна, а інтерфейс працює на належному рівні. Проте візуальне оформлення більше асоціюється з атмосферою музею чи готелю, ніж зі спортивним клубом. Отже, попри високу функціональність, дизайн вебсайту не передає спортивного духу й динаміки, що варто врахувати для покращення позиціонування бренду [27].

З результатами аналізу можна ознайомитися в таблиці 1.1, де також було визначено, як знайдені недоліки в аналогах враховувалися під час розробки власного вебсайту.

Таблиця 1.1 – Порівняльна характеристика вебсайтів тенісних клубів

№	**Назва вебсайту**	**Основні переваги**	**Основні недоліки**	**Рішення у власному проєкті**
1	MARINA Tennis Club	Лаконічний дизайн Форма запису Контактна інформація з картою	Відсутній особистий кабінет Немає системи онлайн-бронювання	Реалізовано особисті кабінети з можливістю перегляду розкладу та бронювання кортів онлайн
2	УТЦ «Viccourt»	Наявність основного функціоналу	Застарілий дизайн Надлишок кольору Відсутність авторизації	Створено сучасний адаптивний інтерфейс з авторизацією за ролями та стилем у спортивній тематиці
3	GRAYS Tennis Club	Зручна навігація Особистий кабінет	Дизайн не передає спортивну атмосферу Стилістика не відповідає темі	Використано кольорову гаму та візуальні елементи, що підкреслюють динаміку та спортивність

Отже, після проведення аналізу, можна чітко сказати, що сучасний вебсайт тенісного клубу має поєднувати зручність використання з естетичним і тематично доречним дизайном. Основними помилками є перевантаженість інтерфейсу, недоречна кольорова гама та загальний стиль. Наявність особистого кабінету, зрозумілого меню та інтеграції з сервісами бронювання та оплати послуг, навпаки ж, дуже покращує користувацький досвід. Такі елементи додають відчуття організованості й сучасності.

Досконалий вебзастосунок тенісного клубу повинен не лише надавати основну інформацію, а й створювати атмосферу спорту та активності, викликати довіру й велике бажання стати членом тенісної родини. Він має бути простим у використанні, ефективним, функціональним і водночас візуально привабливим. Саме гармонійне поєднання технічної складової та доцільного дизайну дає змогу розробити ефективний інструмент комунікації з клієнтами.

1.4 Постановка завдання

У сучасних умовах цифрової трансформації суспільства важливо надати спортивним закладам, зокрема тенісним клубам, ефективні інструменти для автоматизації внутрішніх процесів і покращення взаємодії з клієнтами. Популярність онлайн-сервісів, таких як бронювання кортів, реєстрація на турніри, перегляд розкладу занять та отримання актуальних новин, постійно зростає. Відсутність зручного та функціонального вебзастосунку зменшує можливості клубу в наданні якісного сервісу, знижує ефективність адміністративної роботи та ускладнює комунікацію з користувачами.

Для вирішення цієї проблеми необхідно створити вебсайт, який забезпечить:

- представлення інформації про клуб, тренерів, події, новини та турніри;
- зручний інтерфейс для бронювання кортів;
- можливість реєстрації та створення особистих кабінетів для клієнтів;

– адміністративну панель для керування контентом;
– надійне зберігання та захист персональних даних користувачів.

Розробка має базуватися на передових вебтехнологіях, таких як HTML, CSS, JavaScript та React. Зручний функціонал цього стеку технологій забезпечує високу продуктивність, зручність використання, гнучкість на етапах розробки та підтримки, а також значно кращі можливості для масштабування в майбутньому.

Основними користувачами системи будуть: адміністратори, тренери та відвідувачі клубу. Для кожної з цих категорій буде передбачено свій функціонал. Відвідувачі матимуть змогу створити особистий кабінет, переглядати розклад, бронювати корти та реєструватися на події. Адміністрація клубу матиме доступ до CMS-панелі для редагування та публікації новин, керування списком клієнтів, бронюваннями та подіями. Потрібно врахувати, що інтерфейс має бути інтуїтивно зрозумілим навіть для користувачів без технічної підготовки. Всі функції мають відповідати нормам безпеки, стабільності та зручності.

Особливу увагу слід звернути на надійність системи: це передбачає захист персональних даних, застосування сучасних методів автентифікації та контроль доступу до адмінфункціоналу. Інформацію слід зберігати централізовано в базі даних із можливістю створення резервних копій та їх шифрування.

Враховуючи теоретичні відомості про структуру та види сервісів для користувачів у сфері спорту та ознайомившись з конкурентами, варто визначити обов’язкові вимоги до цифрового сервісу: адаптивний інтерфейс, зручну навігацію, доступ до функцій та розподіл контенту. За таких умов постає задача створення повноцінного вебзастосунку, який відповідатиме сучасним вимогам до продуктивності й безпеки, а також сприятиме підвищенню ефективності функціонування тенісного клубу.

Висновки до розділу 1

У першому розділі було проведено повний аналіз предметної області, що

дозволяє чітко позначити цілі та вимоги до майбутнього вебзастосунку для тенісного клубу. Зокрема, визначено потребу у створенні зручного, інформативного, ефективного та сучасного вебресурсу, що має виконувати інформаційні та комунікаційні функції. Представлення загальної інформації про клуб, демонстрація розкладу занять, можливість ознайомлення з тренерським складом, реєстрація, а також надання простого та інтуїтивного інтерфейсу для відвідувачів є основними можливостями майбутнього вебзастосунку.

У ході аналізу аналогічних вебсайтів було переглянуто кілька прикладів реалізації, що дозволило виявити ефективні дизайнерські та функціональні рішення, а також типові помилки, такі як перевантаження інформацією, складна навігація або відсутність адаптивності. Ці результати аналізу були враховано при формуванні вимог до власного проєкту.

Також було обґрунтовано вибір інструментів реалізації вебсайту. Основним засобом створення інтерфейсу було обрано бібліотеку React, яка дозволяє створювати гнучкі, масштабовані та динамічні користувацькі інтерфейси. Крім того, до складу технологічного стеку включено HTML, CSS та JavaScript, що є базовими технологіями фронтенд-розробки, які забезпечують повноцінну візуалізацію, стилізацію та інтерактивність вебсторінок. Даний вибір обумовлений їхньою широкою підтримкою, доступністю та відповідністю до поставлених технічних вимог.

Таким чином, було сформовано чітке уявлення про специфіку предметної області, а також закладено теоретичну та технічну основи для майбутньої реалізації вебзастосунку. Результати цього етапу стануть основою для створення ефективної структури вебсайту, розробки макетів інтерфейсу та подальшої реалізації логіки функціонування проєкту.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ТЕНІСНОГО КЛУБУ

2.1 Розробка загальної структури вебсайту

Структура вебзастосунку тенісного клубу базується на логіці взаємодії різних типів користувачів із вебсайтом, що дає можливість максимально точно визначити потреби кожної групи аудиторії – від аматорів і дітей до тренерів, батьків та адміністрації клубу [15].

Візуальна схема, що представлена нижче, демонструє загальний каркас вебсайту та взаємозв’язки між його основними розділами. Зверху структури розташована головна сторінка, яка є початковою точкою для навігації. Вона містить початкову інформацію, а також посилання на ключові розділи вебсайту, такі як реєстрація, блог із новинами та інформацію про адміністрацію. Після переходу до реєстраційної системи, користувач обирає свою роль у межах клубу – аматор, батько чи мати, тренер або дитина, що займається у спортивній школі. Надалі цей вибір надає доступ до окремих функцій та сторінок.

Для аматорів передбачені такі можливості, як участь у турнірах, бронювання кортів і взаємодія з тренерами та іншими гравцями клубу. Вони можуть переглядати календар турнірів, реєструватися на обрані події, а також бронювати корти, перевіряючи їх доступність та записуючись на зручний час. Окремий блок передбачено для комунікації з тренерами та партнерами для спарингів. На цій сторінці можна буде переглянути розклад занять і отримати контактну інформацію.

Діти, які займаються у клубі, отримують доступ до розділу дитячої спортивної школи. Цей підрозділ включає календар турнірів, функцію реєстрації, розклад тренувань, а також окрему сторінку, присвячену огляду школи. На ній представлено тренерський склад і контактні дані для батьків або потенційних новачків.

Тренери, в свою чергу, мають дещо розширений доступ до інформації, зокрема можливість переглядати та коригувати розклад занять, а також взаємодіяти з учасниками клубу в межах тренувального процесу. В них також є можливість доступу

до розділу дитячої спортивної школи, що дозволяє контролювати спортивні секції та слідкувати за підготовкою майбутніх тенісистів.

Розділ, присвячений батькам, логічно перетинається як з дитячим блоком, так і з інформацією про тренерів. Батьки можуть переглядати інформацію про тренерський склад, ознайомлюватися з розкладом і підтримувати зв'язок із персоналом клубу. Така структура забезпечує прозорість і зручність для родин, діти яких займаються спортом.

Блогова частина вебзастосунку слугує функцією інформаційного хабу, де розміщуються фотогалереї, новини клубу, поради гравцям і тематичні статті. Цей контент сприяє залученню аудиторії, формуванню спільноти та популяризації тенісу серед відвідувачів вебсайту.

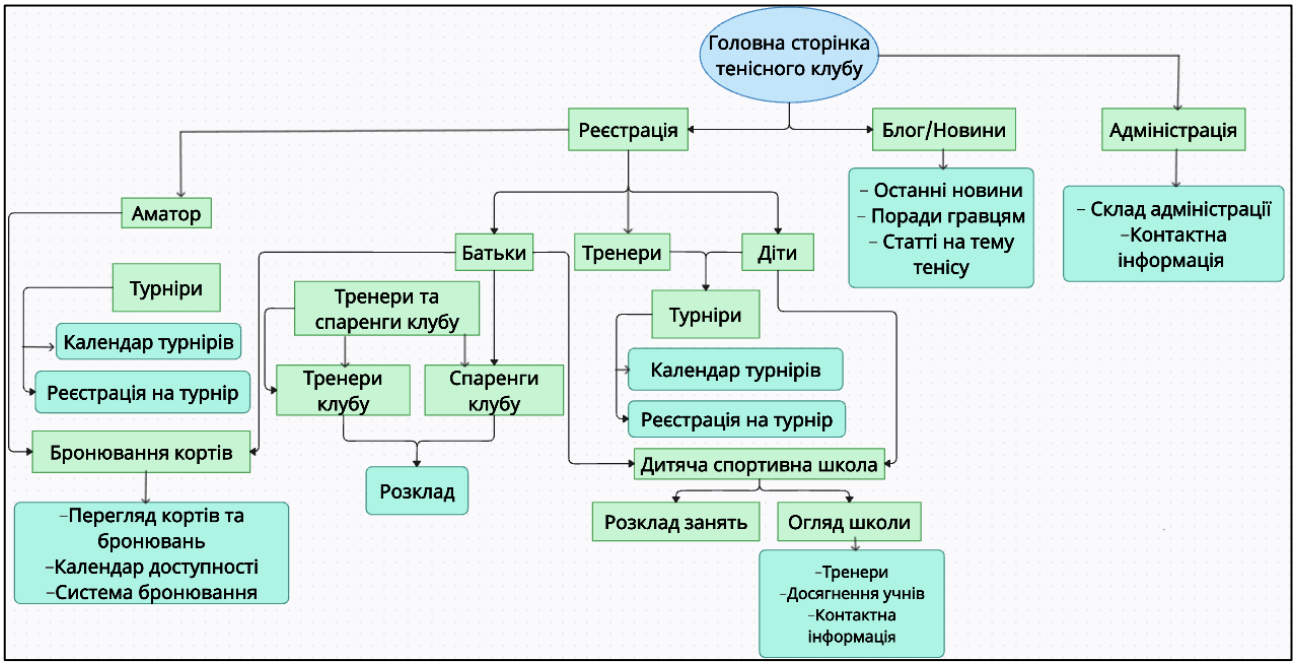


Рисунок 2.1– Розроблена структура вебсайту

Окремо організовано сторінку адміністрації клубу, де міститься службова інформація про склад адміністративної команди та контактні дані. Цей блок важливий для управління внутрішніми процесами клубу й забезпечення зворотного зв'язку з

користувачами.

Продемонстрована багаторівнева структура вебсайту є гнучкою та масштабованою. Вона враховує інтереси різних категорій користувачів і зберігає зручну навігацію навіть за умови подальшого розширення функціоналу. Такий підхід сприяє формуванню якісного користувацького досвіду та підтримує ефективну організацію роботи клубу через вебінтерфейс.

2.2 Розробка дизайну

Дизайн вебсайту тенісного клубу «TL» було розроблено індивідуально для кожної сторінки з урахуванням її змісту. Подібний підхід дозволив зробити кожен розділ візуально унікальним, зберігаючи при цьому стилістичну єдність. В оформленні вебсайту переважає мінімалізм, з акцентом на зручність користувача, контрастність та логічну структуру елементів [5].

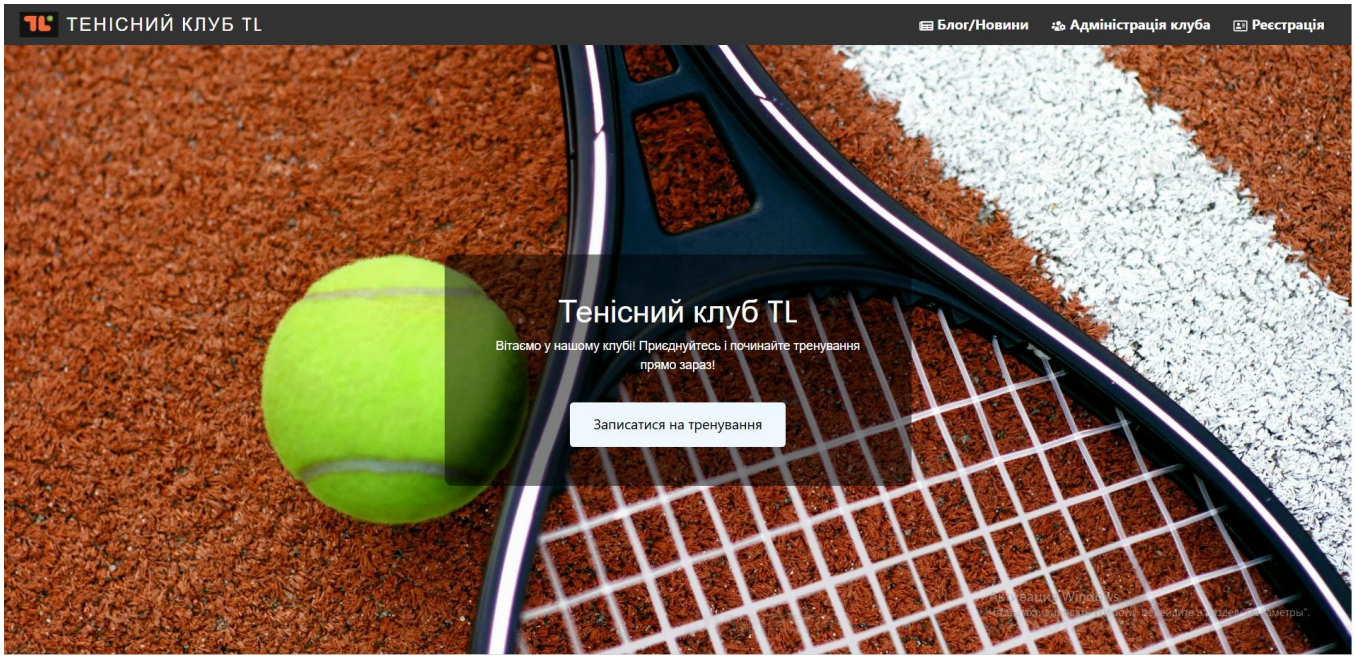


Рисунок 2.2 – Головна сторінка вебсайту

Спочатку було спроєктовано дизайн головної сторінки, де представлено логотип клубу, навігаційне меню, а також основний інформаційний блок. Усі елементи мають достатньо вільного простору для комфортного сприйняття.

Далі було створено дизайн для сторінки «Блог / Новини», який відрізняється яскравою кольоровою палітрою з градієнтним фоном. Кожна новина подана у вигляді кольорової картки з кнопкою переходу до детальної інформації. Угорі розміщено фільтр сортування за датою публікації, що покращує навігацію.

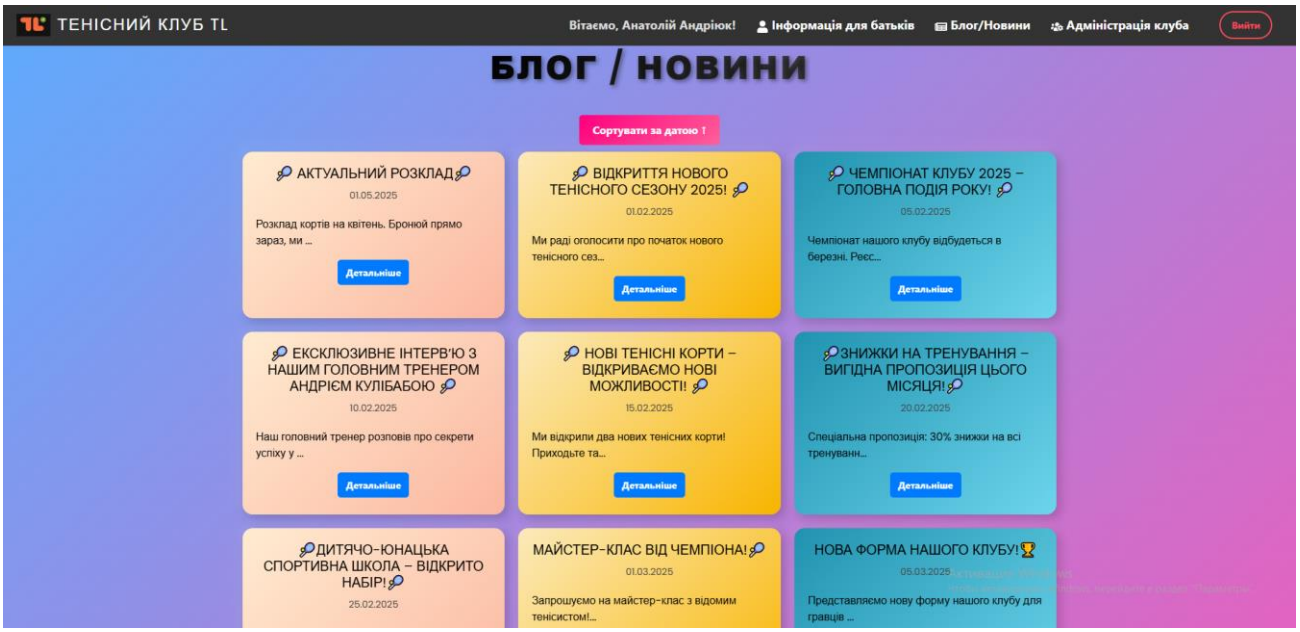


Рисунок 2.3 – Сторінка «Блог / Новини»

Окремо розроблено дизайн сторінки «Адміністрація клубу», виконаний у спокійних відтінках із чітким поділом на блоки: керівництво та адміністративний персонал. Структура сторінки проста, із використанням круглих рамок для виділення імен, що створює легкий та водночас офіційний стиль.

Особливої уваги заслуговує дизайн сторінки реєстрації, де реалізовано зручну та інтуїтивно зрозумілу форму. Візуально вона розміщена в білому контейнері з тінями на фоні, що надає їй виразності. Для кожного поля додано відповідну іконку,

а кольорові кнопки «Зареєструватися» та «Увійти» роблять сторінку зручною для користувача.

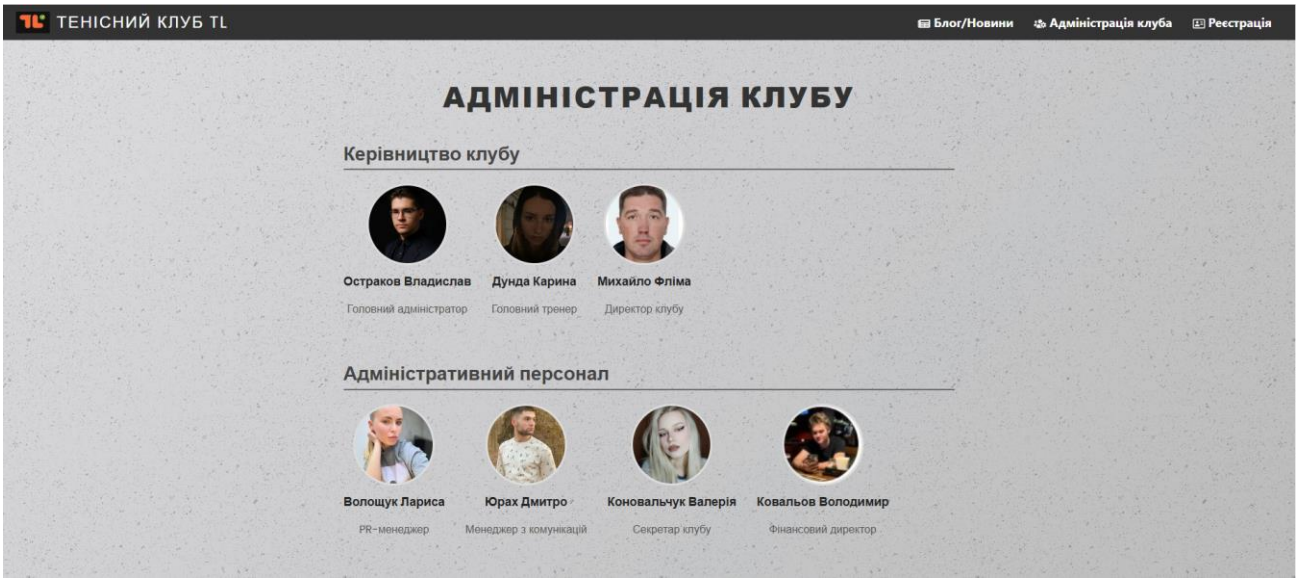


Рисунок 2.4 – Сторінка «Адміністрація клубу»

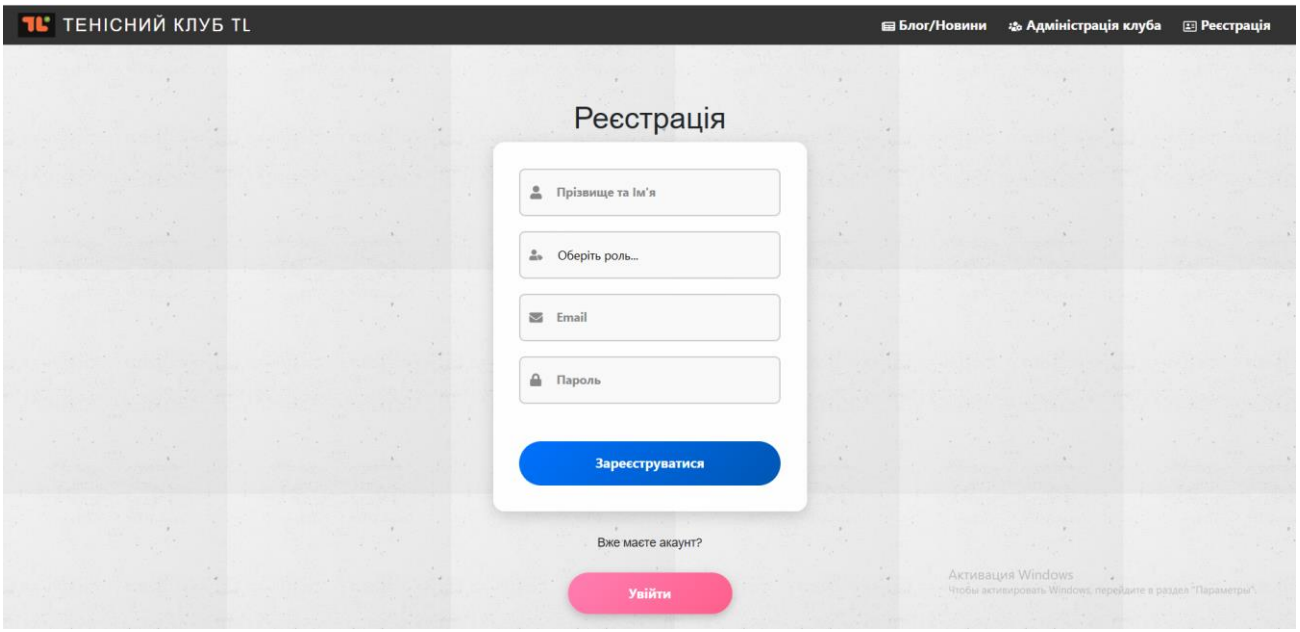


Рисунок 2.5 – Сторінка «Реєстрація»

Хедер (верхнє меню) доступний на всіх сторінках. Він містить логотип клубу, навігаційні посилання до основних розділів, а також адаптується під мобільні

пристрої. Його дизайн – стриманий, із темним фоном, що контрастує з яскравими кольорами основного вмісту.

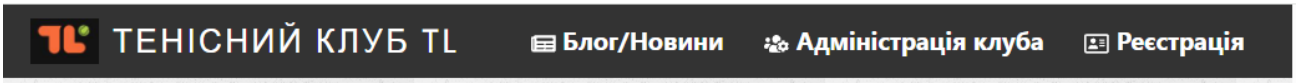


Рисунок 2.6 – Хедер вебсайту

Футер (нижня частина вебсайту) оформлено у єдиному стилі з хедером. У ньому розміщено контактну інформацію, дублювання навігації та посилання на сторінки у соціальних мережах. Футер створює завершений вигляд сторінки та полегшує орієнтацію для користувача.

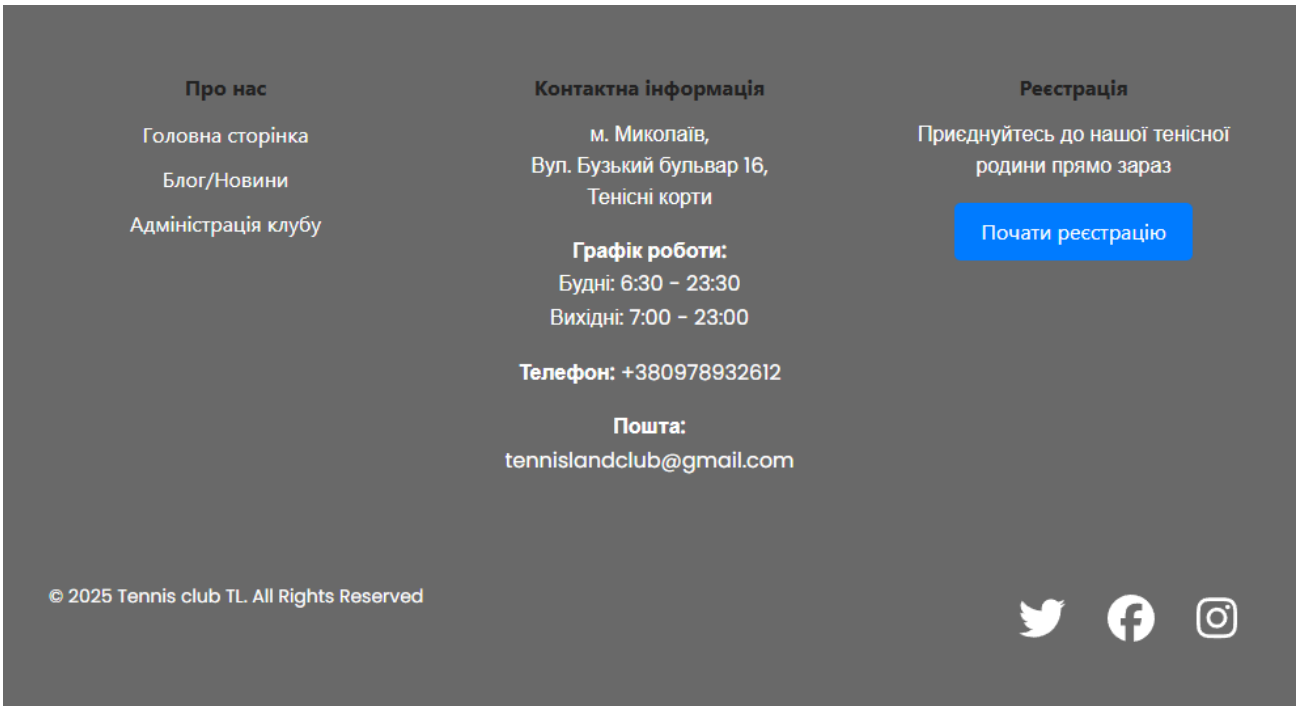


Рисунок 2.7 – Футер вебсайту

Загалом, кожна сторінка вебсайту має власний унікальний дизайн, що дозволяє зробити інтерфейс більш ефективним і привабливим. Водночас усі елементи підпорядковані загальній стилістиці, завдяки чому вебсайт зберігає візуальну цілісність.

2.3 Адаптивність та зручність інтерфейсу

Сучасний вебресурс повинен бути зручним, зрозумілим і простим у використанні для користувача незалежно від того, з якого пристрою він заходить. Саме для цього при розробці вебсайту тенісного клубу було враховано принципи адаптивної верстки, інтерактивності, швидкої навігації та ролезалежної логіки інтерфейсу.

Інтерфейс сайту автоматично підлаштовується під різні розміри екранів, що особливо важливо для користувачів мобільних телефонів, які становлять суттєву частину відвідувачів. Наприклад, головне меню приховується за компактною кнопкою, яка відкриває розгорнуту навігаційну панель лише за потреби. Це дозволяє зберегти простір для основного вмісту.

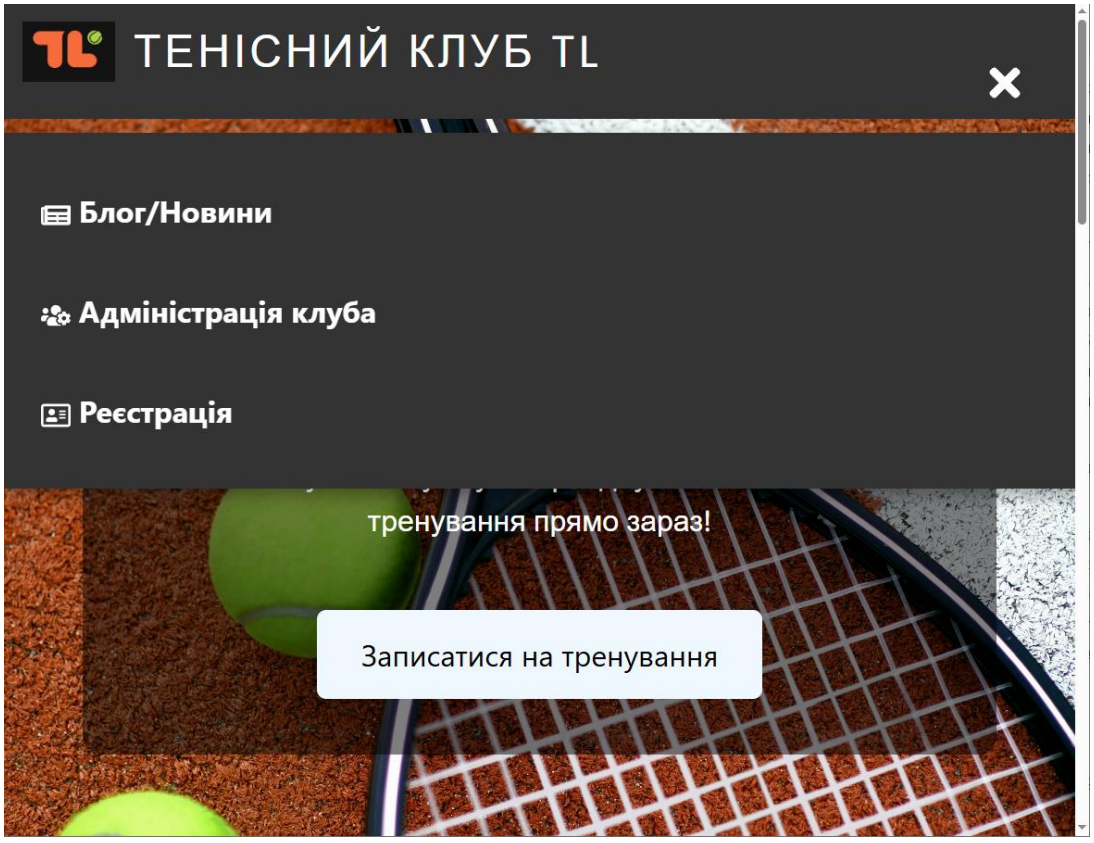


Рисунок 2.8 – Скриншот адаптивного меню на мобільному пристрої

Сторінки на яких розташована велика кількість інформації, як наприклад розклад, база учнів або турнірна панель, реалізовано через вкладкову навігацію з переходом до наступного вмісту без перезавантаження сторінки. Подібне рішення дозволяє поділити функціонал на логічні блоки, а користувачеві – швидше та простіше орієнтуватися в інтерфейсі.

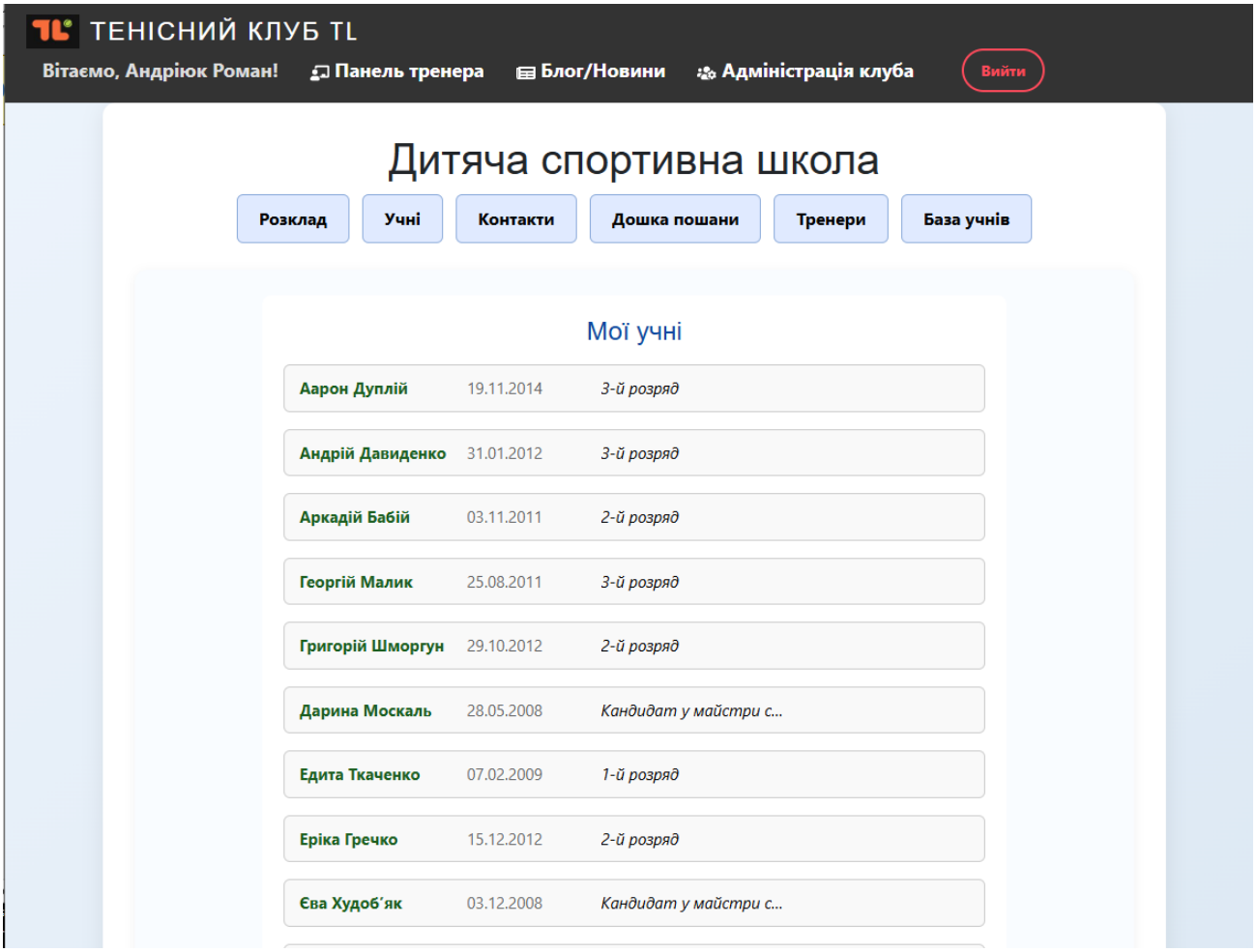


Рисунок 2.9 – Скриншот сторінки з активною вкладкою «Учні»

В інтерфейсі реалізовані як візуальні, так і поведінкові оптимізації. Наприклад, усі інтерактивні елементи (кнопки, чекбокси, поля вводу) мають достатній розмір для сенсорного керування. Для полів введення передбачено плаваючі підказки, які не

перекривають текст і залишаються видимими у процесі взаємодії. Вибір дати чи часу реалізовано у вигляді стандартних календарів або чекбоксів, які швидко адаптуються до розміру екрана.

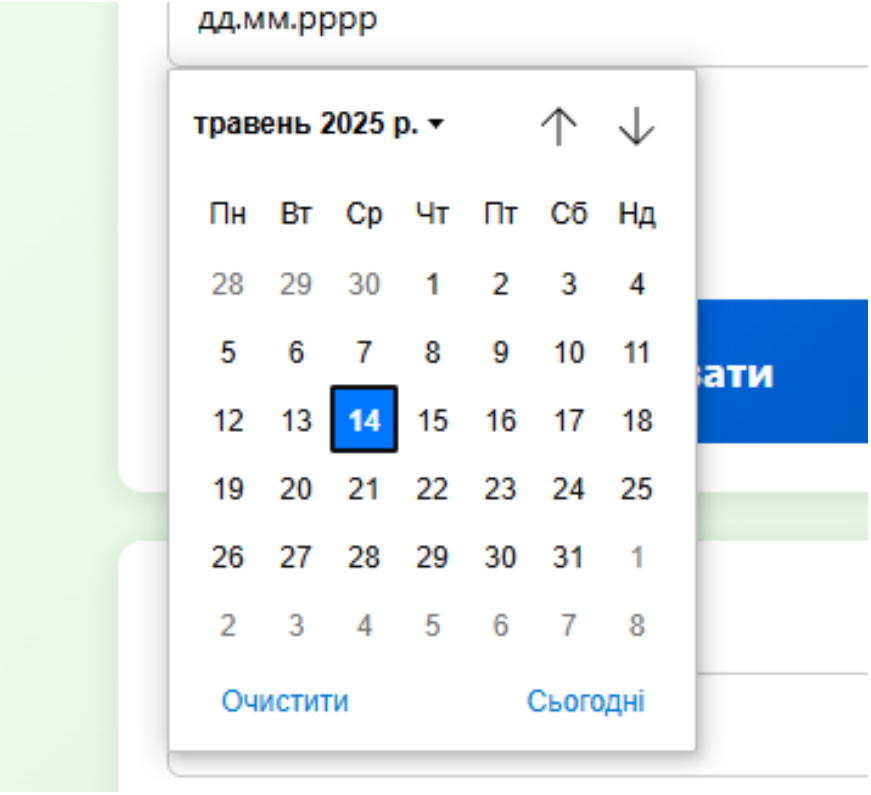


Рисунок 2.10 – Форма з інтерактивними полями, зокрема датою

Ще одним не менш важливим аспектом є побудова інтерфейсу на основі ролі користувача. Тобто, в залежності від обраної ролі при вході, система надає доступ до різного обсягу функцій. Такий підхід не тільки покращує навігацію, а й спрощує інтерфейс – користувач може взаємодіяти лише з тим функціоналом, який йому потрібен. Це особливо актуально для батьків та дітей, для яких надмірна кількість технічних елементів може бути незрозумілою або зайвою.

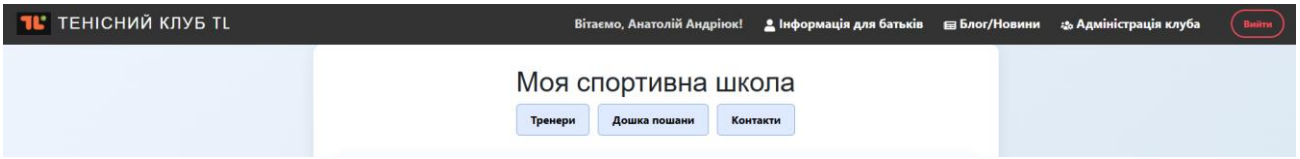


Рисунок 2.11 – Панель користувача з роллю батько

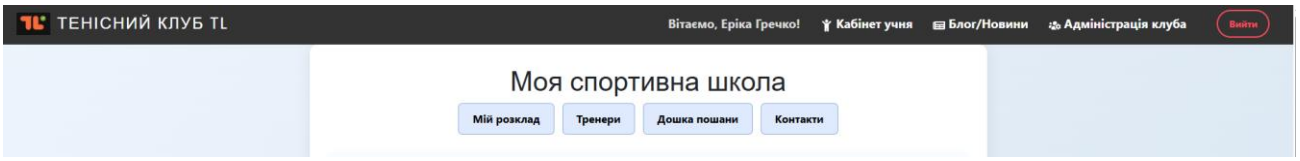


Рисунок 2.12 – Панелі користувача з роллю учень

Кольорова гамма сайту адаптована до кожного розділу, однак не перевантажена – обрано легкі градієнтні фони та контрастні шрифти, що покращують сприйняття контенту навіть при поганому освітленні або на екранах із низькою роздільною здатністю.

Окремої уваги заслуговує швидкість реакції інтерфейсу. Завдяки використанню односторінкової структури та компонентного підходу, взаємодія з елементами таких, як відкриття вкладок, підтвердження дій, перегляд деталей, відбувається без перезавантаження сторінки, що значно підвищує комфорт використання.

Таким чином, адаптивність у поєднанні з розумно продуманим UX стала одним із головних принципів реалізації вебзастосунку. Це дозволило зробити ресурс максимально зручним і водночас універсальним – як для швидких дій, як бронювання або реєстрація, так і для перегляду інформаційних блоків.

2.4 Організація ролей користувачів

Ключовим завданням під час розробки інформаційної системи є впровадження чіткої та зрозумілої системи розподілення прав доступу залежно від ролі користувача. Такий підхід дає змогу підвищити зручність у користуванні вебзастосунком,

забезпечити безпеку даних, а також сформувати персоналізований інтерфейс для кожної категорії відвідувачів сайту.

На етапі реєстрації відвідувач вебсайту обирає одну з чотирьох запропонованих ролей: учень, тренер, аматор або батько/мати. Відповідно до вибраної ролі, після входу в систему користувачу надається доступ до певного набору функцій і сторінок. Подібна модель дозволяє уникнути перевантаження інтерфейсу зайвими елементами та забезпечує цільову взаємодію з системою.

Учні мають змогу переглядати свій розклад тренувань, реєструватися на турніри, знайомитися з тренерським складом та переглядати свої досягнення. Їхній інтерфейс є максимально спрощеним, адаптованим під молодшу аудиторію. Особливу увагу приділено доступності й простоті навігації, щоб навіть без підготовки користувач міг легко орієнтуватися в системі.

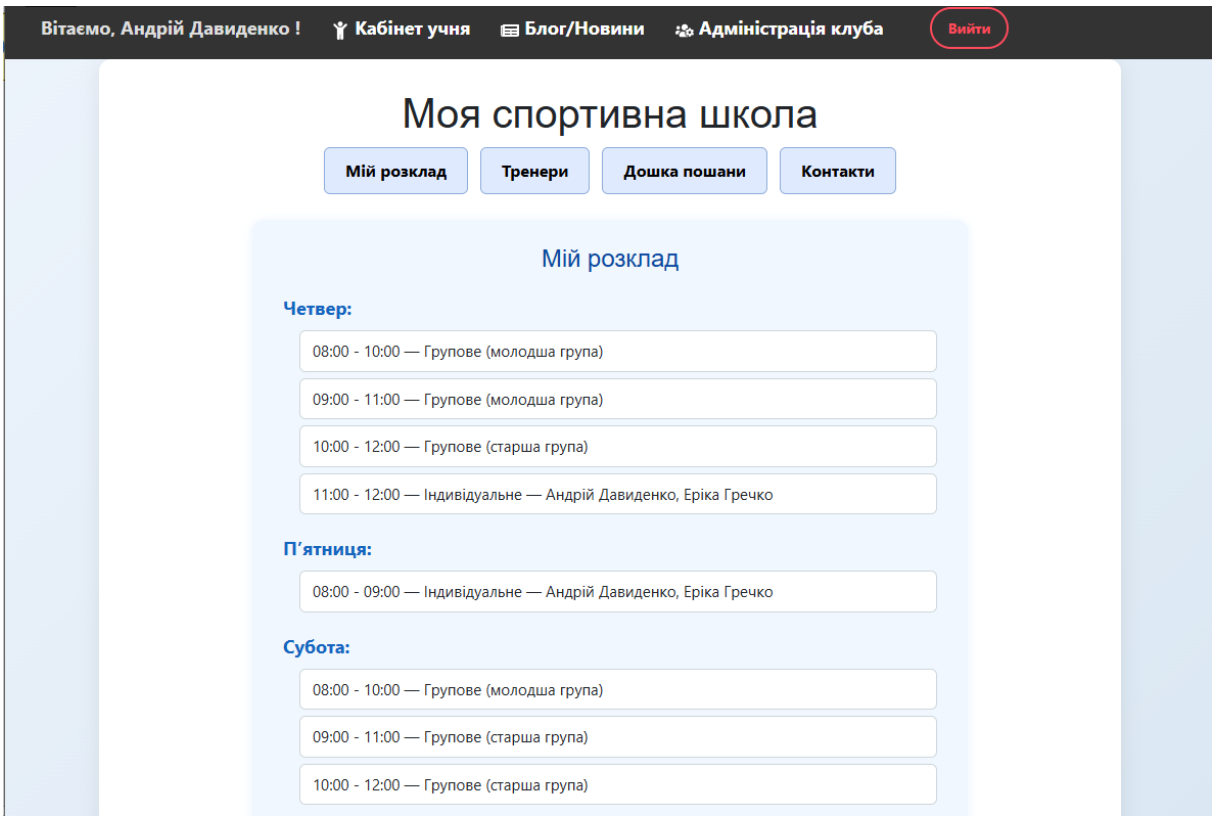


Рисунок 2.13 – Скриншот персонального кабінету учня
з вкладкою «Мій розклад»

Батьки мають доступ до вкладок тренери, контакти школи, досягнення учнів і звісно можуть переглядати розклад тренувань своєї дитини. Це надає зручності та простоти в координуванні тренувань, а також дозволяє бачити зміни в графіку та краще планувати щоденні справи. Цей функціонал забезпечує прозорість тренувального процесу і робить зв'язок між родиною та спортивним клубом міцнішим. Водночас інтерфейс залишається простим і зрозумілим, без зайвих елементів, що не стосуються безпосередньо батьківської ролі.

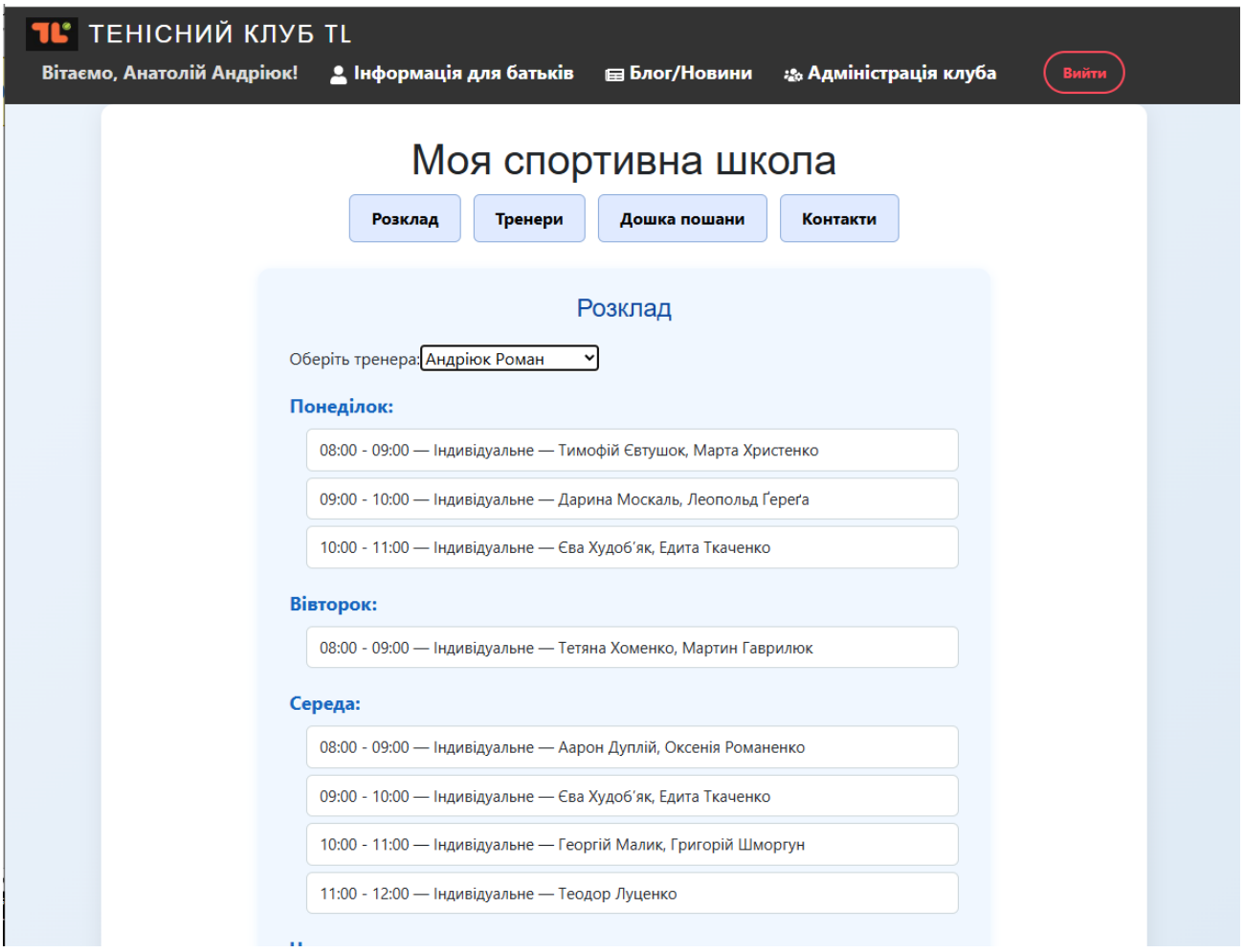


Рисунок 2.14 – Скриншот інтерфейсу батьків
з активною вкладкою розкладу

Для аматорів реалізовано окрему панель, у якій доступні можливості участі в турнірах, бронювання кортів, перегляду тренерів клубу і спарингів клубу. Інтерфейс цієї категорії користувачів зосереджено на взаємодії з клубом у неформальному або змагальному форматі, без прив'язки до навчального процесу. Завдяки чіткому структурованому функціоналу, аматор може швидко знайти потрібну інформацію або виконати дію – наприклад, записатися на турнір чи забронювати корт.

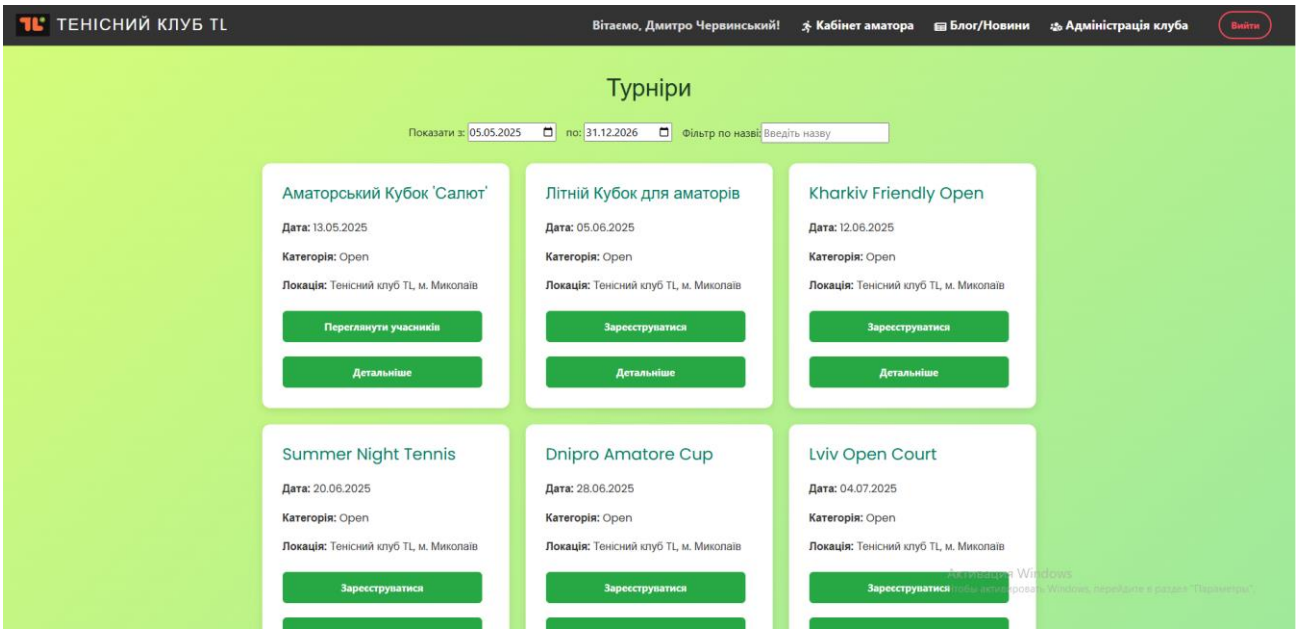


Рисунок 2.15 – Скриншот панелі аматора з активною секцією «Турніри»

Тренери мають розширений доступ до інформації: вони можуть бачити розклад занять, переглядати список своїх учнів, а також керувати навчальним процесом. Крім того, їм доступна аналітична інформація про категорії учнів, їхній вік, розподіл за статусом та активністю, що дозволяє будувати ефективну стратегію тренувального процесу. Вся інформація згрупована за тематичними вкладками, що дозволяє тренеру швидко перемикатися між функціональними розділами.

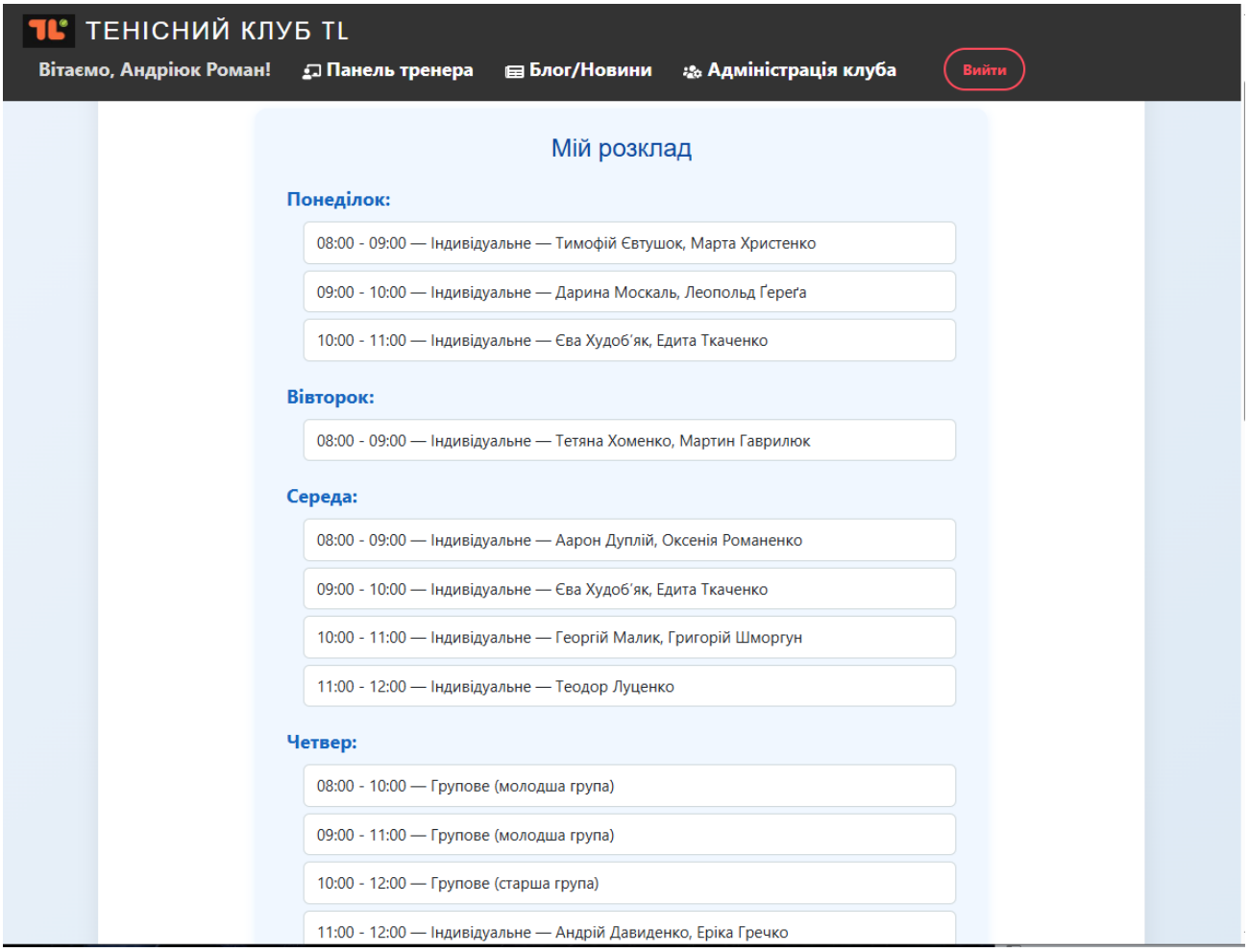


Рисунок 2.16 – Скриншот тренерської панелі зі розкладом тренувань

Адміністративна роль реалізована як прихована, призначена вручну через систему керування користувачами. Після авторизації адміністратор отримує розширений доступ до всієї інформаційної структури сайту. Він має змогу додавати, редагувати та видаляти новини, турніри, учнів, бронювання, а також керувати списками тренерів, кортами й досягненнями. Для адміністратора доступні окремі панелі з інтерфейсом керування контентом, які містять інструменти для швидкого внесення змін у режимі реального часу. Таке розмежування забезпечує централізоване управління даними без ризику стороннього втручання.

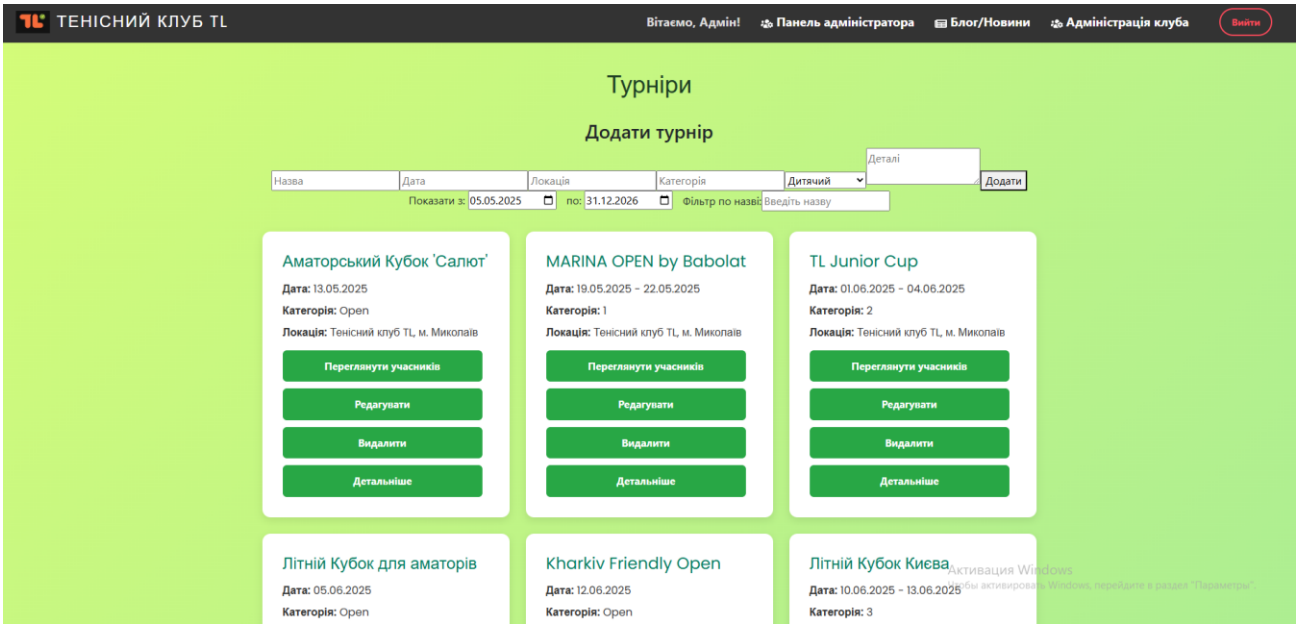


Рисунок 2.17 – Скриншот адміністративної панелі з кнопками редагування контенту

Повна логіка інтерфейсу реалізована так, щоб користувач міг бачити тільки ті елементи, які відповідають його ролі. Це надає додаткову безпеку, зводить до мінімуму кількість помилок і забезпечує індивідуальний підхід до кожного типу взаємодії. Крім того, використовується механізм автентифікації з перевіркою ролі після входу, що робить неможливим зміну ролі вручну через інтерфейс.

Отже, система розподілення ролей у проєкті не лише робить простішою та зрозумілішою структуру інтерфейсу, а й звичайно є основою для безпечного та ефективного функціонування всієї інформаційної системи. Вона дозволяє відвідувачам зосередити свою увагу на тих функціях, які їм дійсно потрібні, а адміністрації – керувати процесами без порушення цілісності даних.

2.5 Проєктування бази даних

Для збереження структурованих даних вебзастосунку тенісного клубу було обрано документоорієнтовану базу даних MongoDB [14, 24]. Такий підхід є оптимальним для сучасних вебресурсів, що базуються на JavaScript-стеку, оскільки

забезпечує гнучке зберігання вкладених об’єктів, високу продуктивність при роботі з масивами даних, а також масштабованість.

База даних була спроєктована відповідно до ключових сутностей сайту та логіки їхньої взаємодії. Загальна структура включає кілька основних колекцій:

– *користувачі*: в даній колекції зберігаються інформація про зареєстрованих осіб, зокрема ім’я, електронну адресу, пароль (захешований), роль (учень, тренер, батько, аматор, адміністратор). У залежності від ролі, один і той самий механізм автентифікації надає доступ до різних частин вебсайту;

– *учні*: це окрема колекція для збереження детальної інформації про кожного учня – ім’я, прізвище, дата народження, розряд, статус (поточний чи випускник), а також ПІБ тренера в якого тренується/тренувався. Ці записи використовуються для генерації розкладу, виведення списків, а також для модулів пошуку і фільтрації в базі даних учнів;

– *турніри:* тут містяться назва, дата проведення, тип (дитячий чи аматорський), категорія, локація, опис, а також вкладений список зареєстрованих учасників. На дитячих турнірах учасники діляться на групи хлопців і дівчат. Усі дані зберігаються в одному документі, що дозволяє швидко витягувати повну інформацію для відображення;

– *бронювання*: дана колекція реалізована за допомогою масивів, які знаходяться всередині об'єктів «кортів». В кожному масиві знаходяться такі дані про корт – унікальний ідентифікатор, номер, тип (відкритий/критий), тип покриття (хард, грунт тощо) та додатковий масив бронювання, в якому зберігаються додаткові дані про бронювання, а саме ім’я користувача, дата, діапазон часу. Такий підхід дає змогу ефективно зберігати пов’язані записи, не створюючи окремих зв’язків [17, 18].

Проєктування бази даних також передбачає підтримку можливостей пошуку та фільтрації. Наприклад, при відображенні бази учнів доступні фільтри за ПІБ, статусом і тренером. У випадку турнірів – пошук за назвою, типом і датами. Це забезпечується динамічними запитами до відповідних колекцій із використанням умов MongoDB.

Усі операції запису, оновлення та видалення даних реалізовано через API-запити до Express-сервера, де відбувається перевірка прав доступу, автентифікація за токенами та валідація вхідних даних. Таким чином, база даних не лише виконує роль сховища, а й є активною частиною захищеної логіки вебзастосунку [19, 20].

У цілому структура бази даних була побудована з урахуванням майбутнього розширення функціоналу: передбачено додавання нових ролей, типів турнірів, розширення системи до аналітики участі, генерації звітів тощо. Завдяки використанню MongoDB система є гнучкою, надійною та зручною в адмініструванні [16].

Висновки до розділу 2

У другому розділі було проведено покрокову розробку структури та дизайну вебсайту тенісного клубу «TL». Спочатку сформовано багаторівневу архітектуру ресурсу, яка охоплює різні категорії користувачів: від аматорів і дітей до тренерів, батьків та адміністрації клубу. Ця модель дозволяє ефективно організувати навігацію, забезпечити швидкий доступ до потрібної інформації та оптимізувати цифрову взаємодію в межах клубу.

Особливу увагу приділено індивідуальному дизайну сторінок, кожна з яких була створена з урахуванням функціонального навантаження та потреб цільової аудиторії. Візуальне оформлення поєднує мінімалізм, логічну структуру, адаптивність та зручність для користувача. Завдяки цьому вебресурс виконує не лише інформаційні та організаційні функції, але й сприяє формуванню позитивного враження про клуб, підвищенню довіри та залученню нових користувачів.

Також було додано адаптивність для різних пристроїв, яка дозволяє користуватися сайтом як на комп’ютерах, так і на планшетах з мобільними пристроями. Адаптивне відображення інтерфейсу, динамічне меню, вкладки та оптимізовані форми підвищують зручність і мобільність взаємодії з сайтом.

Створена система ролей дає роздільний доступ до функцій для різних категорій користувачів. Це дозволяє налаштувати індивідуальні сценарії використання і

підтримувати чітку організаційну логіку.

Окрему увагу заслуговує база даних, в якій визначається логіка зберігання, доступу й фільтрації всієї ключової інформації. Структура колекцій побудована з урахуванням масштабованості, надійності та ефективності подальшої взаємодії між клієнтською частиною та сервером.

Таким чином, у другому розділі було спроєктовано логіку, зовнішній вигляд і структуру даних майбутнього вебзастосунку. Побудована модель поєднує функціональну зручність, технологічну гнучкість і візуальну цілісність, створюючи міцний фундамент для реалізації повноцінного вебсайту тенісного клубу.

3 МОДЕЛЮВАННЯ ФУНКЦІОНАЛУ ІНФОРМАЦІЙНОЇ СИСТЕМИ

При створенні проєкту інформаційної системи одним з найголовніших складових розробки є моделювання. Це поняття уособлює в собі створення абстрактного представлення майбутньої системи для структурованого аналізу і подальшого проєктування. За рахунок моделювання можна легко виявити логічні зв’язки між елементами вебзастосунку, виокремити вимоги користувачів та звичайно передбачити майбутні проблеми для подальшого забезпечення узгодженості між кожним етапом розробки.

UML (англ. Unified Modeling Language) являє собою мову уніфікованого моделювання для створення моделей в контексті розробки програмного забезпечення. Дякуючи цій мові стає можливим забезпечити візуалізацію структури вебсистеми у зрозумілій формі.

Також UML дає можливість створити діаграми, що поєднують в собі як статичні, так і динамічні аспекти. Кожна діаграма має свій тип і виконує свою роль. Наприклад, одні можуть моделювати структуру бази даних, а інші в свою чергу будуть моделювати логіку поведінки користувачів.

Для цього проєкту UML-діаграми були використані для точного представлення функціональності вебсайту тенісного клубу. Завдяки ним було спроєктовано архітектуру системи, визначено сценарії взаємодії, що стало основою реалізації клієнтської і серверної частин проєкту.

3.1 Діаграма прецедентів

На цьому етапі було створено діаграму прецедентів (англ. Use Case), яка демонструє взаємодію основних ролей користувачів із функціоналом вебзастосунку. Діаграма слугує візуальним інструментом для визначення меж системи та опису ключових сценаріїв використання з погляду кінцевого користувача.

У проєкті тенісного клубу «TL» виділено шість акторів:

– гість (незареєстрований користувач);
– учень (дитина, яка відвідує спортивну школу);
– батько/мати (законний представник дитини);
– аматор (дорослий гравець клубу без професійної підготовки);
– тренер;
– адміністратор.

Для кожної з цих ролей було визначено характерні дії, які вони можуть виконувати на вебсайті. Наприклад, гість має змогу переглядати новини, ознайомитися з адміністрацією клубу та розпочинати реєстрацію.

Після вибору ролі користувачі отримують доступ до специфічного функціоналу:

– учень: перегляд розкладу своїх тренувань, турнірів та можливість зареєструватися на них, тренерів, дошки пошани та контактних даних дитячої спортивної школи;
– батьки: доступ до розкладу дитини, перегляд тренерів, дошки пошани і контактів школи, а також бронювання кортів;
– аматор: реєстрація на турніри, бронювання кортів, перегляд тренерів та спарингів клубу;
– тренер: перегляд і керування учнями, перегляд власного розкладу, відстежування турнірів;
– адміністратор: повне редагування турнірів, кортів, учнів, новин та користувачів.

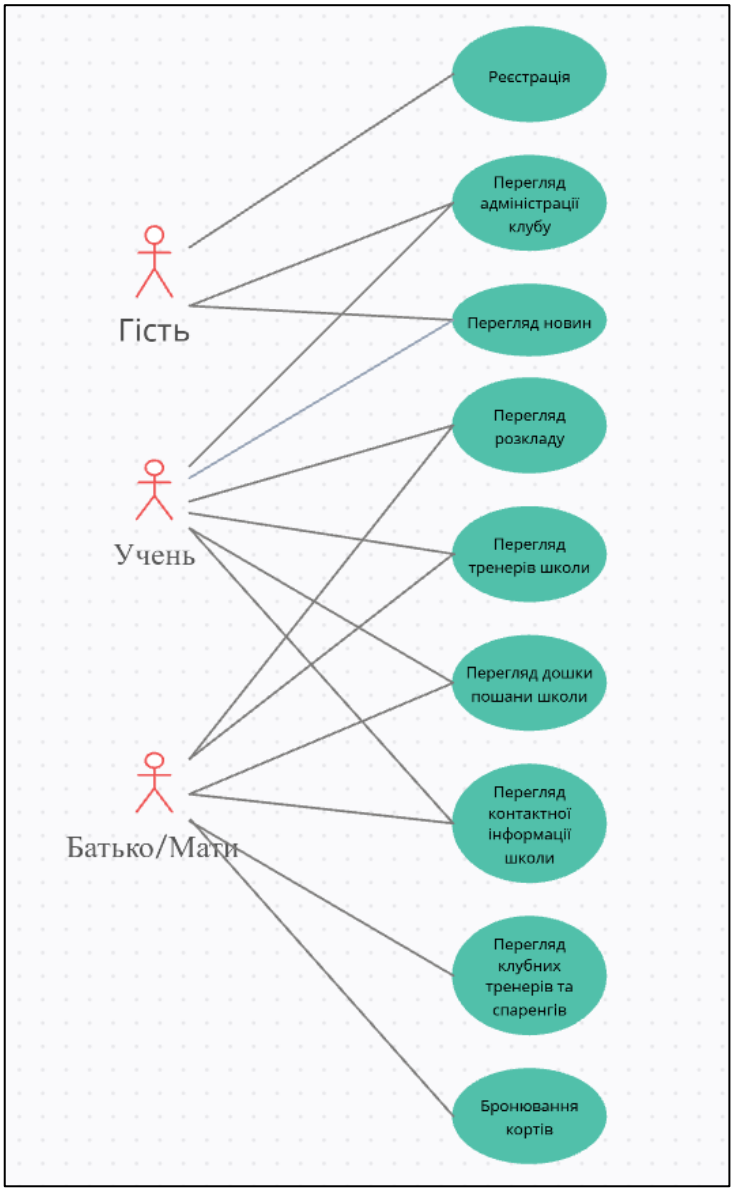


Рисунок 3.1 – Діаграма прецедентів із відображенням акторів (гість, учень, батьки) і їхніх дій у системі

Кожен прецедент представляє собою окрему дію, яку може виконувати користувач. Взаємозв’язки між актором і прецедентом відображають обсяг дозволеної взаємодії з системою. Частина сценаріїв є спільною для кількох ролей (наприклад, реєстрація чи перегляд новин), інші – суто спеціалізованими (редагування розкладу, керування бронюваннями, модерація даних).

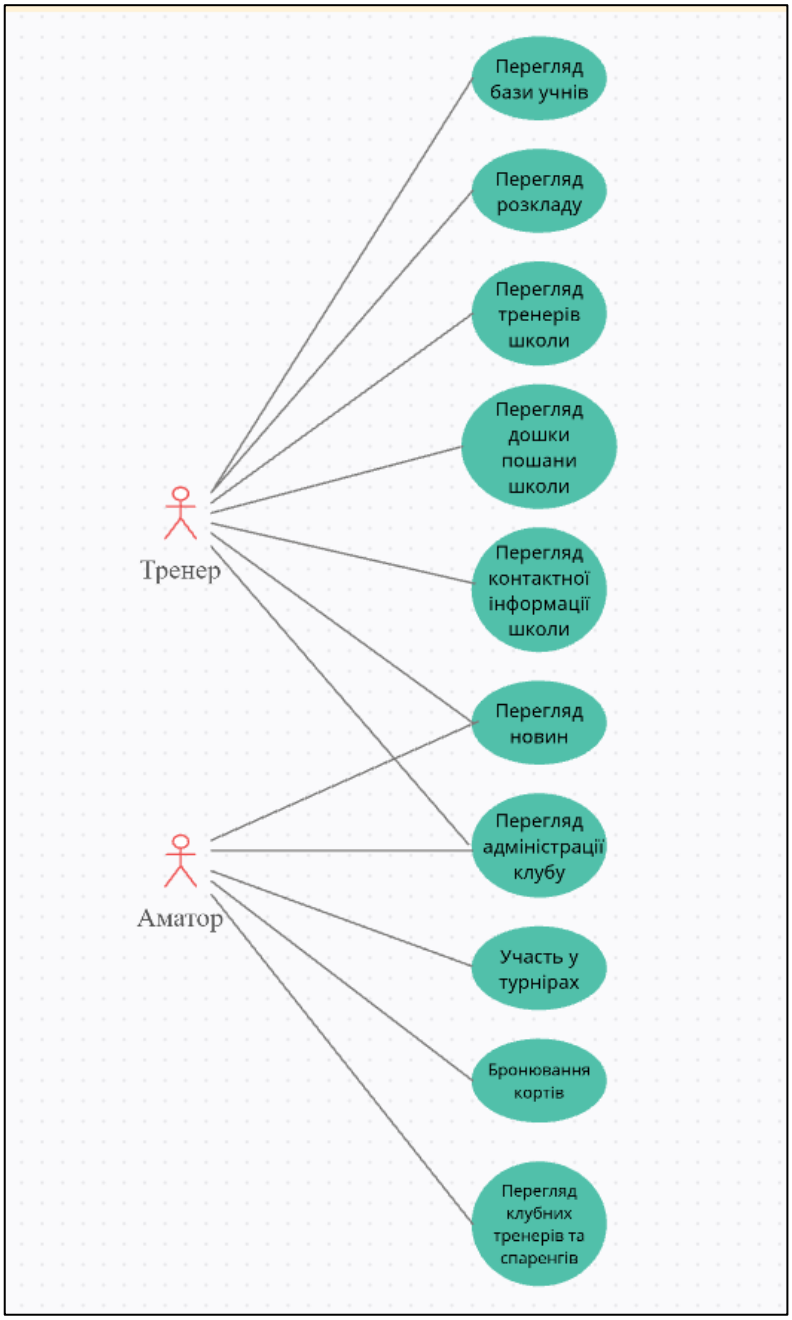


Рисунок 3.2 – Діаграма прецедентів із відображенням акторів (аматор, тренер) і їхніх дій у системі

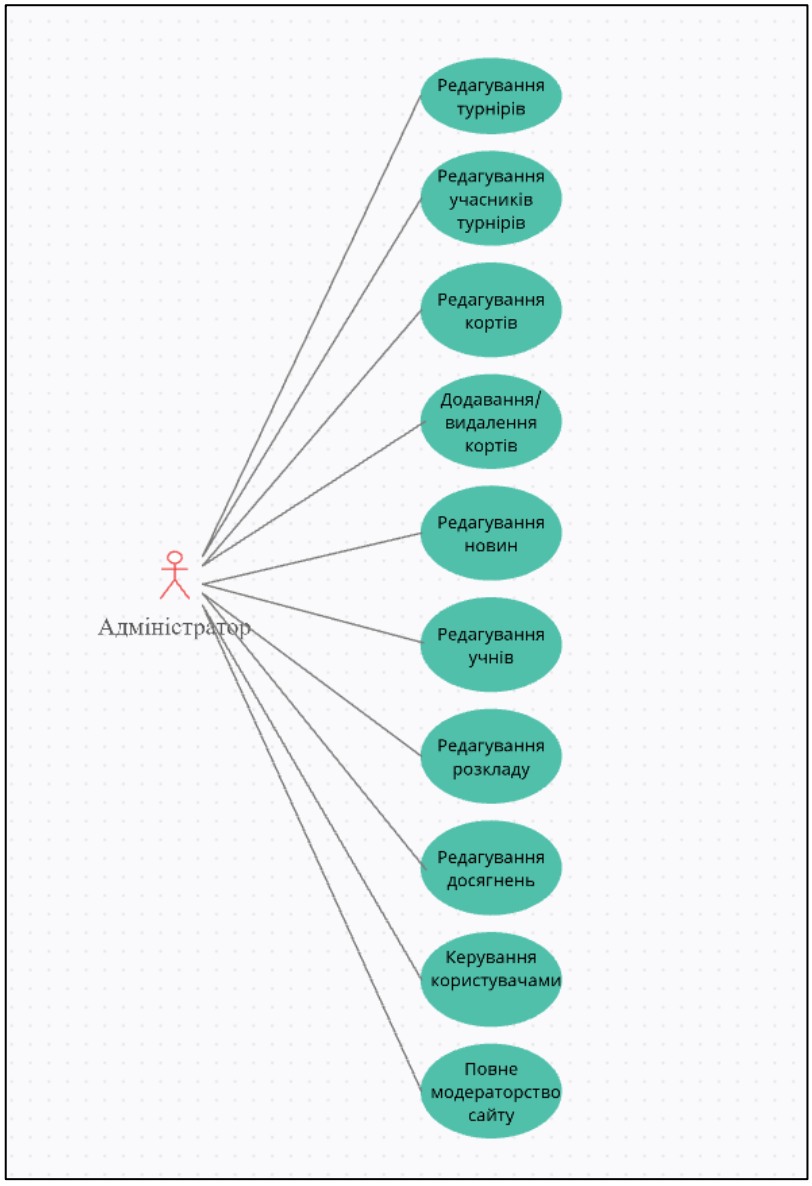


Рисунок 3.3 – Діаграма прецедентів із відображенням актора (адмін) і його дій у системі

Створена діаграма дозволяє представити ключові точки взаємодії користувача з системою і стала основою для побудови детальніших моделей активностей і послідовностей у наступних підрозділах [13].

3.2 Діаграма послідовності: реєстрація та підтвердження електронної пошти

Діаграма послідовності (англ. Sequence Diagram) моделює взаємодію між користувачем і основними компонентами системи під час створення нового облікового запису з підтвердженням електронної пошти. Цей сценарій є безумовно важливим для безпеки, унеможливлення реєстрації під вигаданими адресами та для чіткого розподілу доступу на основі ролей.

У процесі беруть участь наступні об’єкти:

– Користувач: ініціатор реєстрації;

– інтерфейс клієнта (браузер): форма реєстрації;

– сервер застосунку: обробляє дані, надсилає запити до бази даних і сервісів пошти;

– система електронної пошти: надсилає користувачу лист із кодом підтвердження;

– база даних: зберігає облікові записи та тимчасові коди верифікації.

Основна послідовність подій:

– користувач заповнює реєстраційну форму, обираючи роль, вводячи ім’я, email і пароль;

– інтерфейс надсилає ці дані на сервер;

– сервер створює тимчасовий обліковий запис і генерує код підтвердження;

– код надсилається на вказану пошту через поштовий сервіс;

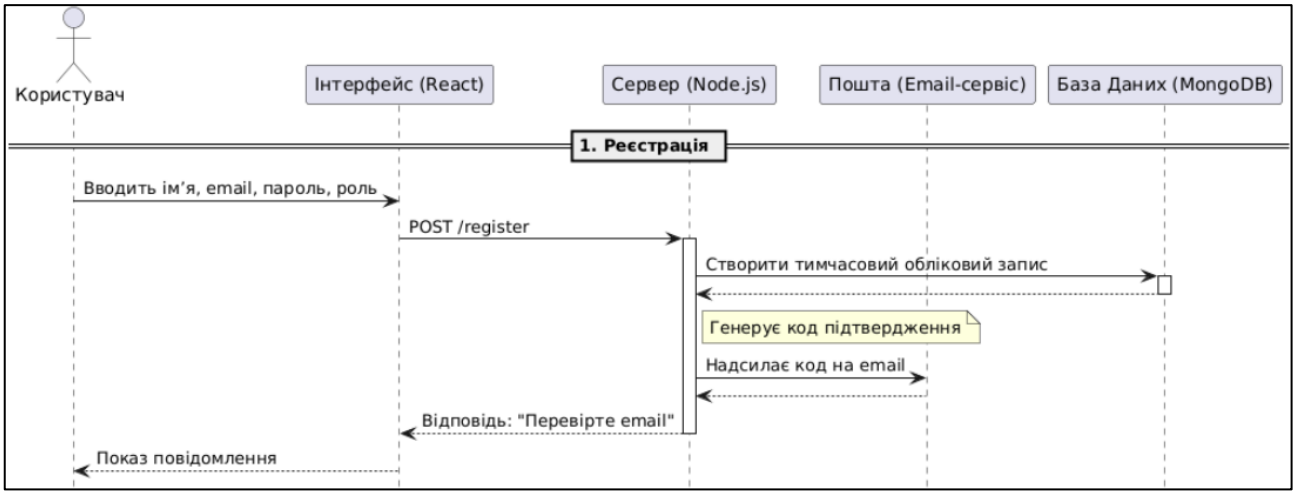
– користувач переходить до форми підтвердження і вводить код;

– інтерфейс передає код на сервер для перевірки;

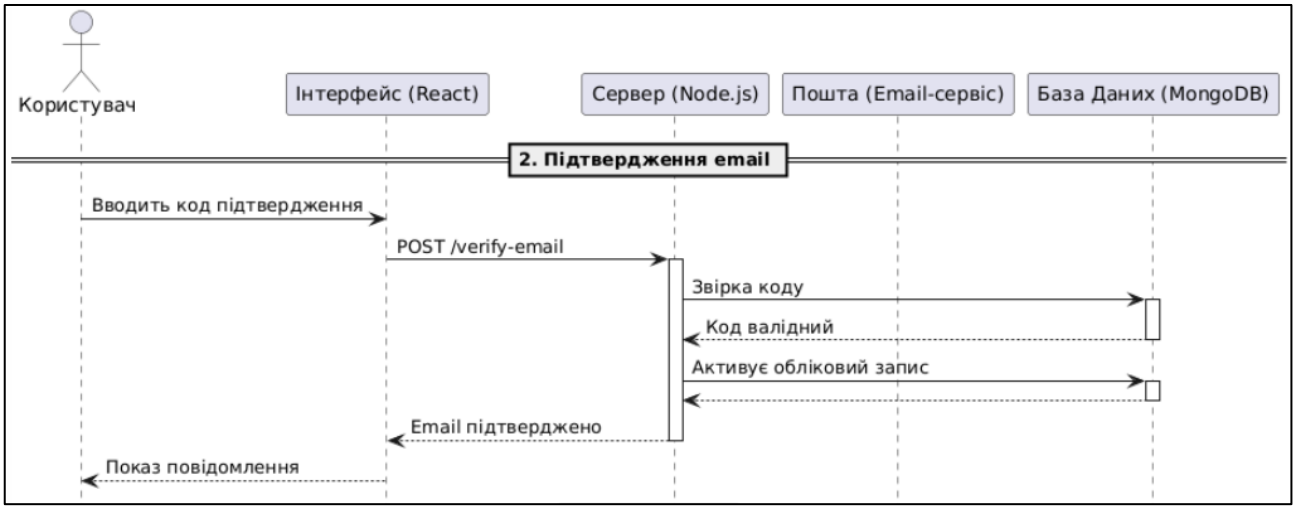
– сервер звіряє код із базою;

– у разі відповідності активується обліковий запис, користувачу повертається повідомлення про успіх;

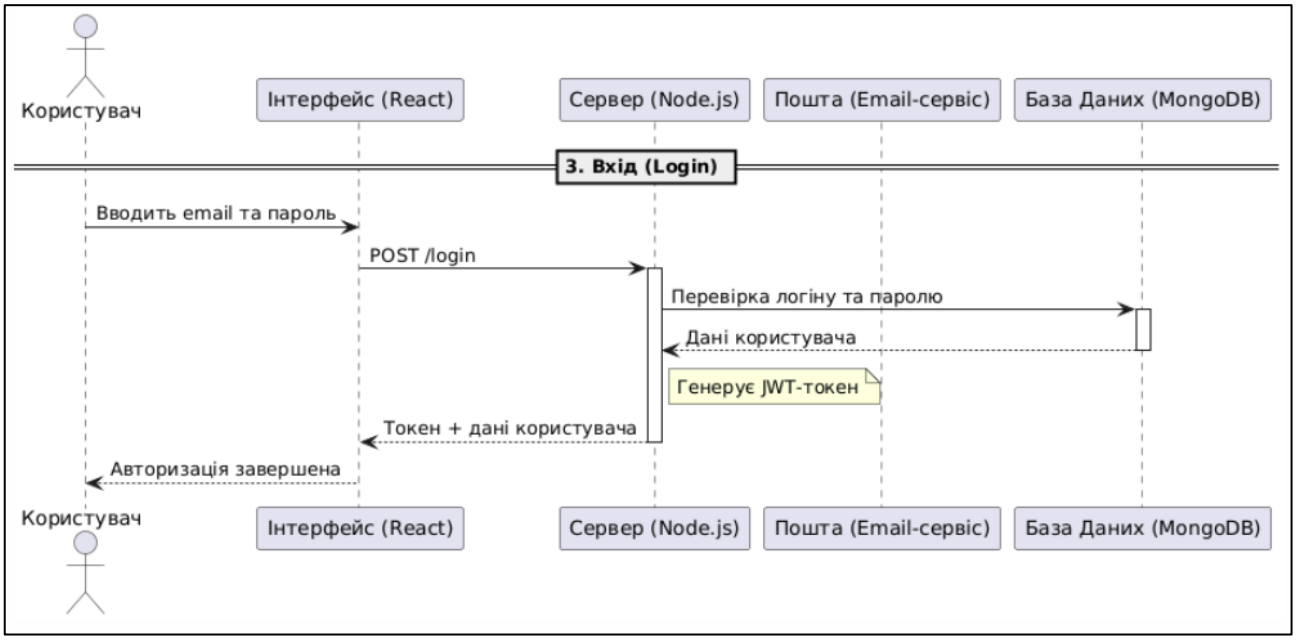
– у разі помилки в коді – повертається повідомлення про відмову.



а)



б)



в)

Рисунок 3.4 – Діаграма послідовності, що відображає обмін повідомленнями між користувачем, клієнтським інтерфейсом, сервером, поштовим сервісом та базою даних під час процесу реєстрації й підтвердження (а, б, в)

Для створення такої послідовності було написано код компонента реалізації користувача вебсайту, де було прораховано можливість перевірки email та підтвердження його через код перевірки.

```
const Registration = () => {
  const navigate = useNavigate();
  const [isRegistered, setIsRegistered] = useState(false);
  const [isVerifying, setIsVerifying] = useState(false);
  const [message, setMessage] = useState('');

  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    role: ''
  });

  const [loginData, setLoginData] = useState({
    email: '',
    password: ''
  });
```

а)

```
const [verificationData, setVerificationData] = useState({
  email: '',
  code: ''
});

useEffect(() => {
  const token = localStorage.getItem('token');
  if (token) navigate('/');
}, [navigate]);

const handleChange = (e) => {
  if (isVerifying) {
    setVerificationData({ ...verificationData, [e.target.name]: e.target.value });
  } else if (isRegistered) {
    setLoginData({ ...loginData, [e.target.name]: e.target.value });
  } else {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  }
};

const handleSubmit = async (e) => {
  e.preventDefault();

  if (isVerifying) {
    try {
      const res = await fetch('http://localhost:5000/api/auth/verify-email', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(verificationData),
      });
      const result = await res.json();
      if (!res.ok) throw new Error(result.message);

      setMessage('Електронна пошта підтверджена! Тепер ви можете увійти.');
      setIsVerifying(false);
      setIsRegistered(true);
    } catch (error) {
      setMessage(error.message);
    }
    return;
  }
```

б)

Рисунок 3.5 – Частина коду Registration.jsx (а, б)

Саме така послідовній моделі допомагає досягти кілька цілей одночасно: підтвердження користувача, збереження логіки ролей, контроль за кількістю облікових записів . Реалізація цього сценарію підвищує рівень довіри до системи, через те що кожен користувач виконує особисте підтвердження перед реєстрацією.

3.3 Діаграма активностей: процес бронювання корту

Діаграма активностей (англ. Activity Diagram) моделює поетапний процес

взаємодії користувача з вебсайтом під час бронювання тенісного корту. Цей сценарій є одним із головних у функціоналі клубу, тому що пов’язаний з прямою участю користувача в організації тренувального або змагального процесу.

Доступ до вкладки з бронюванням надається лише після автентифікації в системі і тільки користувачам з ролями батько/мати чи аматор. Усі дії відбуваються на одній сторінці, без оновлення або переходу між екранами, що забезпечує швидкість і зручність взаємодії.

Основна логіка процесу включає такі кроки:

– користувач авторизується в системі та переходить на вкладку бронювання;

– вибирається тип покриття (грунт або хард), після чого завантажується список доступних кортів;

– залежно від обраного корту, система пропонує вибрати дату;

– після вибору дати з’являється список вільних часових інтервалів у вигляді чекбоксів;

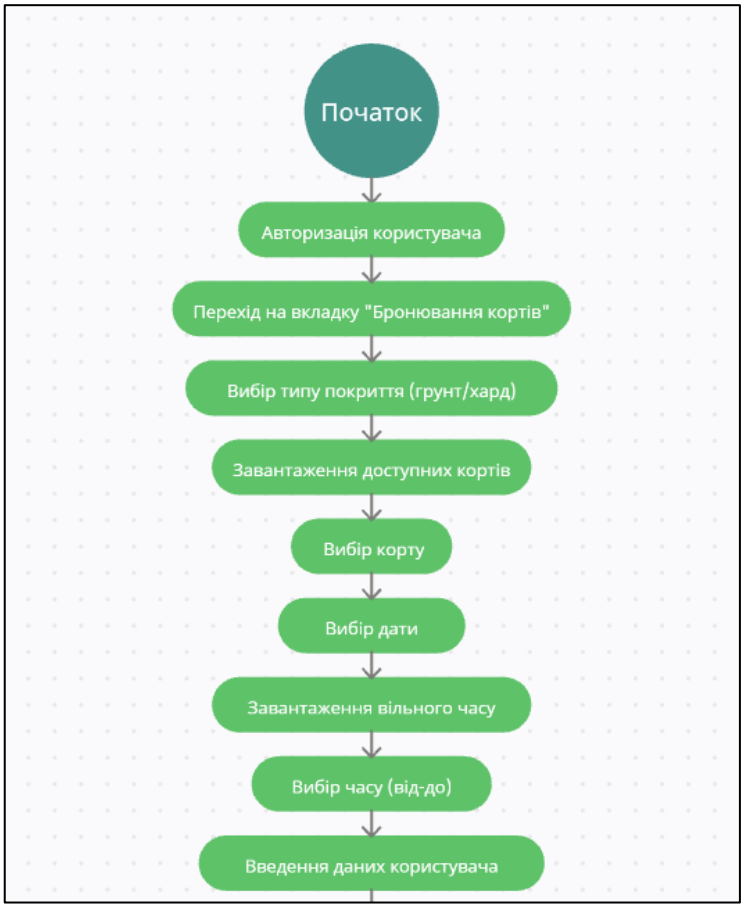
– користувач обирає одне або кілька доступних значень часу (у форматі від-до);

– після заповнення даних та натискання кнопки «забронювати» система проводить перевірку заповнення полів;

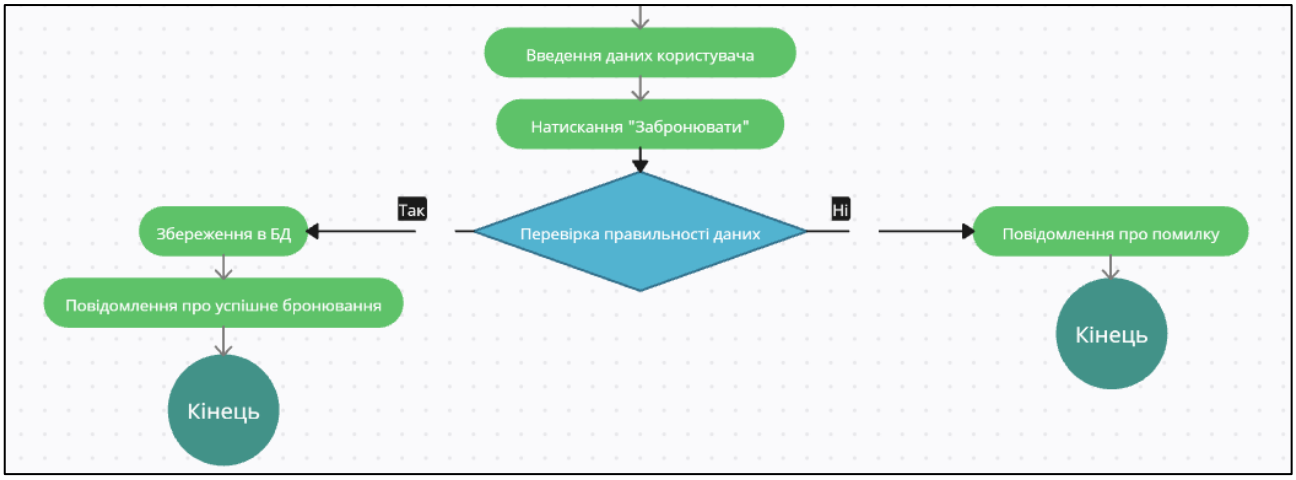
– якщо дані введено коректно, бронювання зберігається в базі даних;

– користувач отримує повідомлення про успішне бронювання або помилку в разі конфлікту часу чи збоїв.

Сценарій також передбачає варіанти скасування або редагування бронювання через вкладку «Мої бронювання», де відображаються лише записи, пов’язані з поточним користувачем.



а)



б)

Рисунок 3.6 – Діаграма активностей, що відображає послідовність дій користувача під час створення бронювання, з розгалуженням на перевірку доступності часу та виведення повідомлення про результат (а, б)

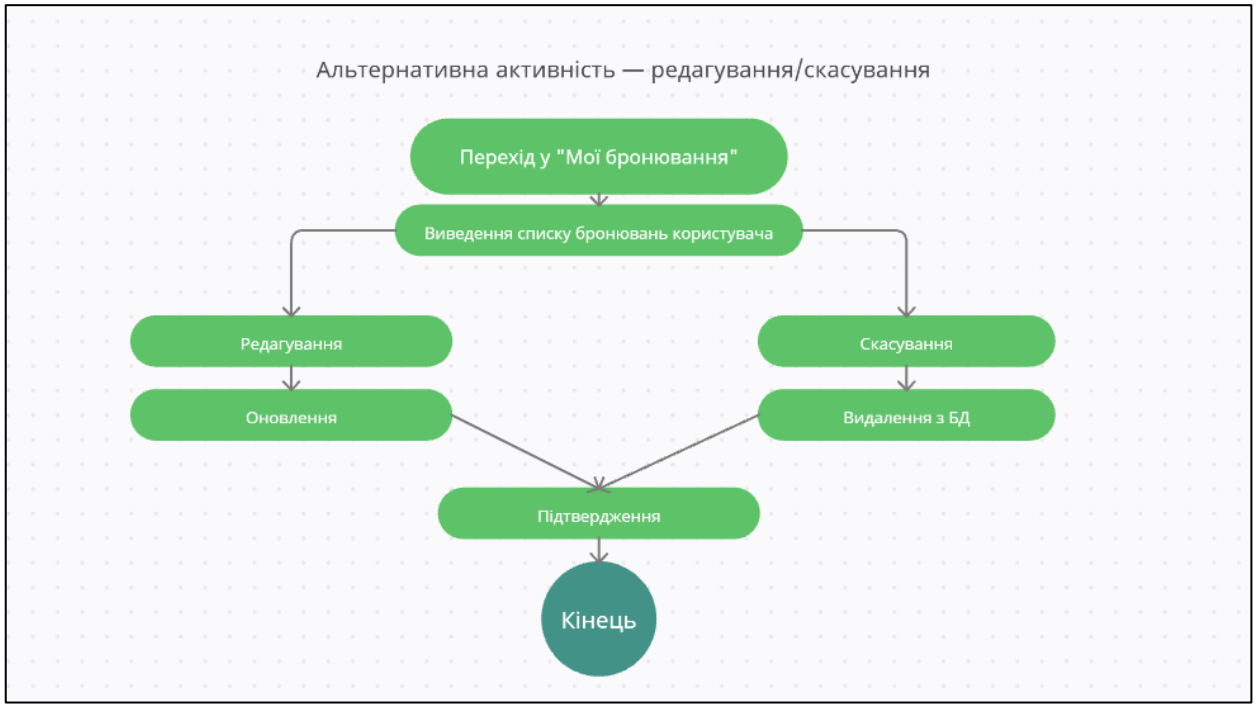


Рисунок 3.7 – Діаграма активностей, що відображає послідовність дій користувача під час альтернативної активності – редагування/скасування

Проаналізована активність гарно демонструється у логіці бронювання кортів, де можна спостерігати доступні динамічні слоти з часом та зберіганням результату у БД.

```
const cancelBooking = async (courtId, date, times) => {
    try {
      for (const time of times) {
        await fetch(`http://localhost:5000/api/courts/${courtId}/cancel`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ date, time })
        });
      }
      setMessage('❌ Бронювання скасовано');
      fetchBookingsByName(userName);
      fetchCourts();
    } catch (err) {
      setMessage('Помилка при скасуванні');
    }
  };
```

а)

```
const updateBooking = async () => {
  const { courtId, originalDate, originalTimes, newDate, newTimes } = editingBooking;
  try {
    for (const time of originalTimes) {
      await fetch(`http://localhost:5000/api/courts/${courtId}/cancel`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ date: originalDate, time })
      });
    }
    for (const time of newTimes) {
      await fetch(`http://localhost:5000/api/courts/${courtId}/book`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ userName, date: newDate, time })
      });
    }
    setMessage('✅ Бронювання оновлено');
    setEditingBooking(null);
    fetchBookingsByName(userName);
    fetchCourts();
  } catch (err) {
    setMessage('❌ Не вдалося оновити');
  }
};
```

б)

Рисунок 3.8 – Код з файлу CourtBooking.jsx (а, б)

Створена діаграма дозволяє візуально відстежувати загальний процес, ураховувати можливі гілки перевірки. Також вона лягла в основу логіки реалізації клієнтської частини відповідного інтерфейсу. Саме поетапна побудова сценарію бронювання забезпечує його зручність, швидкість і захист від помилкових записів або дублювання.

3.4 Діаграма класів: структура основних сутностей

Діаграма класів (англ. Class Diagram) є основним допоміжним інструментом для візуального подання логічної структури даних у проєктованій інформаційній системі. Вона дає змогу описати основні об'єкти, їх атрибути та взаємозв’язки між ними. Подібний варіант підходу допомагає упорядкувати зв’язки між базовими елементами ще до реалізації в базі даних або коді.

Для вебзастосунку тенісного клубу було побудовано загальну модель класів, яка охоплює чотири основні сутності: Користувач, Корт, Бронювання та Турнір. Ці основи є базовими для налагодження процесів взаємодії між системою та користувачами.

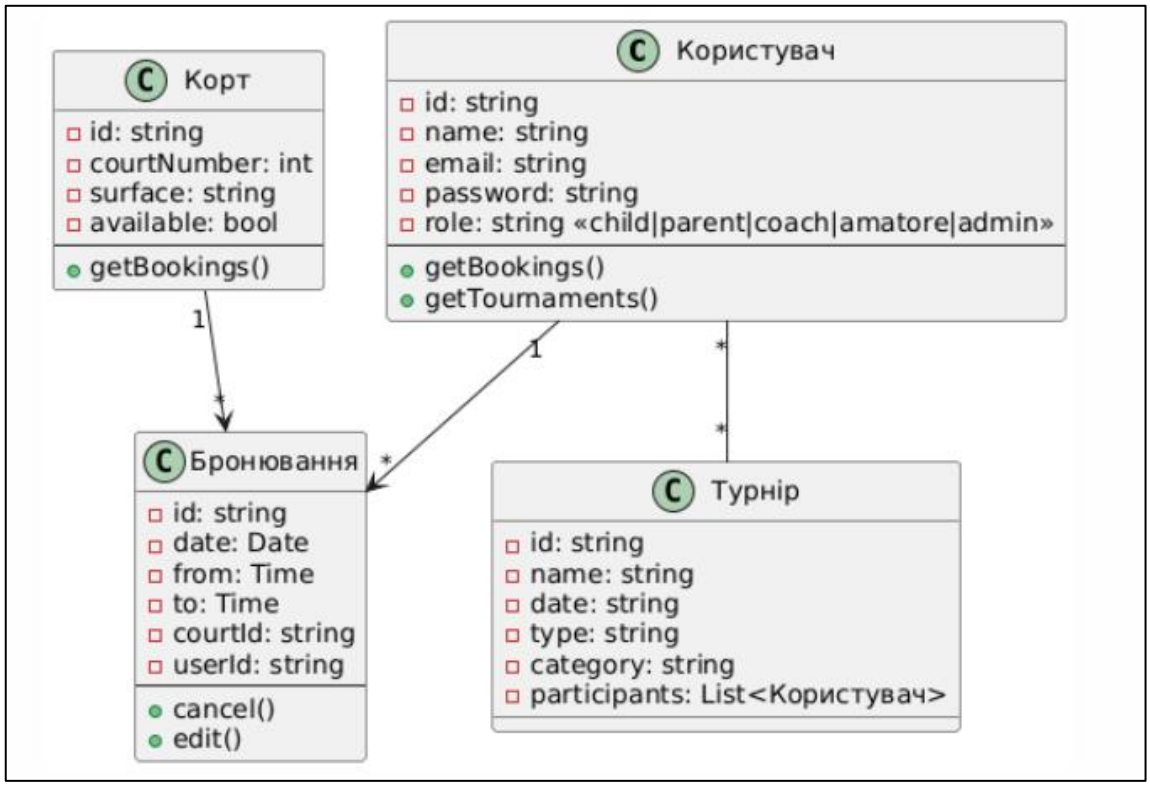


Рисунок 3.9 – Діаграма класів, яка демонструє сутності Користувач, Корт, Турнір і Бронювання, із зазначенням атрибутів і зв’язків між ними.

У структурі:

– користувач має атрибути: ID, ім’я, email, пароль, роль. Кожен користувач може мати багато бронювань і може бути учасником одного або кількох турнірів;

– корт має атрибути: ID, номер, тип покриття, доступність. Один корт може містити багато бронювань, але кожне бронювання прив’язане до конкретного корту;

– турнір має атрибути: назва, дата, тип, категорія, список учасників. Кожен користувач може брати участь у кількох турнірах, а кожен турнір має багато учасників – зв'язок типу «багато до багатьох»;

– бронювання виконує роль проміжної сутності, яка зв'язує користувача з кортом і зберігає інформацію про дату, час початку та завершення. Таким чином, вона формує n-арний зв'язок між користувачем, кортом і часовим інтервалом.

Ця структура була виконана в MongoDB за допомогою бібліотеки Mongoose. Наприклад коди схем для колекцій User і Tournament.

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, required: true, enum: ["amatore", "coach", "child", "parent"] },
  isVerified: { type: Boolean, default: false },
  verificationCode: { type: Number },
}, { timestamps: true });

module.exports = mongoose.model('User', UserSchema);
```

Рисунок 3.10 – Код моделі User

```
const mongoose = require('mongoose');

const tournamentSchema = new mongoose.Schema({
  name: String,
  date: String,
  location: String,
  category: String,
  type: String,
  participants: Number,
  draw: String,
  participantsList: {
    boys: [String],
    girls: [String],
    list: [String]
  },
  details: String
});

module.exports = mongoose.model('Tournament', tournamentSchema);
```

Рисунок 3.11 – Код моделі Tournament

Завдяки такій схемі стало можливим ефективно реалізувати функції системи: автентифікацію, динамічне створення бронювань, участь у турнірах, а також гнучке управління ресурсами з боку адміністратора. Діаграма класів також допомогла на етапі формування колекцій бази даних у MongoDB та побудови запитів у REST API.

3.5 Діаграма станів: життєвий цикл турніру

Діаграма станів (англ. State Machine Diagram) була побудована для відображення змін, які відбуваються з об’єктом турніру впродовж його існування. Вона демонструє, як система реагує на події, пов’язані з турнірами, і які можливі переходи між станами цього об’єкта.

На відміну від інших діаграм, діаграма станів фокусується не на користувачах чи компонентах, а саме на внутрішній логіці життєвого циклу.

Турнір може переходити між такими станами:

– створено – стан після створення запису адміністратором або системою;

– відкрито для реєстрації – учасники можуть надсилати заявки;

– зареєстровано учасників – досягнуто ліміту або завершено реєстрацію вручну;

– активний – турнір триває, відбуваються матчі;

– завершено – турнір закінчено, результати збережено;

– архівований/видалений – неактивний стан, не показується користувачам.

Можливі переходи розпочинаються як діями адміністратора (наприклад, відкриття/закриття реєстрації), так і автоматичними подіями (настання дати початку).

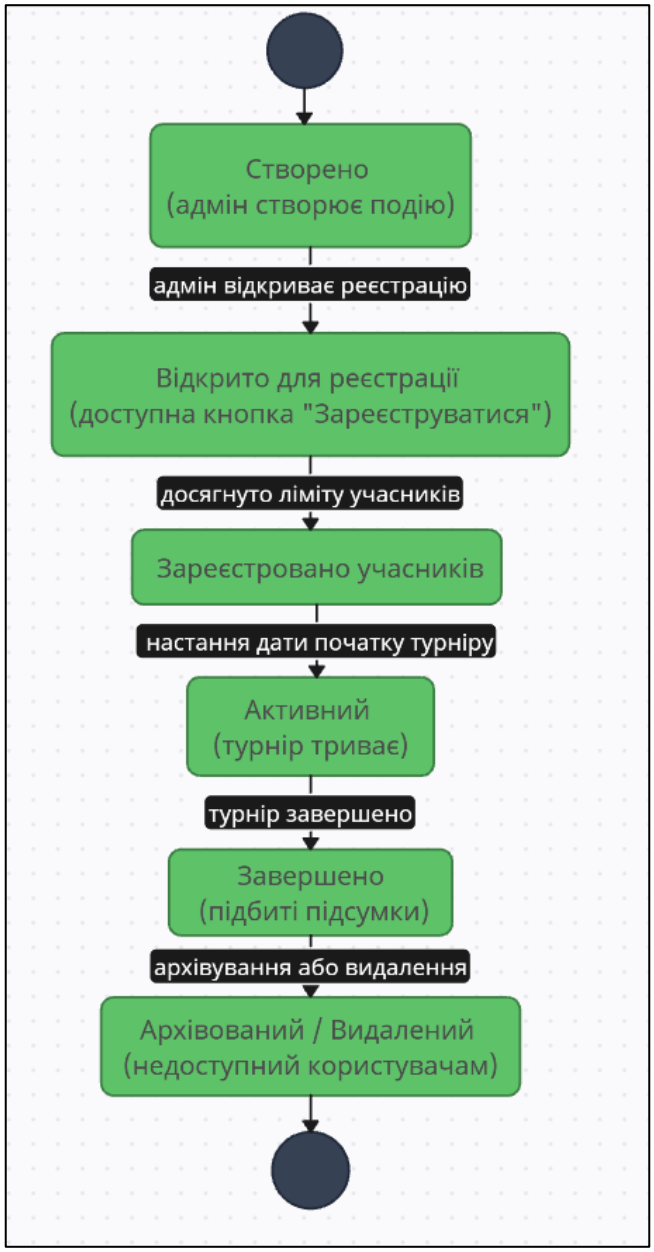


Рисунок 3.12 – Діаграма станів для турніру, яка ілюструє основні стани об’єкта та події, що викликають переходи між ними

Наявність чітко визначених станів дозволяє уникати логічних конфліктів під час взаємодії з турнірами, а також підвищує цілісність і передбачуваність роботи системи.

3.6 Діаграма компонентів: архітектура системи

Для повного розуміння структури вебзастосунку було створено діаграму компонентів (англ. Component Diagram), яка демонструє побудову системи з погляду її основних модулів та взаємодії між ними. Такий тип діаграми дозволяє візуально представити, з яких частин складається система, як вони пов’язані між собою, які дані передаються між модулями та якими інтерфейсами.

Система вебсайту тенісного клубу реалізована як класичний SPA-застосунок (Single Page Application) з архітектурою клієнт-сервер. До складу компонентної моделі входять:

– клієнтська частина (Frontend) реалізована на React. Відповідає за візуальну частину, маршрутизацію сторінок, відображення розкладу, бронювань, реєстраційних форм, турнірів тощо. Всі дії користувача розпочинаються саме з цієї частини. Вона звертається до API-сервера для виконання запитів;

– серверна частина (Backend API) побудована на платформі Node.js з використанням Express. Приймає запити з клієнта, виконує перевірку прав доступу, обробку даних, взаємодіє з базою даних та зовнішніми сервісами, наприклад розсилка підтвердження email;

– база даних (MongoDB) зберігає всю основну інформацію про користувачів, учнів, турніри, бронювання, новини тощо. Дані організовано в колекціях;

– зовнішні сервіси. Email-сервер – використовується для надсилання кодів підтвердження реєстрації, а JWT-механізм – для створення токенів доступу, що використовуються у запитах як підтвердження авторизованого користувача.

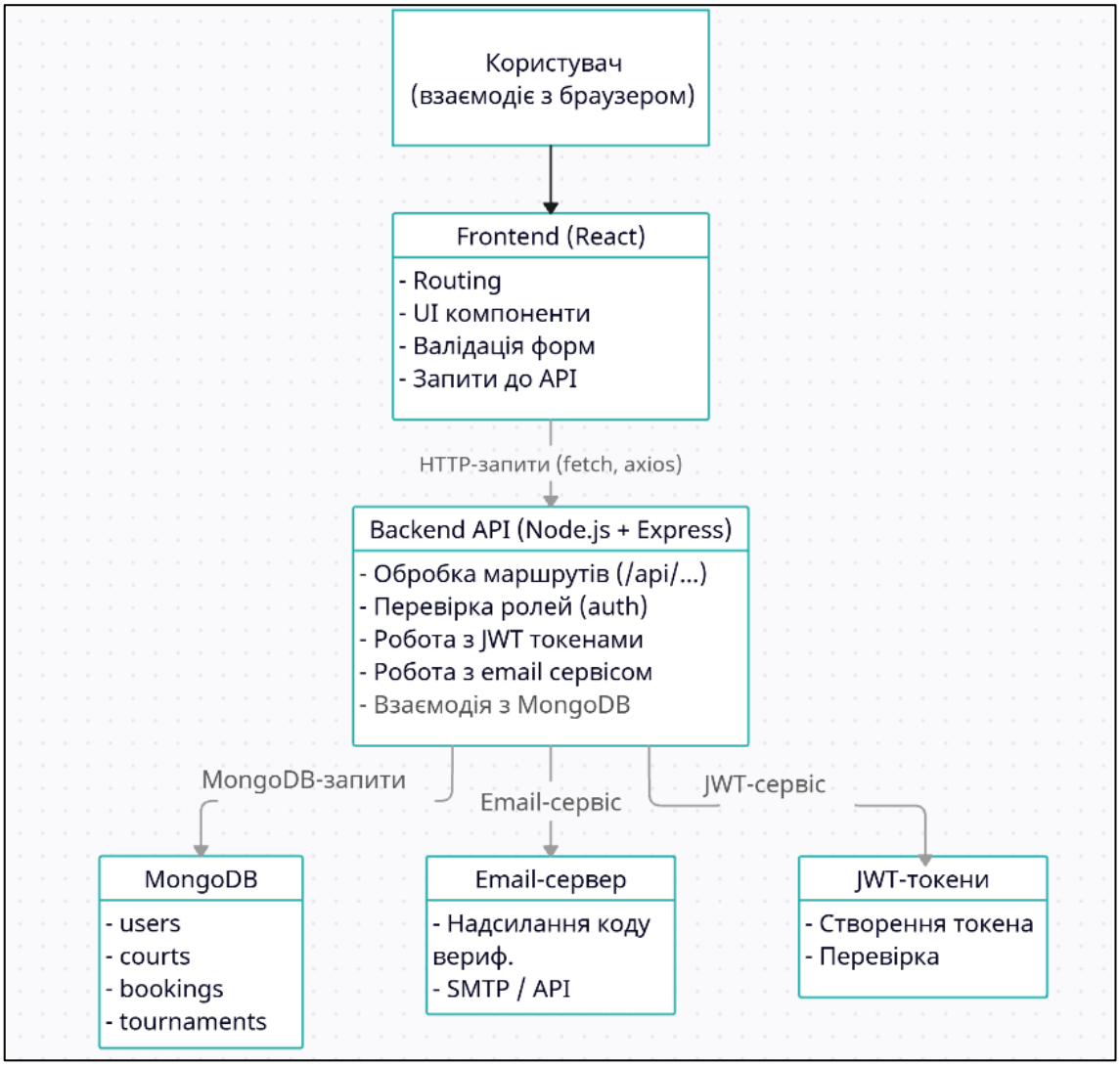


Рисунок 3.13 – Діаграма компонентів із зображенням зв’язків між Frontend, Backend, MongoDB, email-сервером та токен-системою

Завдяки поділу системи на незалежні модулі досягається гнучкість і ефективність у розробці та підтримці, масштабованість, а також можливість розширення функціоналу без впливу на інші частини. Така архітектура також дозволяє легко інтегрувати нові сервіси (наприклад, аналітику або платіжну систему) у майбутньому.

Висновки до розділу 3

У третьому розділі було детально розібрано і показано, як працює інформаційна система тенісного клубу (її функції) і як вона організована (її структура). Це було зроблено за допомогою об'єктно-орієнтованого моделювання, що є способом створення моделей, який допомагає все чітко представити. Застосування UML-діаграм дозволило візуально представити логіку роботи сайту на різних рівнях – від поведінки користувача до взаємодії внутрішніх компонентів.

Побудована діаграма прецедентів окреслила межі системи, продемонструвавши основні сценарії використання для кожної з ролей. Діаграма активностей дала змогу детально змоделювати процес бронювання кортів, а діаграма послідовності допомогла відобразити кроки взаємодії між користувачем, клієнтським інтерфейсом, сервером та поштовим сервісом під час реєстрації.

Класова діаграма структурно описала сутності, що лежать в основі бази даних, а діаграма станів розкрила життєвий цикл турніру в системі. Завершальною стала діаграма компонентів, яка представила логічну архітектуру вебзастосунку, з урахуванням клієнтської, серверної частини, бази даних і зовнішніх сервісів.

Усі побудовані моделі стали теоретичною основою для переходу до етапу практичної реалізації, дозволяючи чітко сформулювати вимоги до кожного програмного модуля та забезпечити узгодженість усіх рівнів системи – від бази даних до інтерфейсу користувача.

4 РЕАЛІЗАЦІЯ ОБРАНИХ ТЕХНОЛОГІЙ З АНАЛІЗОМ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Створення клієнтської частини вебсайту за допомогою React

Створення клієнтської частини вебзастосунку тенісного клубу було виконано за допомогою бібліотеки React, що дало можливість реалізувати динамічний та ефективний інтерфейс. Цей вибір дав змогу зробити інтерфейс зручним, адаптивним і підтримуючим різні взаємодії для кожної ролі користувачів [21].

Основою розробки є React-компонент, який контролює конкретну частину інтерфейсу вебсайту. За маршрутизацію між сторінками системи відповідає бібліотека react-router-dom, що робить роботу навігації без перезавантаження сторінок.

Кожна роль користувача була реалізована через окрему вкладку в особистому кабінеті:

– тренер має можливість зайти на сторінку дитячої спортивної школи, де в нього є вкладки власного розкладу, контактної інформації школи, досягнення, своїх учнів та бази учнів школи, а також зайти на сторінку турнірів де він може відслідковувати як дитячі так і аматорські турніри ;

– учень може переглядати у своєму кабінеті розклад, дізнатись інформацію про тренерів школи та контактну інформацію, а також має доступ до дошки пошани;

– аматор має доступ до сторінки з тренерами та спарингами тенісного клубу та сторінок бронювання кортів або реєстрації на аматорські турніри;

– батько або мати мають доступ до розкладу їх дитини, також можуть переглядати всю інформацію про школу що і їх діти, та бронювати корти для них.

Роль активного користувача використовується на вебсайті для виводу відповідного контенту або контролю доступу до нього. Доступ до певних сторінок або вкладок обмежено перевіркою ролі активного користувача.

Інтерфейс реалізовано за допомогою потужних систем компонування flex і grid, які дозволили гнучко та ефективно розмістити різні елементи на вебсторінках. Було додатково додано візуальне виділення активних вкладок, кнопок дій, наприклад «Зареєструватися», «Забронювати», «Редагувати» тощо. Для повідомлень про успішні або помилкові дії також було додано візуалізацію.

```jsx
import React from 'react';
import { Link } from 'react-router-dom';
import './CoachPanel.css';

const CoachPanel = () => {
  return (
    <div className="coach-panel">
      <h1>Панель тренера</h1>
      <div className="coach-options">
        <Link to="/tournaments" className="option-card option-tournaments">
          ТУРНІРИ
        </Link>
        <Link to="/sports-school" className="option-card option-school">
          ДИТЯЧА СПОРТИВНА ШКОЛА
        </Link>
      </div>
    </div>
  );
};

export default CoachPanel;
```

Рисунок 4.1 – Приклад коду компонента CoachPanel.jsx

```jsx
return (
  <header className="site-header">
    <div className="logo">
      <Link to="/">
        <img src={logo} alt="Tennis Club TL Logo" className="site-logo" />
      </Link>
      <Link to="/" className="site-title">
        <h1>Тенісний клуб TL</h1>
      </Link>
    </div>

    <button
      className="hamburger"
      onClick={() => setMenuOpen(!menuOpen)}
      aria-label="Toggle menu"
    >
      {menuOpen ? <FaTimes /> : <FaBars />}
    </button>
```

а)

```
      <nav className={`main-nav ${menuOpen ? 'open' : ''}`}>
        <ul>
          {user?.name && <li className="user-info">Вітаємо, {user.name}!</li>}
          {user?.role === 'coach' && <li><Link to="/coach-panel"><FaChalkboardTeacher /> Панель тренера</Link></li>}
          {user?.role === 'child' && <li><Link to="/child-panel"><FaChild /> Кабінет учня</Link></li>}
          {user?.role === 'parent' && <li><Link to="/parent-panel"><FaUser /> Інформація для батьків</Link></li>}
          {user?.role === 'amatore' && <li><Link to="/amatore-panel"><FaRunning /> Кабінет аматора</Link></li>}
          {user?.role === 'admin' && <li><Link to="/admin-panel"><FaUsersCog /> Панель адміністратора</Link></li>}
          {!isAdmin && (
  <>
    <li><Link to="/blog"><FaNewspaper /> Блог/Новини</Link></li>
    <li><Link to="/administration"><FaUsersCog /> Адміністрація клуба</Link></li>
  </>
)}


          {!user ? (
            <li><Link to="/registration"><FaRegAddressCard /> Реєстрація</Link></li>
          ) : (
            <li>
              <button className="logout-button" onClick={handleLogout}>
                Вийти
              </button>
            </li>
          )}
        </ul>
      </nav>
    </header>
  );
};

export default Header;
```

б)

Рисунок 4.2 – Фрагмент Header.jsx (а, б)

За взаємодію з бекендом відповідають fetch та axios. До http://localhost:5000/api/ йдуть всі запити, де знаходяться потрібні маршрути. Після чого всі дані записуються у useState-стан і далі використовуються у вебсистемі.

Саме реалізація React-компонентів дала можливість швидко масштабувати вебзастосунок, а саме можливість зміни або додавання вкладок, зміна дизайну сторінок або навіть для під’єднання логіки не потрібно було переробляти всю структуру, що надає вибраному шляху велику перевагу.

4.2 Створення серверної частини вебсайту за допомогою Node.js та бази даних MongoDB

Серверна частина вебзастосунку було розроблена у Node.js разом з фреймворком Express.js [23]. Такий вибір зумовлений високою продуктивністю, легкістю масштабування та легкою інтеграцією з клієнтською частиною, яка було реалізована на React. За допомогою Express.js було створено RESTful API, що представляє собою архітектурний стиль для створення вебсайтів, який підтримує взаємодію клієнта із сервером, використовуючи для цього стандартні методи HTTP, при цьому взаємодіючи з базою даних [24].

Реалізація бази даних у проєкті було зроблена як докуметоорієнтована структура, у якій всі основні дані, такі як користувачі, турніри, бронювання та корти, зберігаються у своїх колекціях у вигляді JSON-подібних документів. Взаємодія з базою даних була реалізована за допомогою бібліотеки Mongoose, що дало можливість чітко і зрозуміло розробити схеми та моделі даних. Такий підхід призвів до уникнення некоректних записів і забезпечив цілісність інформації на вебсайті [22].

Рисунок 4.3 чітко відображає приклад запису в колекції User, який містить ім’я та прізвище користувача, email, пароль, який був автоматично захешованний та роль, яку обрав користувач при реєстрації, що надає доступ до вибіркових функцій сайту.

Рисунок 4.3 – Mongoose-схема для користувача

Детальна інформація про учнів міститься у колекції students. Там можна побачити поля: ім’я, прізвище, дату народження, розряд, статус і тренера учня. Ці дані використовуються для складання розкладу, бази учнів школи та реєстрації на турніри.

Рисунок 4.4 – Структура документа student у MongoDB Compass зі збереженими даними про учня, статус та ім’я тренера

Маршрути для бронювання тенісних кортів відіграють визначальну роль у проєкті. Для кожного окремого бронювання існує перевірка, яка визначає чи є вільним вибраний користувачем час. При ситуації в якій обраний проміжок часу користувачем вже існує, і в БД є відповідний запис, то запит користувача на це бронювання відхиляється з відповідним повідомленням. При успішному бронюванні, відбувається запис до масиву бронювань відповідного корту.

```
router.post('/:id/book', async (req, res) => {
  const { userName, date, time } = req.body;

  if (!userName || !date || !time) {
    return res.status(400).json({ message: 'Всі поля обов'язкові' });
  }

  try {
    const court = await Court.findById(req.params.id);
    if (!court) return res.status(404).json({ message: 'Корт не знайдено' });

    const isTaken = court.bookings.some(b => b.date === date && b.time === time);
    if (isTaken) return res.status(400).json({ message: 'Цей час уже заброньований' });

    court.bookings.push({ userName, date, time });
    await court.save();

    res.status(200).json({ message: 'Бронювання успішне', court });
  } catch (err) {
    res.status(500).json({ message: 'Помилка при бронюванні', error: err.message });
  }
});

// POST /api/courts/:id/cancel — скасувати бронювання
router.post('/:id/cancel', async (req, res) => {
  const { date, time } = req.body;
  try {
    const court = await Court.findById(req.params.id);
    if (!court) return res.status(404).json({ message: 'Корт не знайдено' });

    const canceledBookings = court.bookings.filter(b => b.date === date && b.time === time);
    for (const booking of canceledBookings) {
      const user = await User.findOne({ name: booking.userName });
      if (user?.email) {
        await sendEmail(user.email, "bookingDeleted", user, {
          date: booking.date,
          from: booking.time,
          to: booking.time,
          courtNumber: court.courtNumber
        });
      }
    }

    court.bookings = court.bookings.filter(b => !(b.date === date && b.time === time));
    await court.save();

    res.status(200).json({ message: 'Бронювання скасовано', court });
  } catch (err) {
    res.status(500).json({ message: 'Помилка при скасуванні', error: err.message });
  }
});
```

Рисунок 4.5 – Фрагмент обробника POST /api/bookings,
запис у масив бронювань

Розроблена структура файлової організації бекенду стала важливим аспектом реалізації проєкту. Увесь код був логічно розділений на такі блоки, як моделі, маршрути, контролери. Такий підхід дуже сильно спростила підтримку коду, а також

дозволила зробити багаторазове використання логіки у відповідних частинах вебсайту.

На основі JWT-токенів було реалізовано усю логіку автентифікації. Ці токени генеруються при вході в аккаунт користувача та зберігаються у клієнтській частині. Через серверну складову проходить перевірка наявності і коректності токена для кожного запиту, і вже після цього в залежності від ролі користувача надається чи не надається доступ до певних API-ендпоінтів.

Отже, можна зробити висновок, що реалізована серверна частина надає надійне зберігання та обробку всіх даних та всієї інформації вебзастосунку. Саме завдяки стеку з Node.js, Express.js та MongoDB вдалося зробити стабільну роботу вебсайту з безпечним доступом до даних і з гнучкою архітектурою, яка придатна до розширення.

4.3 Інтеграція основних модулів: авторизація, особисті кабінети, розклад, турніри, бронювання кортів

Наступним кроком розробки вебсайту після реалізації клієнтської та серверної частин стало приєднання основних функціональних модулів, що формують собою повну роботу вебзастосунку. Авторизація, особисті кабінети, бронювання кортів, реєстрація на турніри – саме ці складові повинні працювати синхронно та адаптуватися під кожного користувача.

Основою розробленої вебсистеми є функція авторизації. ЇЇ робота реалізована через JWT-токени. Після входу користувача до особистого кабінету, в залежності від ролі обраної при реєстрації (coach, student, parent, amator), вебзастосунок динамічно підбирає потрібний інтерфейс, тобто видає доступ тільки до тих сторінок і вкладок, перегляд яких дозволений поточному користувачеві.

```
<nav className={`main-nav ${menuOpen ? 'open' : ''}`}>
        <ul>
          {user?.name && <li className="user-info">Вітаємо, {user.name}!</li>}
          {user?.role === 'coach' && <li><Link to="/coach-panel"><FaChalkboardTeacher /> Панель тренера</Link></li>}
          {user?.role === 'child' && <li><Link to="/child-panel"><FaChild /> Кабінет учня</Link></li>}
          {user?.role === 'parent' && <li><Link to="/parent-panel"><FaUser /> Інформація для батьків</Link></li>}
          {user?.role === 'amatore' && <li><Link to="/amatore-panel"><FaRunning /> Кабінет аматора</Link></li>}
          {user?.role === 'admin' && <li><Link to="/admin-panel"><FaUsersCog /> Панель адміністратора</Link></li>}
          {!isAdmin && (
  <>
    <li><Link to="/blog"><FaNewspaper /> Блог/Новини</Link></li>
    <li><Link to="/administration"><FaUsersCog /> Адміністрація клуба</Link></li>
  </>
```

Рисунок 4.6 – Фрагмент коду перевірки ролі користувача в компоненті Header.jsx

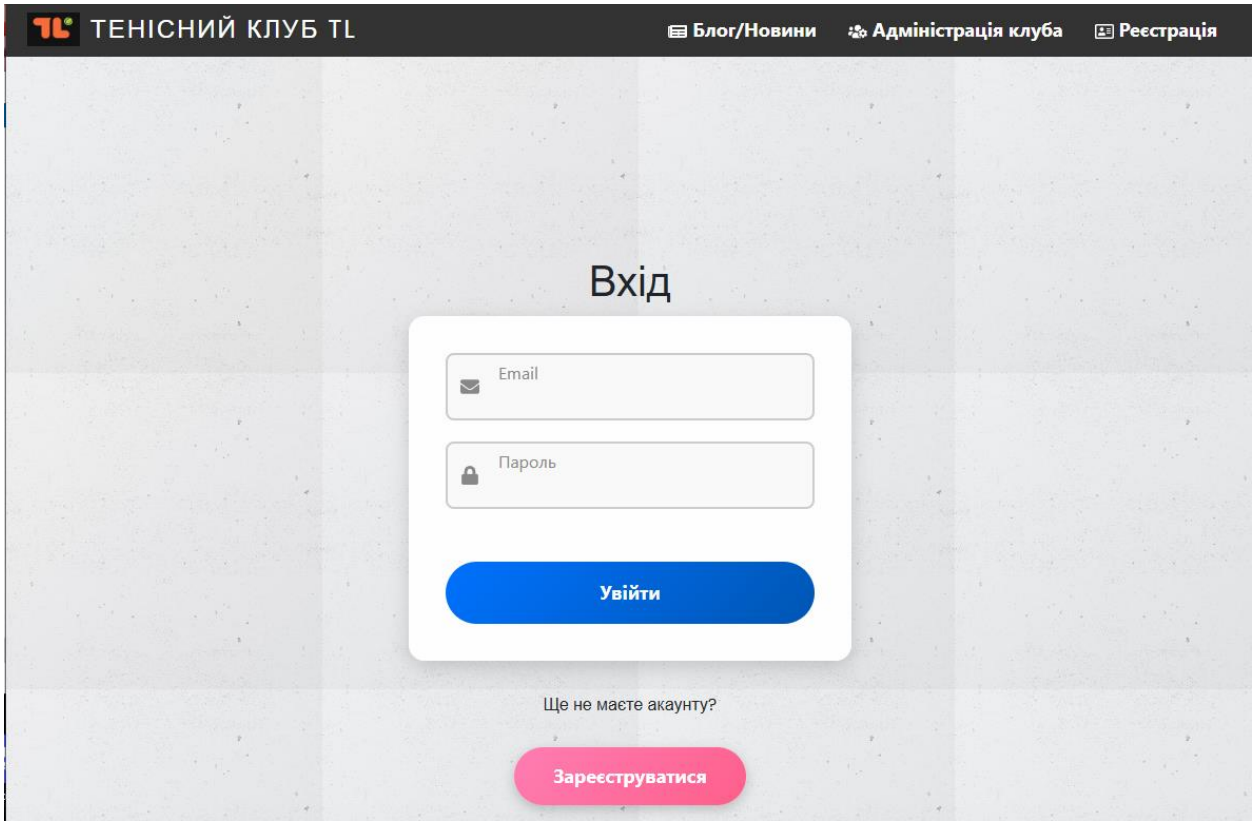


Рисунок 4.7 – Інтерфейс сторінки входу з реалізованою формою авторизації

Далі, закінчивши зі входом користувач перенаправляється до особистого кабінету, в якому реалізовано набір вкладок, кожна з яких має свій особистий вміст відповідно до ролі поточного відвідувача вебсайту. Наприклад, для тренера

виконується вивід вкладок його учнів та розкладу, для батьків це звичайно розклад їх дитини, доступні тренери та контакти дитячої спортивної школи, а у аматорів доступні бронювання кортів та реєстрація на аматорські турніри.

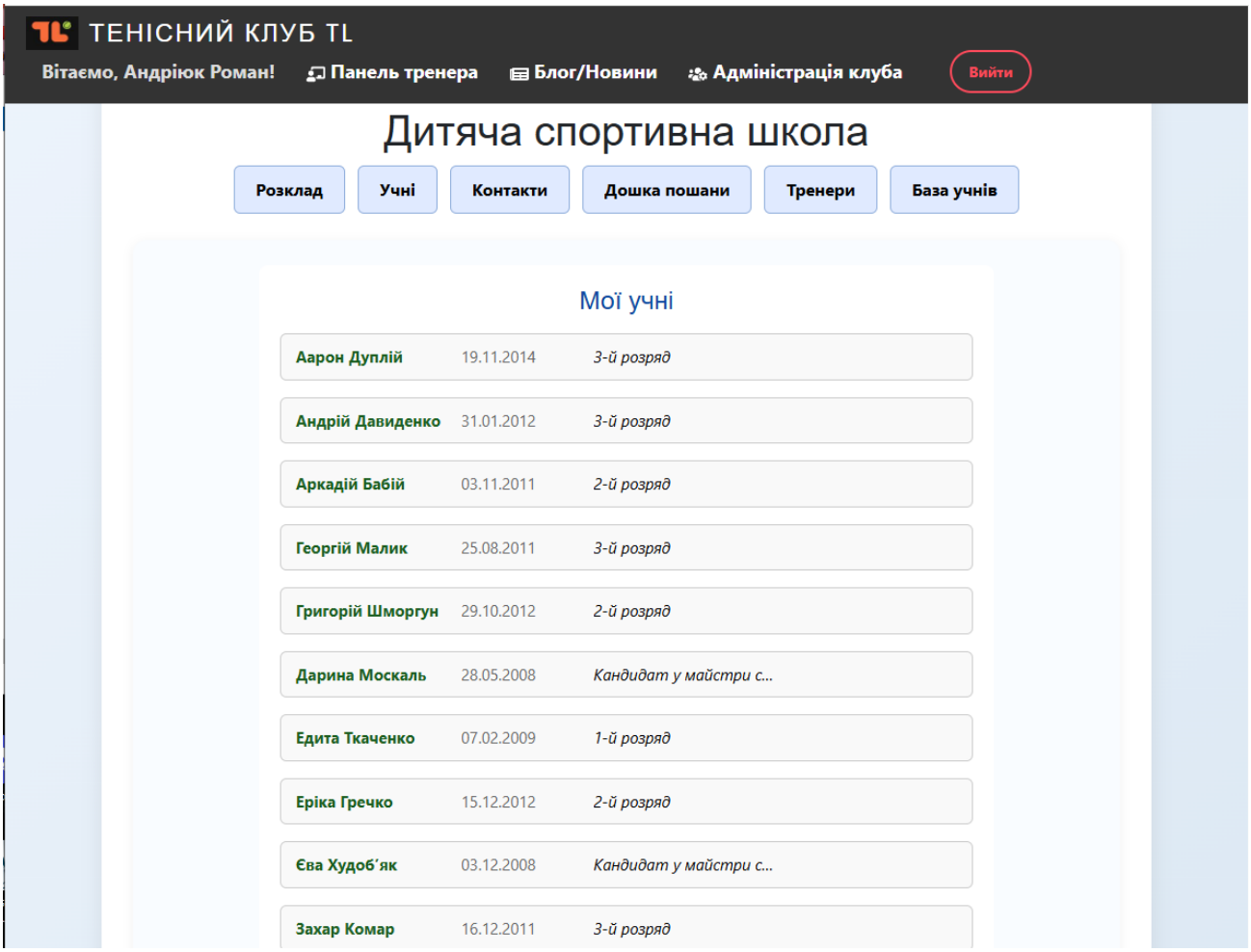


Рисунок 4.8 – Особистий кабінет тренера з активною вкладкою «Мої учні»

Найскладнішим і водночас найбільш динамічнішим модулем є розклад тренувань. Його реалізація основується на принципі розподілу учнів за їх тренерами. Після передачі списку учнів з БД до вебзастосунку, у кожного тренера створюється унікальний розклад тренувань з усіма його учнями. Діти ж в свою чергу можуть бачити лише свій розклад, так само і їх батьки. Розклад розбивається по дням тижня, що робить візуальне орієнтування легким і ефективним.

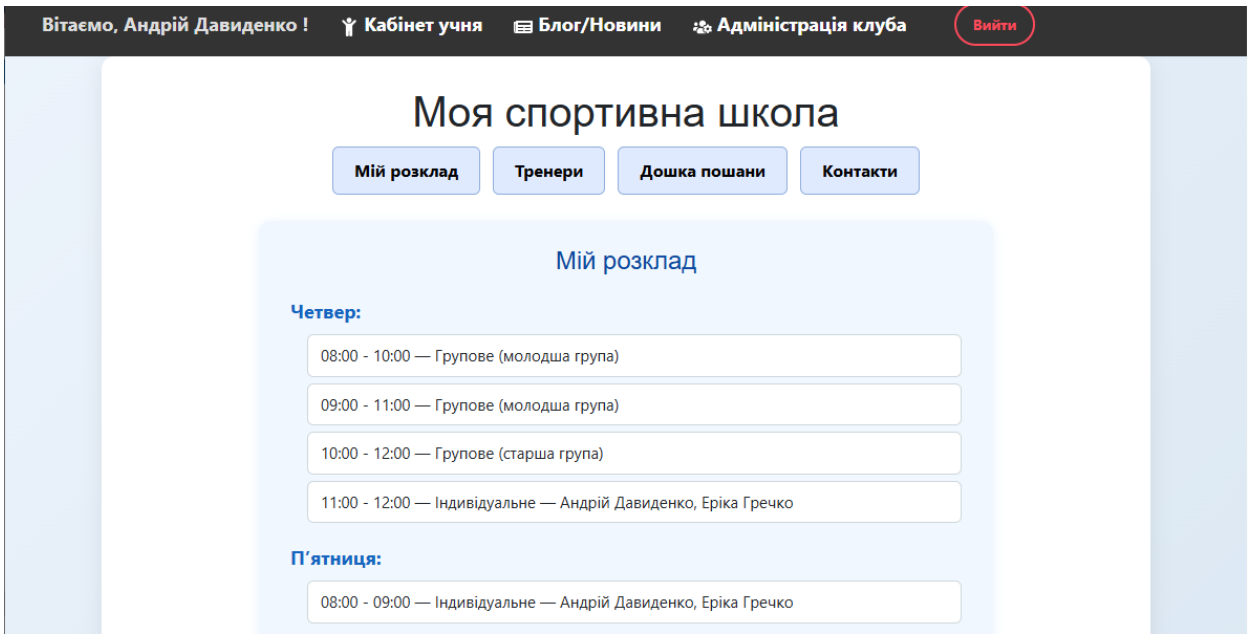


Рисунок 4.9 – Вкладка «Мій розклад» для учня

```
const groupScheduleByDay = (schedule) => {
    return schedule.reduce((acc, item) => {
      if (!acc[item.day]) acc[item.day] = [];
      acc[item.day].push(item);
      return acc;
    }, {});
  };

 const getStudentFilteredSchedule = () => {
  if (!currentUser || currentUser.role !== "child") return [];

  const matchedStudent = allStudents.find((s) =>
    `${s.firstName} ${s.lastName}`.toLowerCase().trim() === currentUser?.name?.toLowerCase().trim()
  );

  if (!matchedStudent) return [];

  const fullName = `${matchedStudent.firstName} ${matchedStudent.lastName}`;
  const coach = matchedStudent.coach;

  return mySchedule.filter((item) => {
    if (item.type === "Індивідуальне" && item.students?.includes(fullName)) {
      return true;
    }
    if (item.type === "Групове" && item.coach === coach) {
      return true;
    }

    return false;
  });
};
```

Рисунок 4.10 – Фрагмент коду компонента ChildrenSportsSchool.jsx

Для турнірного модулю було передбачено розподіл змагань на дитячі та аматорські. На сторінку передаються турніри зі списку, який знаходиться в БД і показуються лише відповідні до ролі користувача турніри. Є можливість переглянути детальнішу інформацію про турнір – дату, локацію, учасників. Для цього реалізовані кнопки «Детальніше» та «Переглянути учасників». Діти та аматори мають доступ до реєстрації на свої турніри, коли тренера в свою чергу можуть лише передивлятись списки учасників всіх турнірів.

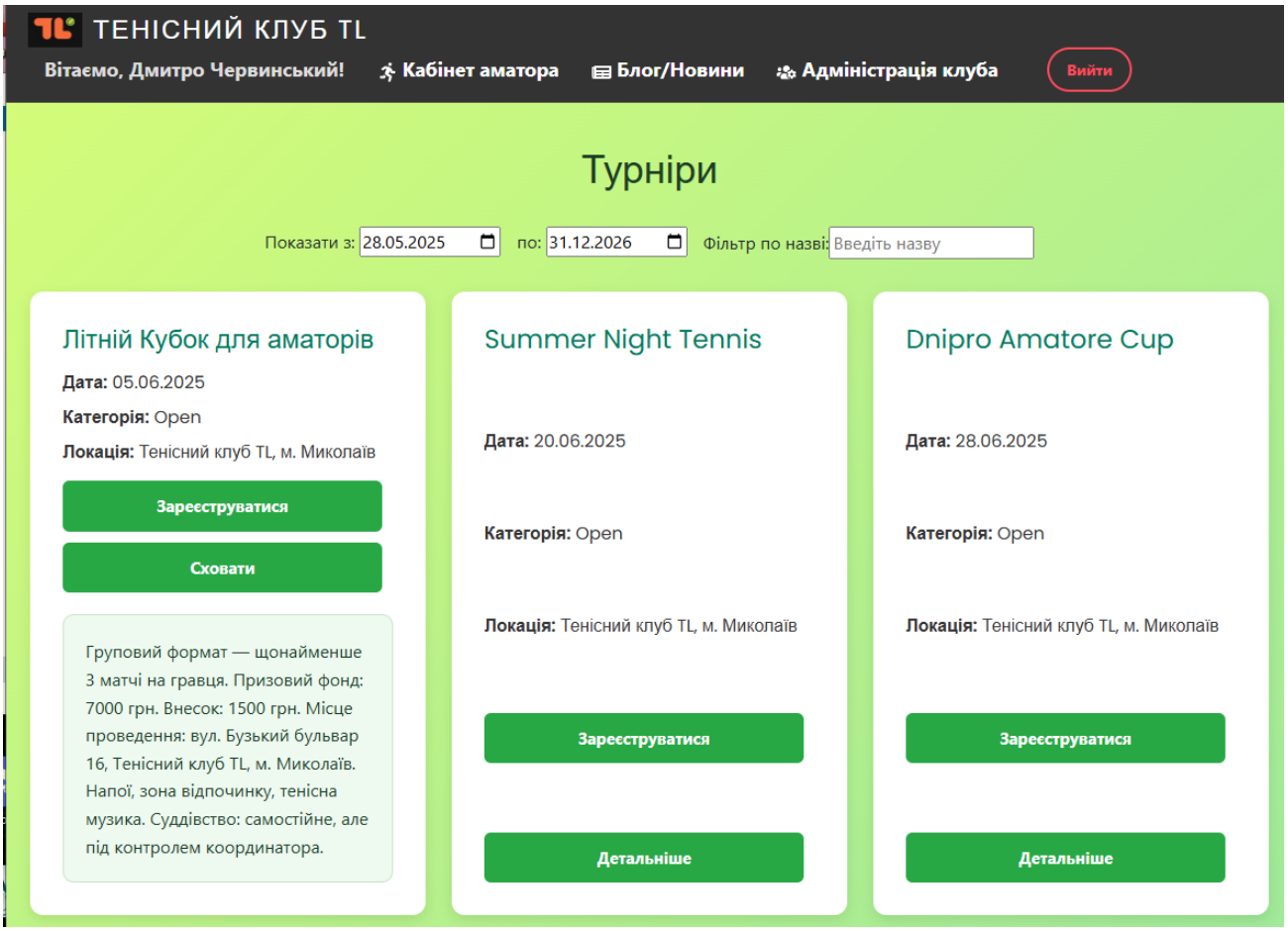


Рисунок 4.11 – Інтерфейс турнірів для аматора
з активним розгортанням інформації

Також одним із найважливіших модулів є бронювання кортів, для якого було реалізоване динамічне оновлення доступних годин на кортах. Клієнт може обрати дат, час та корт для свого тренування. Після перевірки доступності вибраних годин, відбувається запис бронювання у БД.

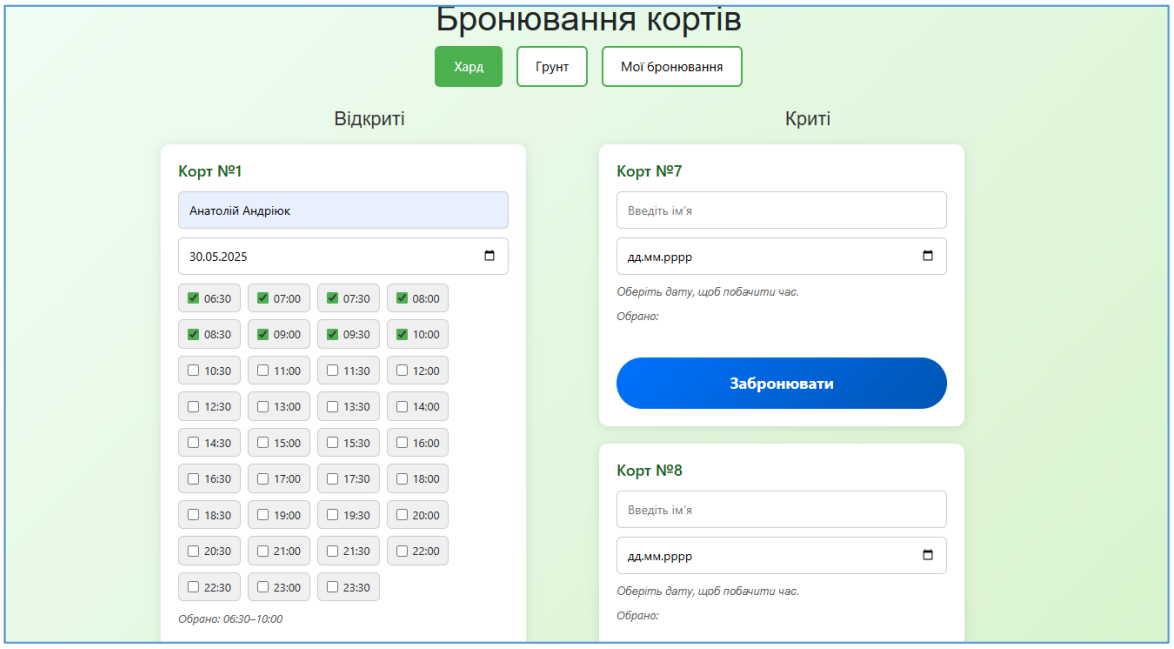


Рисунок 4.12 – Інтерфейс сторінки бронювання кортів

```
    return (
      <form key={court._id} className="booking-form" onSubmit={(e) => handleSubmit(court._id, e)}>
        <h3>Корт №{court.courtNumber}</h3>
        <input type="text" name="userName" required placeholder="Введіть ім'я" value={formData[court._id]?.userName || ''} onChange={(e) => handleChange(court._id, e)} />
        <input type="date" name="date" required value={formData[court._id]?.date || ''} onChange={(e) => handleChange(court._id, e)} />
        {date ? available.length > 0 ? (
          <div className="time-checkboxes">
            {available.map(hour => (
              <label key={hour} className="time-option">
                <input type="checkbox" checked={formData[court._id]?.selectedTimes?.includes(hour) || false} onChange={() => handleCheckbox(court._id, hour)} />{hour}
              </label>
            ))}
          </div>
        ) : <p className="no-times">На обрану дату вільного часу немає.</p> : <p className="no-times">Оберіть дату, щоб побачити час.</p>}
        <div className="selected-range">Обрано: {getTimeRange(formData[court._id]?.selectedTimes || [])}</div>
        {user?.role !== 'admin' && (
  <button type="submit">Забронювати</button>
)}
{user?.role === 'admin' && (
  <div className="admin-court-buttons">

    <button type="button" onClick={() => deleteCourt(court._id)}>🗑 Видалити</button>
  </div>
)}

      </form>
    );
  });

  return (
    <div className="court-booking-page">
      <h1>Бронювання кортів</h1>
      <div className="surface-tabs">
        <button onClick={() => setSelectedTab('хард')} className={selectedTab === 'хард' ? 'active' : ''}>Хард</button>
        <button onClick={() => setSelectedTab('грунт')} className={selectedTab === 'грунт' ? 'active' : ''}>Грунт</button>
        {user?.role !== 'admin' && (
  <button onClick={() => setSelectedTab('бронювання')} className={selectedTab === 'бронювання' ? 'active' : ''}>Мої бронювання</button>
)}
```

Рисунок 4.13 – Фрагмент коду з CourtBooking.jsx

Для редагування, перегляду чи відміни бронювання була реалізована вкладка «Мої бронювання» для кожного користувача. Для відображення записів потрібно увести ім’я активного користувача, після чого видається список усіх бронювань цієї людини.

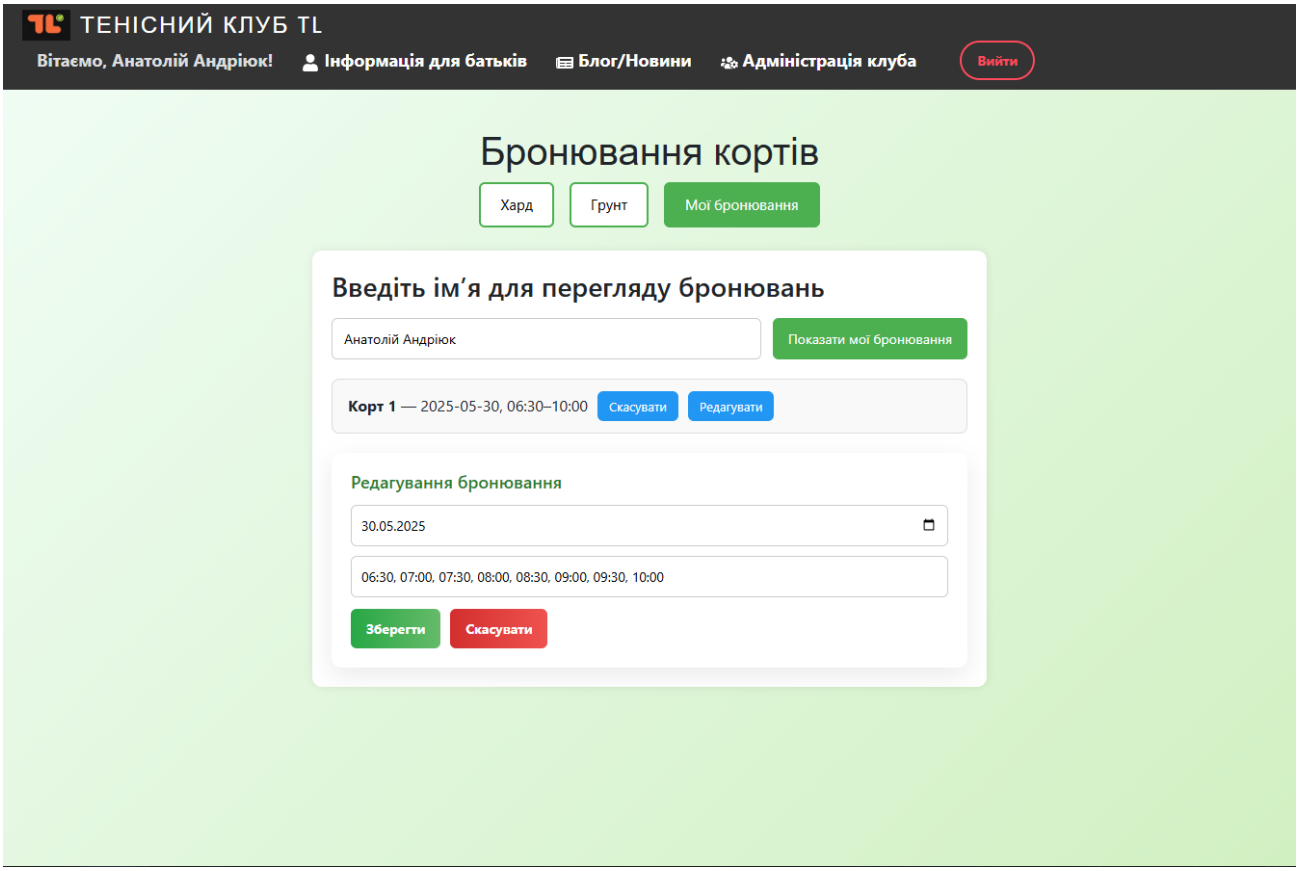


Рисунок 4.14 – Вкладка «Мої бронювання» з можливістю скасування або редагування запису

Також на вебсайті передбачено для адміністратора додаткові можливості, такі як додавання нових кортів, редагування їхніх назв, керування турнірами та учнями тощо. Адмінпанель представляє собою кабінет з розширеними можливостями та окремими кнопками для кожної сторінки вебсайту.

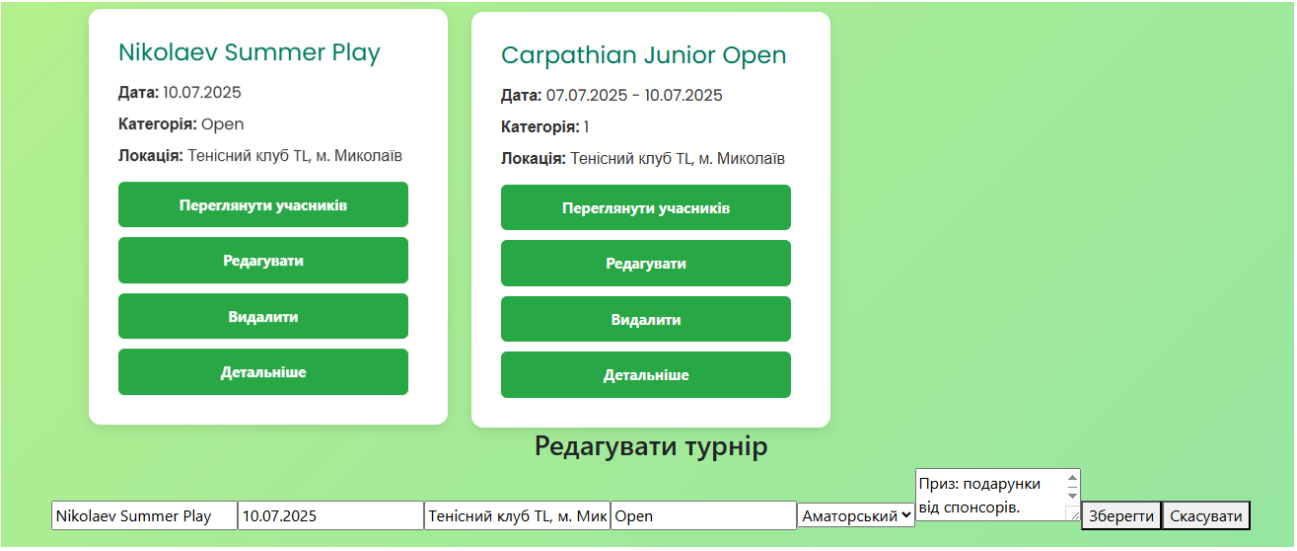


Рисунок 4.15 – Інтерфейс адміністративної панелі
з активним режимом редагування турніру

```
const handleAddTournament = async () => {
    const res = await fetch('http://localhost:5000/api/tournaments', {
      method: 'POST', headers: {'Content-Type': 'application/json'},
      body: JSON.stringify(newTournament)
    });
    if (res.ok) {
      alert('✅ Турнір додано');
      setNewTournament({ name: '', date: '', location: '', category: '', type: 'child', details: '' });
      window.location.reload();
    } else {
      alert('❌ Помилка при додаванні');
    }
  };

  const handleSaveEdit = async () => {
    await fetch(`http://localhost:5000/api/tournaments/${editingTournament._id}`, {
      method: 'PUT', headers: {'Content-Type': 'application/json'},
      body: JSON.stringify(editingTournament)
    });
    alert('✅ Оновлено');
    setEditingTournament(null);
    window.location.reload();
  };

  const handleDeleteTournament = async (id) => {
    if (window.confirm('Ви дійсно хочете видалити турнір?')) {
      await fetch(`http://localhost:5000/api/tournaments/${id}`, { method: 'DELETE' });
      alert('❌ Турнір видалено');
      window.location.reload();
    }
  };
```

Рисунок 4.16 – Код обробки редагування турніру (TournamentsPage.jsx),
із прив’язкою до ID турніру та перевіркою прав доступу

Під час реалізації було об’єднані всі головні функціональні блоки в єдину цілу вебсистему. Через API було реалізовану чітку та ефективну взаємодію між користувачами, сервером і БД. Адаптивний інтерфейс, автономна робота всіх блоків та узгоджена робота всіх частин вебсистеми дозволяють забезпечити повну функціональну роботу та зручність у використанні вебзастосунку.

4.4 Тестування функціональності та аналіз отриманих результатів

Для перевірки коректної роботи вебзастосунку, відповідності функціональних частин до вимог і стабільності використання інтерфейсу користувачем було проведено повноцінне тестування всіх компонентів вебсайту. Для тесту роботи було обране ручне тестування через залучення всіх типів користувачів. Для кожного випадку тестувалися всі доступні і можливі дії користувача на сторінках, починаючи від коректності відображення інформації і закінчуючи можливостями переходу між вкладками.

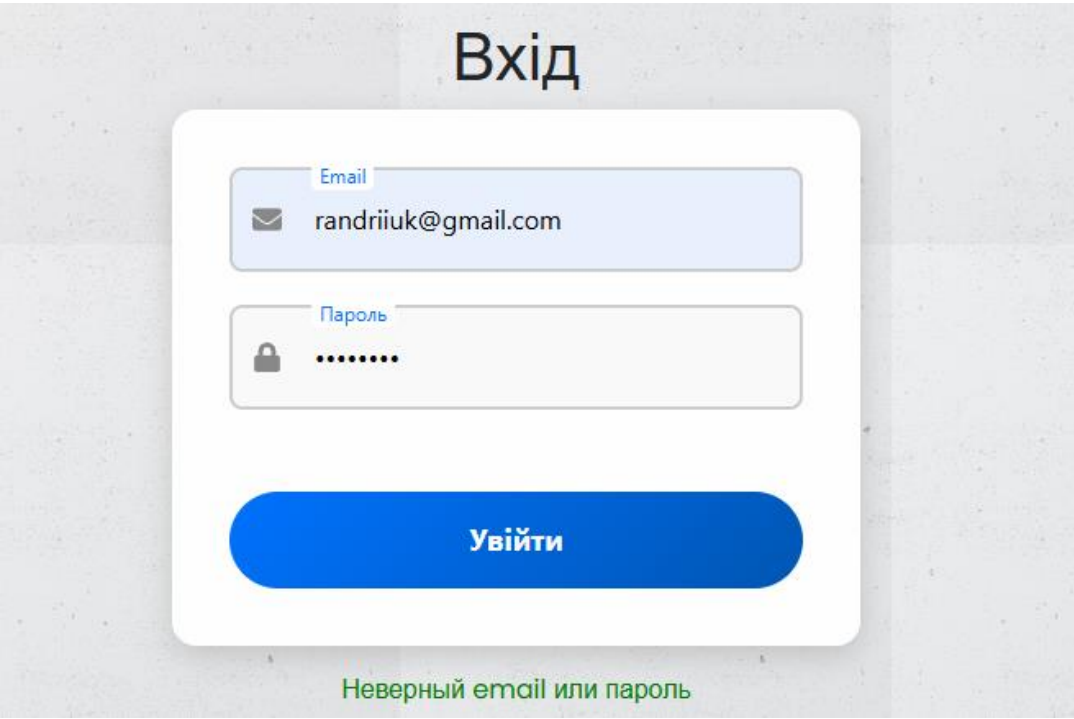


Рисунок 4.17 – Повідомлення про помилку входу при неправильному паролі

Спочатку перевірці піддалася форма реєстрації та авторизації користувача. Окрему увагу було приділено тестуванню обробці випадків з неправильним введенням пароля, некоректних записів у полях з електронною адресую та ім’ям, і звісно ж спроб входу до неіснуючого аккаунту. Вхідні дані були перевірені як на клієнтському так і на серверному рівні, з відображенням усіх відповідних повідомлень.

Додаткової уваги під час тестування потребувала перевірка доступності функцій відповідно до ролі користувача. Кожен клієнт після входу міг бачити лише ті сторінки і вкладки, які відповідали його ролі. Такий підхід дозволив запобігти спробам отримати доступ до недозволеного функціоналу.

Для перевірки функціональності турнірів були проведені такі тести, як реєстрація на турнір користувачів з ролями учень та аматор, перегляд учасників, автоматичне оновлення інформації після підтвердження дій. Після перевірки було доказано, що дитячі турніри доступні тільки учням, а на аматорські можуть подавати заявки тільки аматори, і що списки заявлених гравців автоматично оновлюється без перезавантаження сторінки.

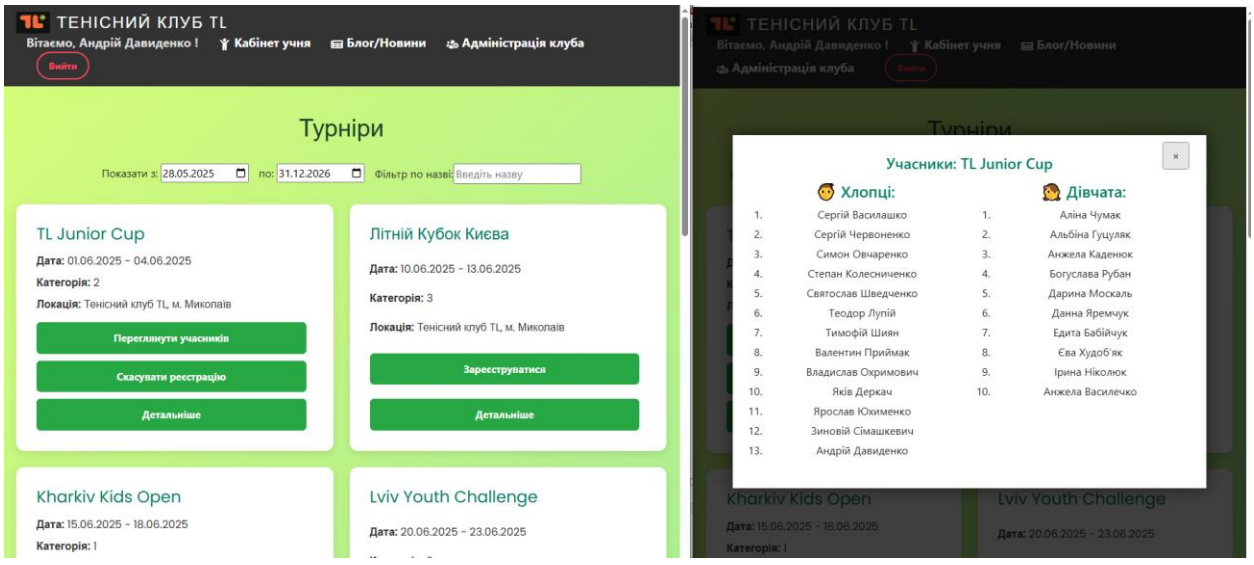


Рисунок 4.18 – Реєстрація учня на дитячий турнір і
динамічне оновлення списку учасників

У модулі бронювання кортів відбувалася перевірка функціональності усього циклу взаємодії – вибір дати, корту та часових слотів, після створення запису та його подальше редагування або видалення. Також протестовано було варіант з перетином часу, тобто при виборі зайнятого вже часу у відповідну дату система повертала повідомлення про неможливість виконати бронювання.

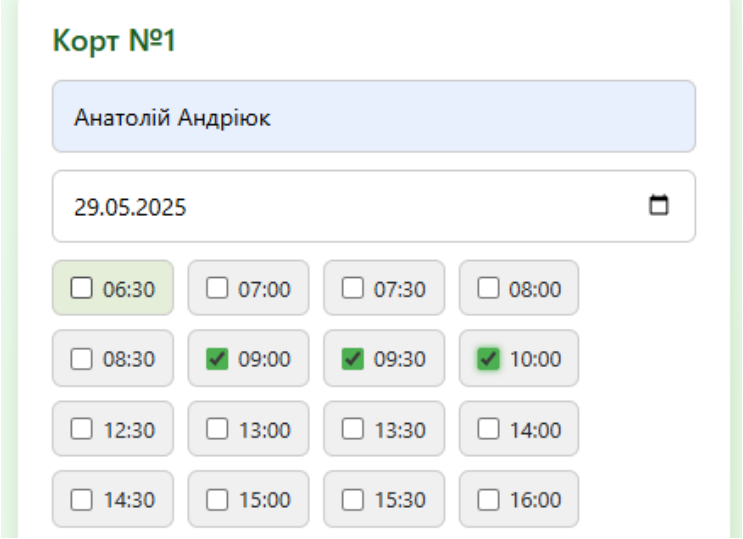


Рисунок 4.19 – Візуалізація вибраного періоду бронювання на інтерфейсі

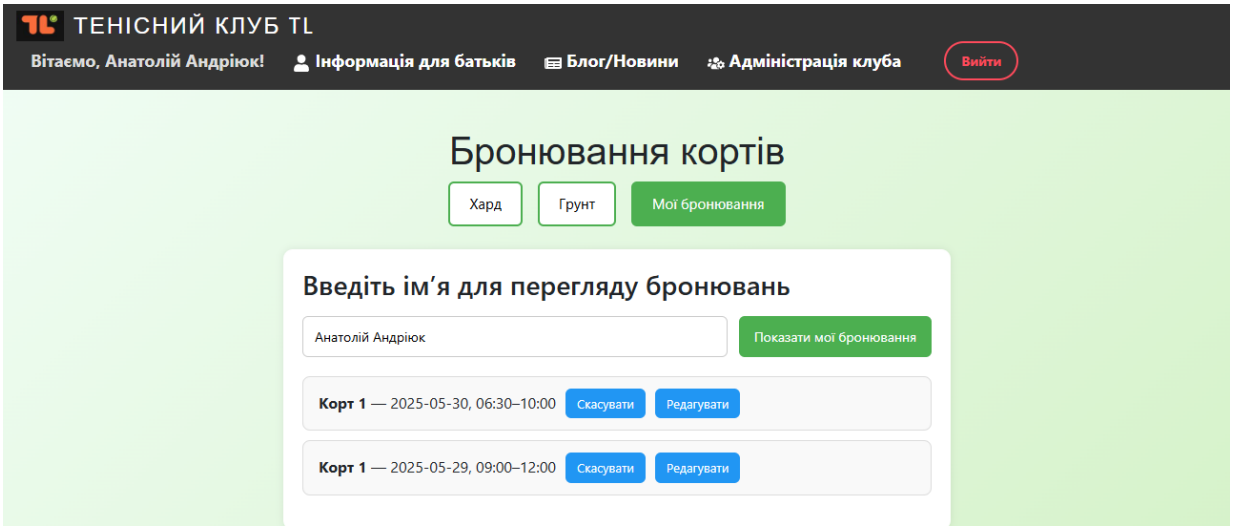


Рисунок 4.20 – Успішне створення бронювання з відображенням у вкладці «Мої бронювання»

Адмін панель мала окреме тестування. Була проведена перевірка на можливість редагування інформації про учнів, додавання новин, турнірів, бронювання тощо. Також було протестовано доступ до цих функцій, щоб жоден інший користувач не міг використовувати їх навіть через пряме посилання.

Інтерфейс вебсайту продемонстрував високу ефективність та стабільність під час проходження тестування. Швидке реагування на дії користувача та коректна обробка помилок стали гарними ознакам вебзастосунку. SPA-архітектура забезпечила зміну більшості даних без додаткових перевантажень вебзастосунку, що дуже гарно впливає на користувацький досвід. Також було проведено перевірку адаптивності дизайну. Результат показав що сторінки коректно відображались на пристроях з різною роздільною здатністю.

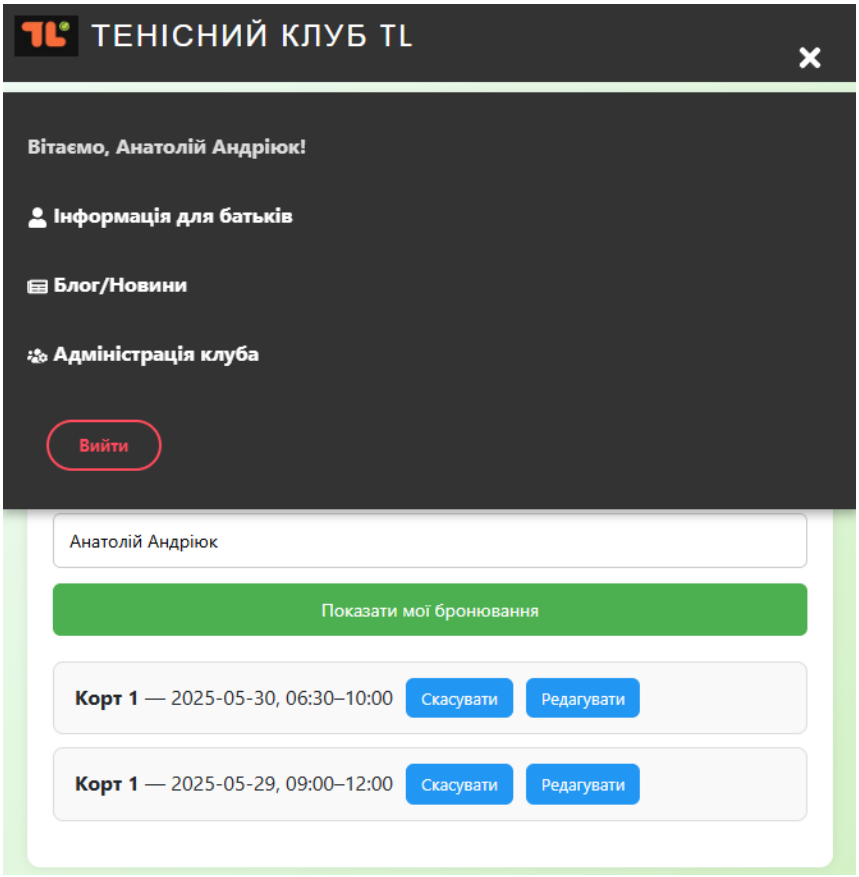
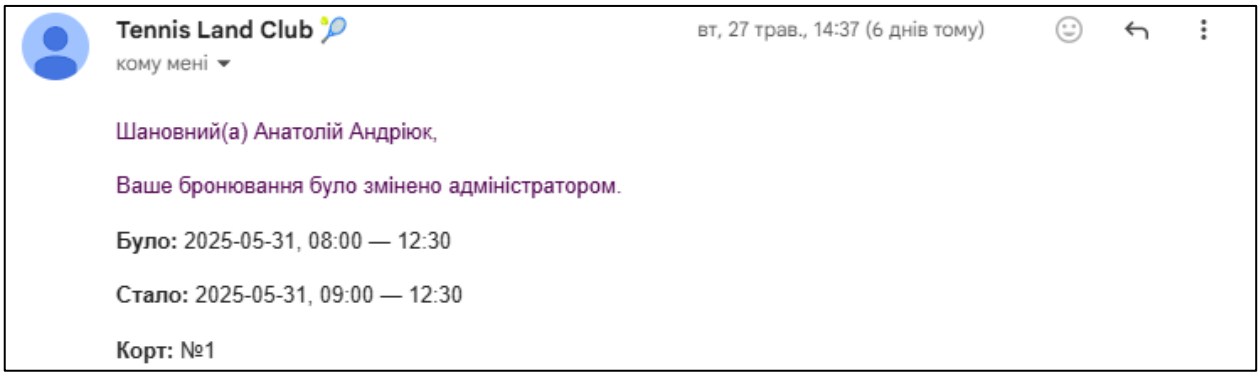


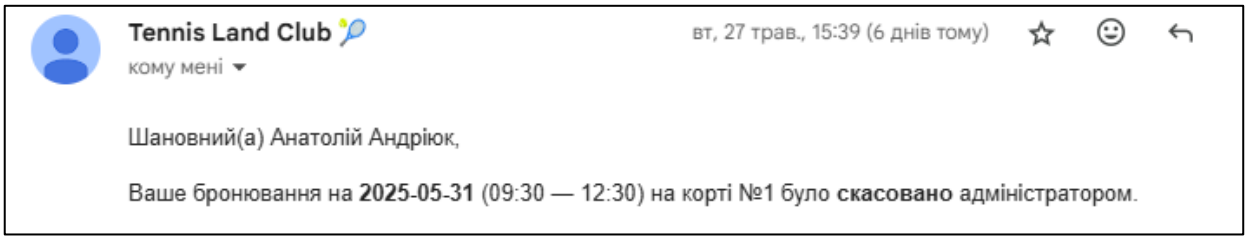
Рисунок 4.21 – Відображення сайту на мобільному пристрої з адаптованим меню

Додатково було протестовано роботу системи автоматичних email-сповіщень. Ця функція була реалізована для швидкого інформування користувачів про зміни, що стосуються з їхніми бронюваннями, заявками на турніри тощо. Система повинна надсилати електронного листа користувачеві за таких ситуацій, як відміна змагань, на які користувач подав заявку, відхилення заявки користувача на турнір, зміна або відміна бронювання користувача, видалення корту зі списку доступних, на яке було вже зроблено бронювання користувачем.

В листі міститься чіткий текст, в якому міститься інформація відповідна сценарію, який відбувся. Повідомлення відправляються за допомогою сервісу nodemailer з використанням SMTP-серверів, як Gmail, що надає можливість стабільної доставки.



а)



б)

Рисунок 4.22 – Приклади повідомлень
про зміну і відміну бронювання (а, б)

Цим тестуванням перевірялась відповідність адреси отримувача електронної пошти даним, які знаходяться у базі даних користувачів. Також було оцінено швидкість доставки. Всі листи було доставлені протягом 5 секунд після виконаної дії. Жодного разу повідомлення не було додано до папки «Спам», що говорить про гарну надійність SMTP-конфігурації.

Наступним кроком тестування стала серія технічних перевірок сайту за допомогою сучасних вебінструментів, які оцінюють продуктивність, адаптивність та якості коду. Основним інструментом для даного тестування був Google Lighthouse – ефективний і потужний засіб аудиту, вбудований у Chrome [28]. За допомогою нього стає можливим оцінити продуктивність (англ. Performance), доступність (англ. Accessibility), дотримання рекомендацій найкращих практик (англ. Best Practices) та пошукову оптимізацію (англ. SEO).

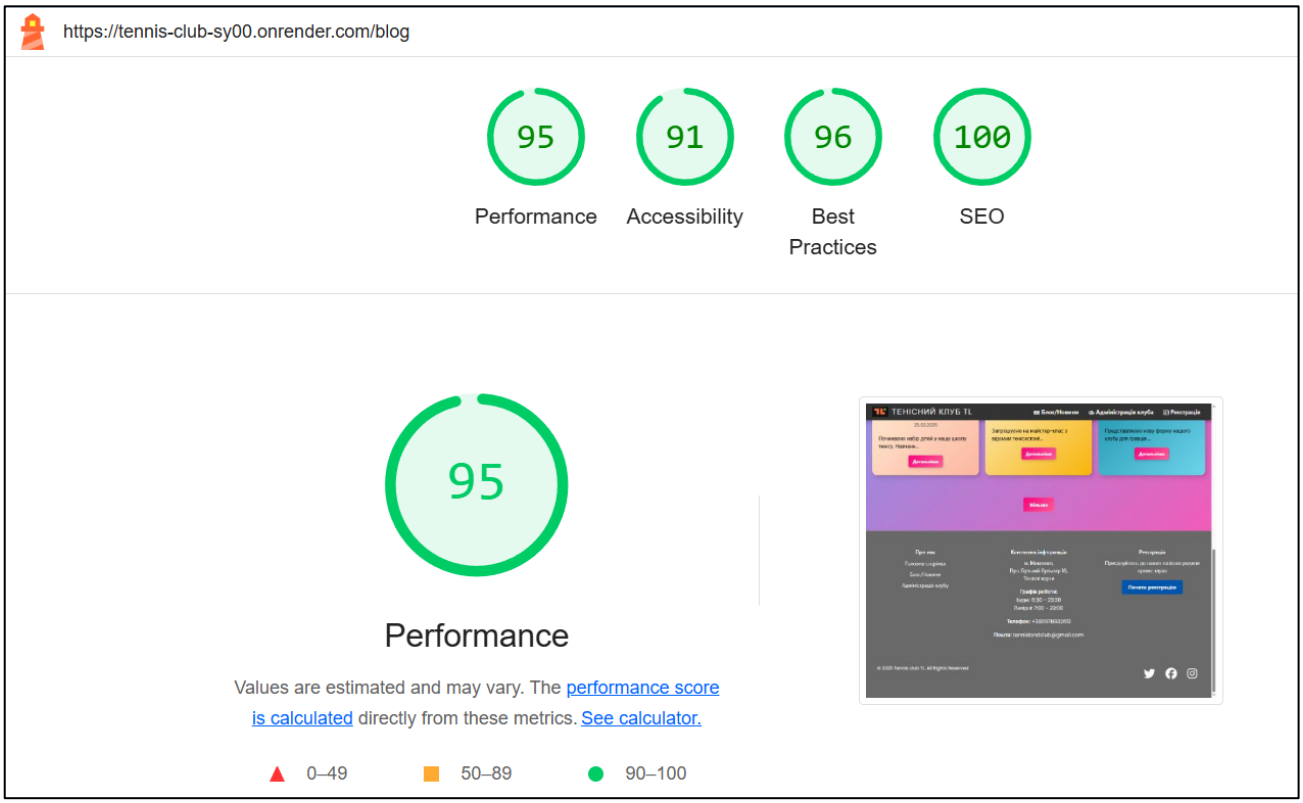


Рисунок 4.23 – Результат тестування вебсайту
за допомогою Google Lighthouse

Результати перевірки говорять про високу швидкість завантаження, належну технічну реалізацію клієнтської частини, доступність сайту для широкого кола користувачів та відповідність вимогам пошукової оптимізації.

Додатково зроблено аналіз сайту через сервіс GTmetrix, який проводить глибинну технічну перевірку продуктивності на основі реального запуску сторінки в браузері [29]. За результатами аудиту сторінка отримала найвищу оцінку з можливих, а саме GTmetrix Grade: A, при цьому показник Performance тут вже склав 100 %, а Structure був 98 %. Час повного завантаження сторінки становив лише 533 мс. Час до першого рендерингу склав 97 мс, що говорить про високу швидкість відповіді сервера.

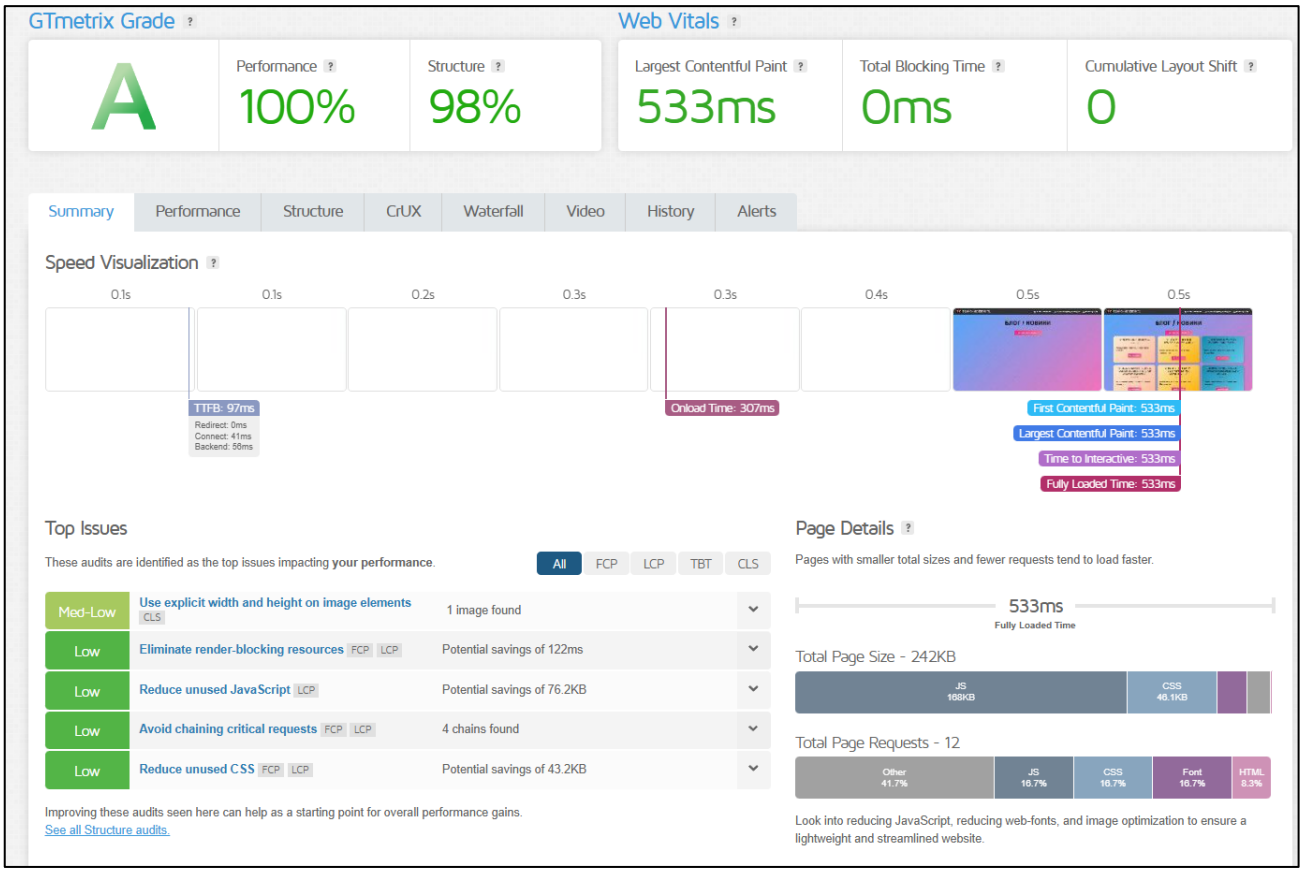


Рисунок 4.24 – Результат тестування вебсайту
за допомогою GTmetrix

Результати всіх тестувань лише підтвердили відповідність функціоналу вебсайту до заявлених вимог. Абсолютно всі модулі працюють узгоджено, розподіл користувачів за відповідними ролями працює коректно і обробка даних дає можливість забезпечити захист від помилок. Автоматизовані перевірки за допомогою сервісів Google Lighthouse та GTmetrix лише підтвердили високу швидкодію, оптимізацію та адаптивність сайту. Отже, розроблений вебзастосунок тенісного клубу успішно пройшов функціональне тестування.

Висновки до розділу 4

У четвертому розділі було проведено повні створення та реалізація функціональних частин вебсайту для тенісного клубу. Була реалізована клієнтська частина за допомогою фреймворка React. Такий підхід забезпечив динамічний, ефективний, адаптивний та зрозумілий інтерфейс. Серверна частина в свою чергу була створена через стек Node.js та Express.js із використанням MongoDB як БД, що створило можливість виконати ефективну структуру вебсистеми.

Основні блоки було інтегровано з урахуванням типів ролей користувачів. Усі частини працюють через API-запити, що сприяє стабільній передачі даних між клієнтом і сервером. Також була розроблена перевірка прав доступу, що гарантує захист особистих даних і коректну поведінку системи.

Повне функціональне тестування продемонструвало правильну роботу основних сценаріїв використання вебсайту та відповідність очікуваним вимогам. Вебзастосунок має стабільну роботу при різних взаємодіях та підтримку адаптивного відображення на різних пристроях.

У результаті, можна сміливо сказати, що реалізоване технічне рішення повноцінно відповідає поставленим задачам і може в майбутньому використане як основний вебзастосунок для сучасного тенісного клубу.

ВИСНОВКИ

Під час виконання кваліфікаційної бакалаврської роботи було реалізовано вебзастосунок для сучасного тенісного клубу, що дає змогу автоматизувати процеси взаємодії між учасниками клубу. Досягнуто поставленої мети – створено зручний, ефективний, швидкий сервіс, що відповідає всім вимогам цифрового обслуговування спортивної установи.

Завдання КБР було виконано в повному обсязі, зокрема:

- проведено огляд предметної сфери та проаналізовано аналогічні вебсайти;
- сформулювано вимоги до вебсайту тенісного клубу;
- розроблено структуру сайту з урахуванням різних ролей для користувачів;
- створено дизайн, адаптований до різних типів пристроїв;
- реалізовано вебсайт із повним функціоналом (напр., логіку реєстрації, бронювання, турнірів тощо);
- протестовано вебзастосунок на різних категоріях користувачів.

Вебсайт побудований на основі сучасного технологічного стеку, а саме React, Node.js, Express та MongoDB. Усі ці складові забезпечують надійність, гнучкіст і швидкодію створеного вебзастосунку. Усі компоненти логічно пов’язані між собою, що гарантує узгоджену й безперебійну роботу всіх модулів.

Особливу увагу було приділено адаптації інтерфейсу під різні ролі користувачів. Діти мають доступ до особистого розкладу, турнірів та інформації про школу. Батьки можуть переглядати розклад своїх дітей та бронювати корти. Тренери отримують доступ до власного розкладу, списку учнів та заявок на турніри. Аматори працюють в унікальному інтерфейсі з власними розділами та доступами. Адміністратори мають найширші права з усіх користувачів, вони можуть редагувати інформацію про користувачів, розклад, бронювання, новини, турніри тощо.

Функціонал вебсистеми було розширено завдяки таким важливим можливостям, як автоматичне формування розкладу, система email-повідомлень про

зміну чи скасування подій, фільтрація учнів за параметрами, динамічне оновлення списків без перевантаження сторінок. Окремої уваги заслуговує модуль бронювання кортів, який дозволяє не тільки обирати дату й час, а й редагувати чи скасовувати бронювання. Усі ці дії супроводжуються повідомленнями на електронну пошту користувача, що значно підвищує зручність та прозорість роботи вебсайту.

На етапі тестування було використано як ручні методи перевірки всіх можливих сценаріїв взаємодії, так і автоматизовані сервіси, такі як Google Lighthouse і GTmetrix. Усі показники продуктивності, доступності та стабільності підтвердили високу якість розробленого вебзастосунку. Результати продемонстрували що вебсайт є адаптивним, безпечним та масштабованим.

Важливо зазначити, що реалізований проєкт має потенціал для подальшого розвитку. Його можна доповнити інтеграцією платіжної системи, розширеним аналітичним модулем для тренерів, мультикористувацькою статистикою та мобільним застосунком.

Отже, розроблений вебзастосунок повністю виконує поставлені завдання, відповідає технічним і функціональним вимогам, демонструє високу продуктивність, зручність для різних категорій користувачів і готовність до майбутнього практичного впровадження в роботу тенісного клубу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Karczewski D. What Are The Best Frontend Frameworks To Use In 2023? *Ideamotive*. URL: https://www.ideamotive.co/blog/best-frontend-frameworks (Last accessed: 29.04.2025).

2. Безверхий О., Куценко О. Ефективність застосування бібліотеки React. *Інформаційні технології та суспільство*. 2022. Т. 2, № 4. С. 13–19. DOI: 10.32689/maup.it.2022.2.2.

3. Світличний В. А. Захист персональних даних в умовах воєнного стану в Україні. *Право і безпека*. 2023. Т. 90, № 3. С. 226–236. DOI: 10.32631/pb.2023.3.19.

4. Jain R., Shrivastava V., Pandey A., Sharma A. Modern Web Development using CSS & HTML. *International Journal of Emerging Science and Engineering*. 2024. Vol. 12, No. 6. P. 13–16. DOI: 10.35940/ijese.G2574.12060524.

5. Fedorchuk A., Usata O., Nakonechna O. Web Design and Web Programming in the Modern Internet World. *Municipal Economy of Cities*. 2023. Vol. 6, No. 180. P. 12–20. DOI: 10.33042/2522-1809-2023-6-180-12-20.

6. Lauer P., Zintel M. React – Einführung in ein Framework. University of Applied Sciences Kaiserslautern. 2020. URL: https://www.researchgate.net/publication/357684856_React_-_Einfuhrung_in_ein_Framework (Last accessed: 29.04.2025).

7. Attardi J. Modern CSS: Master the Key Concepts of CSS for Modern Web Development. Apress: Berkeley, CA, USA. Jan 2020. pp. 289. DOI: 10.1007/978-1-4842-6294-8. ISBN: 978-1-4842-6293-1.

8. Тищенко О. Що таке показник відмов і на що він впливає? *Wedex*. URL: https://wedex.com.ua/blog/shho-take-pokaznyk-vidmov-ta-na-shho-vin-vplyvaye/ (дата звернення: 30.04.2025).

9. Sy C. The ultimate guide to website performance metrics. URL: https://www.web.com/blog/website-performance-metrics/ (Last accessed: 29.04.2025).

10. Contentsquare. Average Session Duration Benchmarks by Industry. *Contentsquare Blog*. URL: https://contentsquare.com/blog/average-session-duration-by-industry/ (Last accessed: 30.04.2025).

11. Андріюк Р., Обухова К. розробка інформаційної системидля тенісного клубу на базі фреймворка React. *«Могилянські читання – 2024: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти»*: тези доп. XXVII Всеукр. наук.-практ. конф. (підсекція «Інтелектуальні інформаційні системи», м. Миколаїв, 6–10 листопада 2024 р.).

12. Alqahtani A., Khan A., Alghamdi N., Alkhaldi H. Developing and Implementing a Website for Sports Clubs. Inter*national Journal of Scientific Research in Science and Technology*. 2020. Vol. 7, No. 3. P. 76–83. DOI: 10.32628/IJSRST207157.

13. Belchior M., Schwabe D., Silva Parreiras F. Role-Based Access Control for Model-Driven Web Applications. *In book: Web Engineering*. Berlin, Heidelberg : Springer. 2012. P. 106–120. DOI: 10.1007/978-3-642-31753-8_8.

14. Sahasrabuddhe S. The Database System for the Sports Club Management: Data Modelling, Management and Governance. MSc *Project Management with Data Analytics*. 2024. P. 1–16. DOI: 10.13140/RG.2.2.32106.12489.

15. Salafi M.I.E., Tiran F., Nunes M., Cardoso M.J. App Development in a Team Sports: A Systematic Literature Review. *In book: Proceedings of the 6th Yogyakarta International Seminar on Health, Physical Education, and Sports Science*. 2024. P. 126–149. DOI: 10.2991/978-94-6463-356-6_16. ISBN: 978-94-6463-356-6.

16. Zhang Y. Research on the Intelligent Management Model of Sports Club Based on Database Technology. Applied Mathematics and Nonlinear Sciences. 2024. Vol. 9, No. 1. P. 259–267. DOI: 10.2478/amns-2024-3195.

17. du Mortier G. A Detailed Guide to Database Schema Design. *Vertabelo*. URL: https://vertabelo.com/blog/database-design-guide/ (дата звернення: 16.05.2025).

18. Create a New Database Diagram. *Microsoft Learn.* URL: https://learn.microsoft.com/en-us/ssms/visual-db-tools/create-a-new-database-diagram-visual-database-tools ((Last accessed: 16.05.2025).

19. Laranjeiro N., Pinto A. M. ONDA: ONline Database Architect. Jan. 2024. (Preprint). DOI: 10.48550/arXiv.2401.16552.

20. Software Ideas Modeler. UML Diagrams. URL: https://www.softwareideas.net/c/1058/UML (Last accessed: 16.05.2025).

21. MARINA Tennis Club. URL: https://marinatennisclub.com.ua/ (Last accessed: 29.05.2025).

22. Viccourt Ukrainian Tennis Center. URL: https://viccourt.com.ua/ (Last accessed: 29.05.2025).

23. GRAYS Tennis Club. URL: https://grays.com/ (Last accessed: 29.05.2025).

24. Learn React. URL: https://reactjs.org/docs/getting-started.html (Last accessed: 29.05.2025).

25. What is MongoDB? URL: https://www.mongodb.com/docs/manual/ (Last accessed: 29.05.2025).

26. Node.js v24.1.0 documentation. URL: https://nodejs.org/en/docs (Last accessed: 29.05.2025).

27. Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com/ (Last accessed: 29.05.2025).

28. Google Lighthouse. URL: https://developer.chrome.com/docs/lighthouse/ (Last accessed: 01.06.2025).

29. Test your website speed. *GTmetrix*. URL: https://gtmetrix.com (Last accessed: 02.06.2025).

ДОДАТОК А

Лістинг коду

Файл AdminPanel.jsx

```
import React, { useEffect, useState, useRef } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import './AdminPanel.css';

const AdminPanel = () => {
  const [users, setUsers] = useState([]);
  const [showUserList, setShowUserList] = useState(false);
  const [showLinks, setShowLinks] = useState(false);
  const userListRef = useRef(null);
  const linksListRef = useRef(null);
  const navigate = useNavigate();
  const currentUser = JSON.parse(localStorage.getItem('currentUser'));

  useEffect(() => {
    if (!currentUser || currentUser.role !== 'admin') {
      return navigate('/');
    }

    fetch('http://localhost:5000/api/auth/users')
      .then(res => res.json())
      .then(data => {
        const filtered = data.filter(u => u.role !== 'admin');
        setUsers(filtered);
      })
      .catch(err => console.error("Помилка завантаження користувачів:", err));
  }, [currentUser, navigate]);

  const handleNavigate = (user) => {
    navigate(`/admin/user/${user._id}`);
  };

  return (
    <div className="admin-panel">
      <h1>Панель адміністратора</h1>

      <div className="admin-column">
        <button onClick={() => setShowUserList(!showUserList)} className="toggle-users-
button">
          {showUserList ? 'Сховати список користувачів' : 'Зареєстровані користувачі'}
        </button>

        <div
```

```
            className={`user-list-wrapper ${showUserList ? 'expanded' : 'collapsed'}`}
            ref={userListRef}
            style={{ maxHeight: showUserList ? `${userListRef.current?.scrollHeight}px` :
'0px' }}
          >
            <ul className="user-list">
              {users.map(user => (
                <li key={user._id} onClick={() => handleNavigate(user)} className="user-
item">
                  {user.name} — <em>{user.role}</em>
                </li>
              ))}
            </ul>
          </div>
        </div>

        <div className="admin-column">
          <button  onClick={()  =>  setShowLinks(!showLinks)}  className="toggle-links-
button">
            {showLinks ? 'Сховати навігацію' : 'Навігація сайтом'}
          </button>

          <div
            className={`links-list-wrapper ${showLinks ? 'expanded' : 'collapsed'}`}
            ref={linksListRef}
            style={{  maxHeight:  showLinks  ?  `${linksListRef.current?.scrollHeight}px`  :
'0px' }}
          >
            <ul className="links-list">
              <li className="links-item">
                <Link to="/tournaments">Сторінка турнірів</Link>
              </li>
              <li className="links-item">
                <Link to="/club-trainers">Тренери та спаринги клубу</Link>
              </li>
              <li className="links-item">
                <Link to="/court-booking">Бронювання кортів</Link>
              </li>
              <li className="links-item">
                <Link to="/administration">Адміністрація</Link>
              </li>
              <li className="links-item">
                <Link to="/blog">Блог/новини</Link>
              </li>
            </ul>
          </div>
```

```
      </div>
    </div>
  );
};

export default AdminPanel;

Файл Registration.jsx
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import './Registration.css';
import { FaUser, FaEnvelope, FaLock, FaUserTag, FaKey } from 'react-icons/fa';

const Registration = () => {
  const navigate = useNavigate();
  const [isRegistered, setIsRegistered] = useState(false);
  const [isVerifying, setIsVerifying] = useState(false);
  const [message, setMessage] = useState('');

  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    role: ''
  });

  const [loginData, setLoginData] = useState({
    email: '',
    password: ''
  });

  const [verificationData, setVerificationData] = useState({
    email: '',
    code: ''
  });

  useEffect(() => {
    const token = localStorage.getItem('token');
    if (token) navigate('/');
  }, [navigate]);

  const handleChange = (e) => {
    if (isVerifying) {
      setVerificationData({ ...verificationData, [e.target.name]: e.target.value });
    } else if (isRegistered) {
      setLoginData({ ...loginData, [e.target.name]: e.target.value });
```

```
    } else {
      setFormData({ ...formData, [e.target.name]: e.target.value });
    }
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (isVerifying) {
      try {
        const res = await fetch('http://localhost:5000/api/auth/verify-email', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(verificationData),
        });
        const result = await res.json();
        if (!res.ok) throw new Error(result.message);

        setMessage('Електронна пошта підтверджена! Тепер ви можете увійти.');
        setIsVerifying(false);
        setIsRegistered(true);
      } catch (error) {
        setMessage(error.message);
      }
      return;
    }

    const url = isRegistered ? 'http://localhost:5000/api/auth/login' :
'http://localhost:5000/api/auth/register';
    const data = isRegistered ? loginData : formData;

    try {
      const res = await fetch(url, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data),
      });
      const result = await res.json();
      if (!res.ok) throw new Error(result.message);

      if (isRegistered) {
  localStorage.setItem('token', result.token);
  localStorage.setItem('currentUser', JSON.stringify(result.user));
  const role = result.user.role;
  if (role === 'coach') navigate('/coach-panel');
else if (role === 'child') navigate('/child-panel');
```

```
      else if (role === 'amatore') navigate('/amatore-panel');
      else if (role === 'parent') navigate('/parent-panel');
      else if (role === 'admin') navigate('/admin-panel');
      else navigate('/');

      }
       else {
              setMessage('Код підтвердження надіслано на вашу електронну пошту.');
              setIsVerifying(true);
              setVerificationData({ email: formData.email, code: '' });
            }
          } catch (error) {
            setMessage(error.message);
          }
        };

        return (
          <div className="registration-page">
            <h1>{isVerifying ? 'Підтвердження Email' : isRegistered ? 'Вхід' :
'Реєстрація'}</h1>
            <form onSubmit={handleSubmit} className="form-container">

              {isVerifying ? (
                <>
                  <div className="input-container">
                    <FaKey className="input-icon" />
                    <input     type="text"     name="code"     value={verificationData.code}
onChange={handleChange} required placeholder=" " />
                    <label className="input-label">Код підтвердження</label>
                  </div>
                  <button type="submit" className="submit-button">Підтвердити</button>
                </>
              ) : isRegistered ? (
                <>
                  <div className="input-container">
                    <FaEnvelope className="input-icon" />
                    <input      type="email"      name="email"      value={loginData.email}
onChange={handleChange} required placeholder=" " />
                    <label className="input-label">Email</label>
                  </div>
                  <div className="input-container">
                    <FaLock className="input-icon" />
                    <input    type="password"    name="password"    value={loginData.password}
onChange={handleChange} required placeholder=" " />
                    <label className="input-label">Пароль</label>
                  </div>
```

```
                <button type="submit" className="submit-button">Увійти</button>
              </>
            ) : (
              <>
                <div className="input-container">
                  <FaUser className="input-icon" />
                  <input        type="text"        name="name"        value={formData.name}
onChange={handleChange} required placeholder=" " />
                  <label className="input-label">Прізвище та Ім'я</label>
                </div>
                <div className="input-container">
                  <FaUserTag className="input-icon" />
                  <select    name="role"    value={formData.role}    onChange={handleChange}
required>
                    <option value="">Оберіть роль...</option>
                    <option value="amatore">Аматор</option>
                    <option value="coach">Тренер</option>
                    <option value="child">Учень ДЮСШ</option>
                    <option value="parent">Батько/Мати</option>
                  </select>
                </div>
                <div className="input-container">
                  <FaEnvelope className="input-icon" />
                  <input       type="email"       name="email"       value={formData.email}
onChange={handleChange} required placeholder=" " />
                  <label className="input-label">Email</label>
                </div>
                <div className="input-container">
                  <FaLock className="input-icon" />
                  <input    type="password"    name="password"    value={formData.password}
onChange={handleChange} required placeholder=" " />
                  <label className="input-label">Пароль</label>
                </div>
                <button type="submit" className="submit-button">Зареєструватися</button>
              </>
            )}
          </form>

          {message && <p className="message">{message}</p>}

          {!isVerifying && (
            <div className="toggle-container">
              <p>{isRegistered ? 'Ще не маєте акаунту?' : 'Вже маєте акаунт?'}</p>
              <button  onClick={()  =>  setIsRegistered(!isRegistered)}  className="toggle-
button">
                {isRegistered ? 'Зареєструватися' : 'Увійти'}
```

```
            </button>
          </div>
        )}
      </div>
    );
};

export default Registration;
```

ДОДАТОК Б

Апробація матеріалів КБР

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Підсекція:
ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ СИСТЕМИ

УДК 004.43

Андріюк Р. А.,
бакалаврант спеціальності 122 «Компютерні науки»,
Обухова К. О.,
ст. викладач кафедри комп'ютерної інженерії,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ТЕНІСНОГО КЛУБУ НА БАЗІ ФРЕЙМВОРКА REACT

У сучасному цифровому світі вебсайт став невід'ємним інструментом надання інформації та взаємодії з користувачами. Для тенісного клубу створення власної інформаційної системи дозволяє підвищити ефективність взаємодії з членами клубу, залучити нових користувачів і покращити обслуговування постійних відвідувачів. У цій роботі розглядається розробка вебсайту тенісного клубу на основі сучасних вебтехнологій, зокрема React, який є одним із найпопулярніших фреймворків для побудови інтерфейсів.

Сучасні технології веброзробки пропонують широкий спектр інструментів та рішень для створення швидких, безпечних та адаптивних вебсайтів. Використання передових технологій забезпечує високу продуктивність, масштабованість та надійність вебресурсів.

Розробка вебсайту включає як фронтенд-, так і бекенд-складові. Фронтенд розробка охоплює створення зручного користувацького інтерфейсу, що базується на HTML для структури сторінок, CSS для стилізації та створення адаптивного дизайну, а також JavaScript для динамічної взаємодії. Бекенд-складова реалізована на PHP для обробки даних, форм та взаємодії з базою даних. Разом ці технології забезпечують повноцінний функціонал для взаємодії користувачів із сайтом.

Одним із важливих етапів розробки вебсайту є створення чітко продуманої структури, що забезпечує логічну організацію контенту та полегшує навігацію для користувачів. Крім того, добре структурований сайт сприяє кращій індексації пошуковими системами, що позитивно впливає на його видимість в інтернеті.

Отже, було розроблено структуру вебсайту, яка забезпечить логічну та інтуїтивно зрозумілу навігацію для користувачів (рис. 1). Вона

враховує ключові розділи та підрозділи, забезпечуючи легкий доступ до інформації.

Головна сторінка – це перша сторінка, яку бачать користувачі. Вона надає огляд клубу та основні шляхи доступу до всіх розділів сайту.

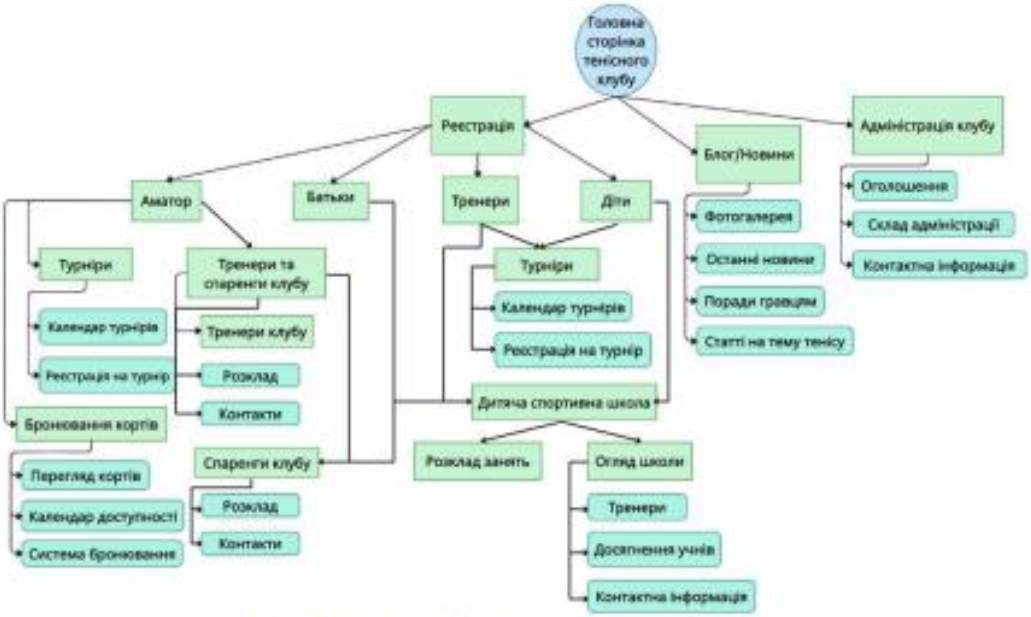


Рис.1. Розроблена структура вебсайту

Реєстрація – забезпечує зручний доступ для нових членів клубу та інших користувачів, зокрема:

– батьки можуть реєструвати своїх дітей для участі в дитячій спортивній школі або турнірах;

– тренери отримують доступ до управління розкладом і контактів з учнями;

– діти – реєструються на заняття у спортивній школі та на турніри;

– аматор – окремий розділ для людей, які займаються тенісом на аматорському рівні:

– можливість перегляду тренерів клубу, розклад занять, спорядження клубу та контакти.

Турніри – це важливий розділ, де користувачі можуть ознайомитися з календарем майбутніх турнірів, а також зареєструватися для участі у змаганнях. Є окремі підрозділи для дорослих та дітей.

Дитяча спортивна школа – розділ для дитячої аудиторії. Він містить інформацію про розклад занять та огляд школи. Тут також представлені тренери, досягнення учнів та контактна інформація.

160

Бронювання кортів – дозволяє користувачам переглядати доступність кортів і зручно бронювати їх через систему бронювання. Включає огляд корту, календар доступності та можливість бронювання.

Адміністрація клубу – блок для внутрішнього управління клубом. Включає склад адміністрації, контактну інформацію та важливі оголошення для членів клубу.

Блог/Новини – спеціальний розділ для публікацій новин, порад і статей на тему тенісу. Також тут можна знайти фотогалерею клубу.

Технологічні рішення. Для створення сайту використано сучасні та актуальні технології, такі як HTML, CSS, JavaScript і PHP, а також фреймворк React. Цей фреймворк забезпечує побудову динамічних, адаптивних і високопродуктивних інтерфейсів завдяки компонентній архітектурі, що дозволяє повторно використовувати код і спрощує масштабування проєкту. Це особливо важливо для сучасних вебсайтів, які потребують швидкого відгуку та інтерактивності.

React використовує віртуальну DOM, що значно прискорює оновлення інтерфейсу і знижує навантаження на систему. Це дозволяє створювати динамічні вебзастосунки, які забезпечують високу продуктивність і зручність використання. Зважаючи на зростаючі вимоги користувачів до швидкості та інтерактивності, React стає важливим інструментом для розробників.

При цьому також використовуються технології HTML, CSS та PHP, які залишаються базовими складовими веброзробки. HTML забезпечує структуровану основу кожної сторінки, і залишається базовим стандартом для побудови вебсайтів. Його використання є надзвичайно актуальним через постійну підтримку та розвиток стандартів вебіндустрії. CSS додає стиль і візуальну привабливість, що важливо для створення професійних і сучасних інтерфейсів. В умовах постійного розвитку вебтехнологій, особливої актуальності набуло використання CSS-фреймворків, таких як Bootstrap, який забезпечує адаптивність сайту на різних пристроях. У світі, де мобільний доступ домінує, адаптивний дизайн став необхідною умовою для успіху вебресурсу.

Для бекенду використано одну з актуальних мов програмування для вебсайтів – PHP. У сучасній веброзробці PHP вважається оптимальним рішенням для обробки серверних запитів, таких як:

– обробка форм реєстрації, що дозволяє безпечно і швидко збирати дані від користувачів;

– управління користувацькими даними, яке є критичним для забезпечення персоналізованого досвіду;

– взаємодія з базою даних для бронювання кортів та реєстрації на турніри, що є актуальною потребою для спортивних та комерційних сайтів.

React дозволяє створювати динамічні та високопродуктивні вебзастосунки завдяки компонентній архітектурі, що дозволяє повторно використовувати код і спрощує підтримку та масштабування проєкту. Серед переваг використання React можна виділити наступне:

– висока продуктивність: React використовує віртуальну DOM (Document Object Model), що мінімізує операції оновлення реального DOM і, як результат, покращує швидкість роботи веб-застосунків;

– компонентний підхід: Розробка з React дозволяє будувати інтерфейси з незалежних компонентів, які можна легко модифікувати або повторно використовувати. Це підвищує гнучкість у розробці та скорочує час на внесення змін або додавання нових функцій;

– широка підтримка спільноти: React має велику та активну спільноту розробників, що забезпечує доступ до безлічі бібліотек, інструментів і ресурсів, які постійно оновлюються та підтримуються;

– масштабованість: React підходить для розробки як невеликих сайтів, так і великих веб-застосунків завдяки своїй здатності ефективно управляти складними інтерфейсами з великою кількістю динамічних даних;

– адаптивність: Завдяки можливості легко адаптувати компоненти під різні платформи та пристрої, React є ідеальним вибором для створення адаптивних і мобільних вебзастосунків, що важливо для користувачів, які відвідують сайт з різних пристроїв.

Вибір таких інструментів для розробки є надзвичайно важливим етапом. Оскільки, це допомагає забезпечити необхідну гнучкість, масштабованість, безпеку та високу продуктивність, що робить їх незамінними для створення конкурентоспроможних вебсайтів.

Розробка сайту для тенісного клубу дозволяє автоматизувати та оптимізувати ключові процеси, такі як реєстрація на турніри, управління розкладом занять та бронювання кортів. Структурована платформа, що включає різні розділи для користувачів, тренерів та адміністрації, покращить взаємодію між клубом та його членами, забезпечуючи зручність користування та доступність усіх послуг.

Список використаних джерел

1. Lauer Ph., Zintel M. React - Einführung in ein Framework. 2020. 66 p. URL: https://www.researchgate.net/publication/357684856_React_-_Einfuhrung_in_ein_Framework (Last accessed: 08.10.2024).

2. Sabol D., Skalka J. The Architecture of Visual Design in Modern Web Applications. *In book: Microlearning*. Springer International Publishing, 2022. P. 171–194. DOI: 10.1007/978-3-031-13359-6_11.

3. Fedorchuk A., Usata O., Nakonechna, O. Web Design and Web Programming in The Modern Internet World. *Municipal economy of cities*. 2023. Vol. 6. No. 680. P. 12–20. DOI: 10.33042/2522-1809-2023-6-180-12-20.