Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система з адміністрування футбольних секцій. Front-end

(тема)

Виконав:
студент 4 курсу, групи ПЗПІпз-22-1

Персіанов С.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Валенда Н.А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Персіанову Сергію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система з адміністрування футбольних секцій. Front-end

Затверджена наказом по університету від 17.06. 2024р. № 93 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 22.07.2024

3. Вихідні дані до роботи Розробити клієнтську частину програмного застосунку з адміністрування футбольних секцій в Україні, котрий містить в собі режими адміністратора, менеджера секцій, тренера та опікуна неповнолітніх гравців, з використанням мови програмування TypeScript та бібліотеки ReactJS.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	17.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	04.06.2024	*виконано*
4	Розробка ПЗ	28.06.2024	*виконано*
5	Тестування ПЗ	30.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.07.2024	*виконано*
7	Підготовка презентації та доповіді	12.07.2024	*виконано*
8	Попередній захист	17.07.2024	*виконано*
9	Нормоконтроль, рецензування	17.07.2024	*виконано*
10	Здача роботи у електронний архів	18.07.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.07.2024	*виконано*

Дата видачі завдання 06 травня 2024р.

Студент (ка)_____ Персіанов С.С._____
(підпис)

Керівник роботи _____ _____доц. Валенда Н.А.___
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 84 стор., 32 рисунків, 1 таблиця, 3 додатків, 21 джерело.

БІБЛІОТЕКА REACTJS, МОВА ПРОГРАМУВАННЯ TYPESCRIPT, ФУТБОЛ В УКРАЇНІ, REST API.

Об’єкт розробки – клієнтська частина програмного застосунку з адміністрування футбольних секцій в Україні.

Мета розробки – створення клієнтської частини програмного застосунку, який підтримуватиме чотири ролі: адміністратора, менеджера секцій, тренера та опікун.

Метод рішення – бібліотека веб-програмування ReactJS, Jetbrains Webstorm, мова програмування TypeScript.

Результатом роботи стало створення клієнтської частини додатку, який дозволяє зберігати, керувати та редагувати дані про користувачів, футбольні секції, програми, заняття та команди. Додаток забезпечує можливість реєстрації гравців на різні програми створені тренерами, додавання їх до футбольних команд, а також пошуку та фільтрації інформації, згідно режиму, або ролі, користувача. Крім того, система підтримує розсилку сповіщень електронною поштою про зміни в розкладі занять і надає статистику щодо реєстрацій на програми.

The object of development is back-end for a software application dedicated to football management and football sections administration.

The purpose of the work is to create a software application that will provide 4 modes of football management: administrator, section manager, coach, and guardian.

Solution method – Django web development framework, JetBrainds Pycharm development environment, and Python programming language.

The result of the work was the creation of the client part of the application, which allows to store, manage and edit data about users, football sections, programs, classes and teams. The application provides the ability to register players for various programs created by coaches, add them to football teams, as well as search and filter information, all accordingly to the modes/roles authorization. In addition, the system supports the sending of notifications by e-mail about changes in the schedule of classes and provides statistics on registrations for programs.

Я, Персіанов Сергій Сергійович, студент гр. ПЗППпз-22-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система з адміністрування футбольних секцій. Back-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Управління футбольними тренуваннями, що включає в себе планування занять, взаємодію з гравцями та їхніми родичами, представляє собою складне завдання, яке вимагає великих зусиль з боку тренерського складу. Значущим аспектом для гравців та їхніх опікунів є можливість співпраці з професійними тренерами, які мають позитивну репутацію. Окрім цього, ключовим є просування футбольної культури в Україні та будівництво міцних відносин між учасниками футбольного середовища.

Завдання, які стоять перед футбольною спільнотою, відкривають шлях для впровадження автоматизованих рішень, зокрема розробки систем для ефективного управління футбольними тренуваннями в межах України. Основна ціль цього дослідження полягає у створенні ключових елементів для порталу, призначеного для тренерів.

Завдяки проведеній роботі була розроблена програмна система, яка дозволяє здійснювати реєстрацію учасників, включення їх до команд, а також управління даними про тренування, команди та програми. Програма надає функціонал для пошуку та фільтрації інформації, отримання статистичних даних щодо участі в програмах і повідомлення гравців про зміни у розкладі тренувань через електронні листи. Розроблений інструмент забезпечує інтуїтивно зрозумілий інтерфейс, що дозволяє особам з будь-яким рівнем підготовки легко адаптуватися до його використання.

У процесі створення програми було задіяно низку інструментів та технологій, включаючи розробницькі середовища JetBrains, мова програмування TypeScript.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної області

Платформа для адміністрування спортивних секцій дозволить тренерам в Україні виявляти та формувати футбольні секції, а також уможливить запис гравців або їхніх батьків/опікунів на футбольні програми. Управлінці секцій зможуть вести керування своїми футбольними секціями, авторизувати програми та перевіряти кваліфікації тренерів, що проводять заняття в секціях. Застосунок спрямований на підтримку та розвиток системи управління спортивними відносинами у сфері футболу в Україні [1].

Футбольні секції являють собою центральні елементи в архітектурі програмної системи з адміністрування футбольних секцій, з'єднуючи фізичну локацію футбольного поля з конкретною адресою, що є у власності організації, де тренери проводять тренування з гравцями. Для кожної секції призначений відповідальний менеджер.

Ці секції стають базою для тренерів для створення та ведення навчальних програм, які можуть включати різноманітні заняття і відкривати шлях для кількох програм. Тренери несуть відповідальність за управління командами та адаптацію розкладу занять за потребою.

Гравці беруть участь у тренуваннях під керівництвом тренера (або декількох тренерів), вступаючи до складу команд (або більш ніж однієї). Молодші гравці мають призначених опікунів, які служать контактною особою для екстрених випадків.

Ролі користувачів включає тренерів, які займаються забезпеченням освітнього процесу, організацією та проведенням занять за програмами, виховною діяльністю та оновленням інформації про досвід та заняття. Опікуни гравців відповідають за запис на футбольні програми та взаємодію з тренерами щодо організаційних аспектів.

Початок навчального процесу включає вибір тренерів гравцями та їхніми опікунами, реєстрацію на футбольні програми, включення в команди, планування тренерами тренувань за розкладом.

Інформаційні потреби охоплюють відомості про гравців, зареєстрованих на програми та в командах, деталі програм і команд, а також актуальні дані про призначення тренера на програму, забезпечені через електронні розсилки.

Таким чином, доменна модель включає в себе п'ять основних типів сутностей: користувачі, секції, програми, заняття, команди, та три типи взаємодій між ними: реєстрація гравців на програми, включення гравців до складу команд, призначення опікунів для гравців.

Наразі на ринку присутні додатки для пошуку футбольних тренерів, які виступають як прямі конкуренти розроблюваного застосунку. В рамках аналізу конкуренції було розглянуто наступні аспекти: методи монетизації, стратегії залучення користувачів, та ключові чинники успіху.

Основним конкурентом є платформа “Trener” [2], яка охоплює майже всі види спорту і дозволяє записувати безпосередньо до тренера без тематичного поділу. Монетизація здійснюється за рахунок збору плати з тренера за розміщення профілю. Основна перевага “Trener” полягає у вузькій спеціалізації на пошуку тренерів, завдяки чому в Україні відсутні прямі конкуренти в цьому сегменті.

До прямих конкурентів також відносяться дошки оголошень: сервіс замовлення послуг “Кабанчик” [3], платформа пошуку репетиторів “BUKI” [4], сервіс пошуку та доставки товарів “OLX” [5], та спеціалізований веб-сервіс пошуку навчальних закладів “Edusearch” [6].

Монетизація в цих сервісах здійснюється шляхом збору плати з користувачів за розміщення оголошень про свої послуги. Велика користувацька база та широкий спектр послуг є ключовими перевагами цих платформ. Клієнтам подобається можливість шукати послуги на вже знайомому сайті, що вважається популярним та безпечним, з відомим інтерфейсом.

Сервіс “Trener”, найбільш відповідний до тематики даного проекту, представлений веб-сервісом, де тренери можуть реєструватися та заповнювати свої профілі (рис. 1.1) [2].

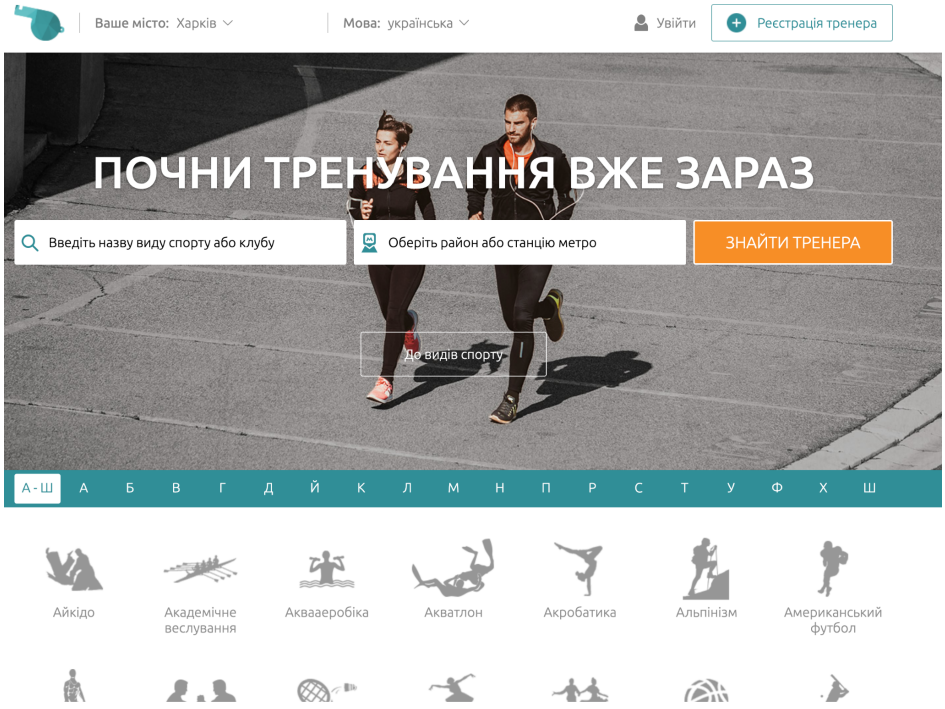


Рисунок 1.1 – Інтерфейс головної сторінки веб-сервісу «Trener» [2]

Анонімні користувачі, потенційно гравці або їхні опікун/батьки, можуть знаходити тренера за видами спорту, містом, районом міста або станцією метро, віком, статтю, вартістю послуг, розкладом (рис.1.2-1.3).

Сервіс “Trener” надає детальну інформацію про тренера, включаючи сертифікації, досвід роботи, вік тощо. Основними перевагами веб-сервісу “Trener” є його зручність у користуванні, високоякісний інтерфейс, локалізація, охоплення різноманітних видів спорту, підтримка обширних фільтрів для пошуку тренерів, та активна підтримка з боку розробників. Однак, серед недоліків варто відзначити відсутність функціоналу для допомоги тренерам у керуванні програмами, командами та заняттями, а також для гравців у плануванні занять.

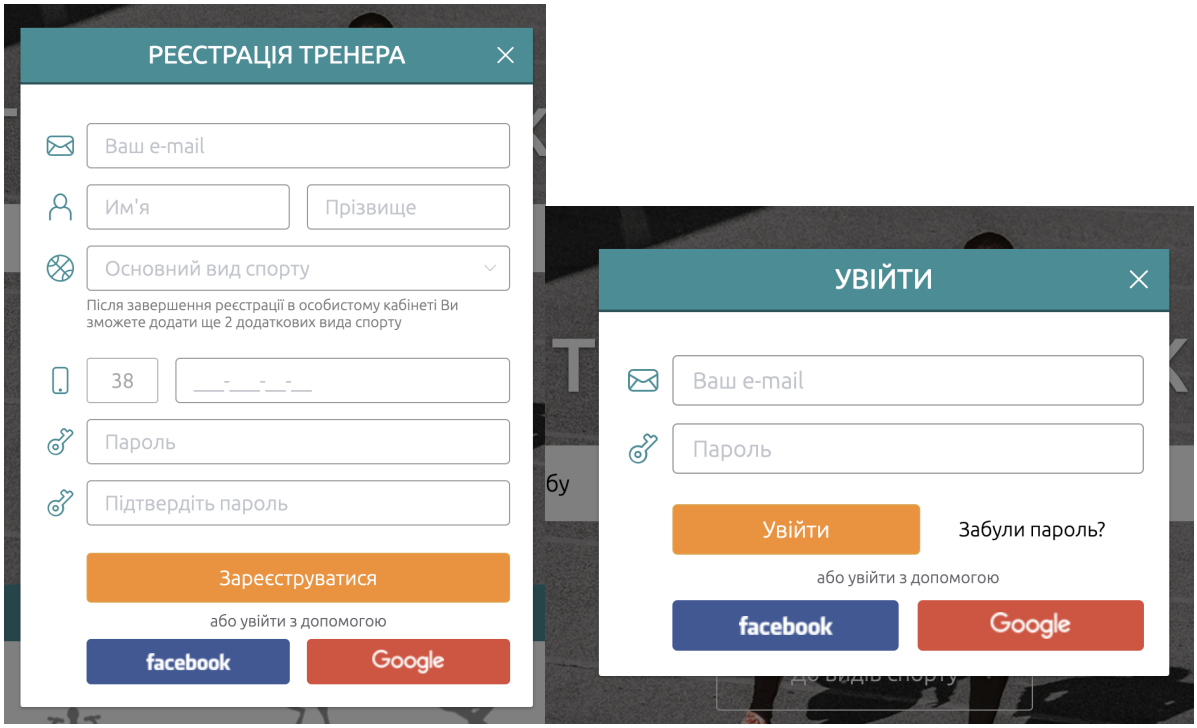


Рисунок 1.2 – Інтерфейс реєстрації та логіну веб-сервісу «Trener» (реєстрація та логін тренера) [2]

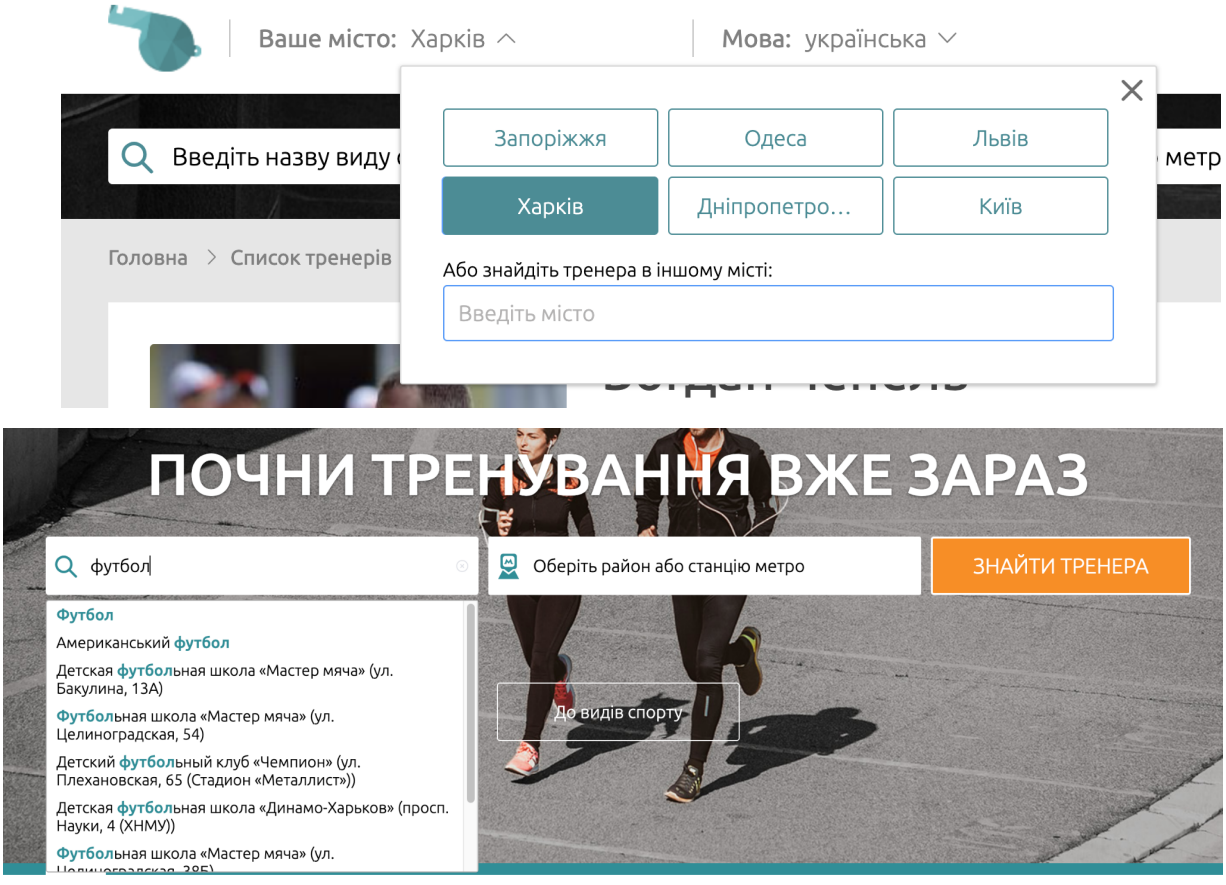


Рисунок 1.3 – Інтерфейси налаштувань пошуку тренерів у «Trener» [2]

1.2 Виявлення проблем

Зосередження на покращенні відносин в спорті (SRM - Sport Relationship Management) передбачає створення якісних взаємин між всіма учасниками процесу та підвищення задоволеності від участі в футбольних секціях [1]. Розроблювана програмна система має намір запровадити ряд конкурентних переваг та функціональності для досягнення цієї мети.

Серед ключових напрямків покращення варто виокремити наступне: зосередженість на компетентнісному підході дозволяє користувачам придбати структуровані навчальні програми, що чітко вказують на переваги послуг тренера. Контроль якості послуг тренерів передбачає перевірку наявності вищої спортивної освіти, сертифікації в Українській Асоціації Футболу (UAF) [7,8], з можливістю завантаження та перегляду сертифікатів тренерів. Зручний календар для планування занять та консолідація даних спрощують ведення розкладу.

Дане покращення буде реалізоване за рахунок сукупності конкурентних переваг та функціональності, що їх забезпечуватиме. По-перше, мова йде про сфокусованість на набутті навичок (компетентнісний підхід): користувач придбаває конкретний курс - структуровану відкриту навчальну програму. По-друге, варто згадати організаційне супроводження користувачів (тренерів та гравців): наявність менеджерів секцій, які можуть підтримати як гравців, так і тренерів. У менеджерів секцій є доступ до програм власних секцій - більше прав у менеджерів секцій, ніж у інших користувачів. По-третє, застосунок характеризує низький рівень складності UI / UX функціональності, що вирішує конкретну задачу - адміністрування своїх футбольних секцій користувачами (всіх категорій: гравцями, тренерами, менеджерами футбольних секцій). Дану перевагу буде реалізовано за рахунок наступних факторів: розробка ПЗ на основі мокапів; розробка неперевантаженого мінімалістичного інтерфейсу, його чітке розділення за функціональністю, зменшення візуального навантаження.

Таким чином, програмна система має на меті не просто полегшити пошук футбольних тренерів, але й значно спростити процеси адміністрування для тренерів, гравців та їхніх опікунів/батьків, а також для менеджерів футбольних секцій, сприяючи формуванню якісних спортивних відносин, адже місією продукту буде розвиток командних видів спорту в Україні та поширення його популярності серед всіх верств населення, змалку до велика, - завдяки побудові довгострокових відносин між гравцями будь-якого віку, тренерами, та представниками інфраструктури.

1.3 Вибір засобів розробки

Реалізація клієнтської частини додатку повинна базуватися на використанні бібліотеки React та пов'язаних з нею інструментів, що забезпечують високу продуктивність і зручність розробки. React, завдяки своїй компонентній архітектурі, дозволяє створювати масштабовані додатки, де кожен компонент відповідає за певну функціональність. Це сприяє більшій гнучкості та повторному використанню коду, що значно прискорює процес розробки. Екосистема React включає різноманітні бібліотеки та інструменти, які спрощують створення складних інтерфейсів.

Одним із основних завдань є створення керуванням стану застосунку за допомогою Redux у Single Page Applications, яка містить інформацію про всіх користувачів системи, включаючи тренерів, гравців, опікунів, менеджерів та адміністраторів, а також про програми, секції, заняття та команди. Це означає, що всі зміни в даних повинні автоматично оновлюватися в системі, запобігаючи будь-яким можливим розбіжностям. Наприклад, коли новий гравець реєструється, його дані мають бути негайно доступні тренерам і менеджерам для включення у відповідні програми та команди. Аналогічно, якщо тренер оновлює інформацію про тренування або заняття, ці зміни повинні відразу відображатися для всіх

користувачів, які мають до них доступ. Це забезпечить актуальність і точність інформації в будь-який момент часу.

Для захисту даних користувачів керування станом повинно використовувати сучасні методи шифрування і забезпечувати контроль доступу, щоб лише уповноважені особи могли переглядати або змінювати інформацію. Це допоможе запобігти несанкціонованому доступу та захистити конфіденційні дані від можливих загроз. Крім того, система повинна підтримувати функції пошуку, фільтрації та сортування даних, щоб користувачі могли швидко і легко знайти необхідну інформацію. Це включає можливість пошуку гравців за іменем, фільтрацію програм за датою або сортування занять за місцем проведення, що забезпечить зручність та ефективність використання системи.

Кожна футбольна секція повинна бути асоційована з конкретним фізичним місцем проведення тренувань, включаючи точну адресу та відповідального менеджера. Це гарантує, що всі тренування та заходи проводитимуться у визначених місцях, що мінімізує можливі непорозуміння щодо локації. Інформація про місцезнаходження секцій має бути доступною у системі для всіх користувачів, що полегшить тренерам і гравцям пошук необхідної секції та планування своїх дій відповідно до місця проведення занять. Менеджер, відповідальний за секцію, відіграватиме ключову роль у координації діяльності, забезпечуючи належний стан футбольного поля, наявність необхідного обладнання та ресурсів для проведення тренувань.

Тренери отримають можливість створювати та керувати навчальними програмами, планувати тренування та змінювати розклад занять відповідно до потреб команди або окремих гравців. Це включає створення нових програм, редагування існуючих та управління списками учасників. Гнучкість у плануванні дозволить максимально ефективно використовувати доступний час та ресурси, забезпечуючи високий рівень організації тренувального процесу. Гравці зможуть записуватися на тренувальні програми, створені тренерами, та приєднуватися до

команд. Молодші гравці матимуть призначених опікунів, які будуть контактною особою у разі екстрених ситуацій. Опікуни, як правило, є батьками або легальними опікунами, і отримуватимуть сповіщення про розклад тренувань, зміни в програмах та іншу важливу інформацію, що забезпечить їхню обізнаність і готовність реагувати на будь-які ситуації.

Інтуїтивно зрозумілий інтерфейс користувача з простим та мінімалістичним дизайном є надзвичайно важливим для успішного використання системи. Такий підхід дозволить користувачам з різним рівнем підготовки швидко адаптуватися до роботи з додатком. Важливо, щоб кожен елемент інтерфейсу був розташований логічно і був легко доступним, забезпечуючи безперешкодну навігацію. Дії, як-от реєстрація на програми, додавання гравців до команд та управління розкладом занять, повинні бути виконувані без зайвих зусиль і зрозумілі з першого погляду.

Пріоритетним аспектом є автоматизація процесів у системі. Зокрема, система повинна самостійно відправляти сповіщення електронною поштою гравцям та їх опікунам про будь-які зміни у розкладі занять. Також важливо, щоб система надавала статистичні дані щодо участі в програмах. Це дозволить значно зменшити навантаження на тренерів та менеджерів, даючи їм змогу зосередитися на організаційних та навчальних аспектах роботи. Автоматизовані сповіщення гарантують, що гравці та опікуни завжди будуть в курсі актуальної інформації про тренування, включаючи перенесення, скасування або зміну часу занять. Це сприяє підтриманню високого рівня організованості та допомагає уникнути непорозумінь, що можуть виникнути через недоотримання інформації.

Для забезпечення високого рівня надійності та безпеки даних необхідно впровадити сучасні методи аутентифікації та авторизації користувачів. Одним із найбільш ефективних способів є використання JWT (JSON Web Tokens) для авторизації, що гарантує безпечну передачу даних між клієнтською та серверною частинами системи. Під час входу в систему генеруються токени, які зберігаються у браузері користувача, що забезпечує швидкий і безперебійний доступ до ресурсів

системи без необхідності повторної аутентифікації. Використання JWT також дозволяє зберігати контекст користувача між різними сеансами, забезпечуючи зручність і безперервність роботи. Це включає збереження налаштувань користувача, прав доступу та іншої важливої інформації, що покращує загальний користувацький досвід.

Використання React також забезпечує високу продуктивність завдяки віртуальному DOM, який мінімізує кількість прямих маніпуляцій з DOM та покращує швидкість роботи додатку. Це особливо важливо для великих додатків з великою кількістю користувачів. Сучасні методи розробки з використанням React гарантують, що додаток буде швидким, надійним та легко масштабованим, що є ключовими факторами для успішного проекту.

Серед нефункціональних вимог до системи важливими є доступність, супроводжуваність та переносимість. Система повинна бути доступною для користувачів з обмеженими можливостями, дотримуючись принципів WCAG (Web Content Accessibility Guidelines). Це включає забезпечення високого контрасту кольорів, надання текстових альтернатив для зображень, а також підтримку навігації за допомогою клавіатури та інших допоміжних пристроїв. Ці заходи гарантують, що всі користувачі, незалежно від фізичних можливостей, зможуть ефективно використовувати додаток. Супроводжуваність додатку вимагає добре структурованого та задокументованого коду. Використання зрозумілих імен змінних, функцій та класів, а також коментарів та докладної документації сприятиме швидкому освоєнню додатку розробниками. Це значно спростить процес підтримки та подальшого розвитку системи, зменшить час на навчання нових членів команди та знизить ризик виникнення помилок.

Також важливою вимогою є переносимість додатку. Додаток повинен коректно працювати у різних сучасних браузерах, таких як Chrome, Mozilla, Safari та Edge, незалежно від операційної системи. Використання стандартних веб-технологій, як-от HTML5, CSS3 та JavaScript, забезпечить сумісність з різними

платформами. Регулярне тестування додатку на різних пристроях та у різних браузерах дозволить виявляти та виправляти можливі проблеми, забезпечуючи стабільну та надійну роботу системи для всіх користувачів.

1.4 Постановка задачі

Розробка програмної системи для управління футбольними секціями передбачає вирішення кількох ключових завдань, необхідних для забезпечення ефективної роботи системи та задоволення потреб усіх користувачів. Основна ціль полягає у створенні зручного, продуктивного та надійного додатку, який дозволяє тренерам, менеджерам секцій, гравцям та їх опікунам легко взаємодіяти та організовувати свою діяльність. Проєктування зрозумілого, гнучкого та ефективного користувацького інтерфейсу за допомогою інструменту Figma є одним із пріоритетів. Важливо, щоб додаток працював стабільно, без збоїв і затримок, забезпечуючи швидку обробку великих обсягів даних.

Тренери повинні мати можливість створювати та керувати навчальними програмами, планувати тренування та відстежувати прогрес гравців. Менеджери секцій повинні мати доступ до інструментів для організації розкладу занять і контролю за участю гравців. Розробка клієнтської частини (реактивного інтерфейсу) програмної системи з адміністрування футбольних секцій за допомогою бібліотеки ReactJS та її екосистеми дозволить забезпечити високий рівень зручності для користувачів. Гравці та їх опікуни повинні мати зручний доступ до інформації про тренування, можливість реєструватися на програми та отримувати сповіщення про зміни.

Забезпечення реєстрації гравців на тренувальні програми, створені тренерами, є однією з ключових функцій системи. Крім цього, система має підтримувати додавання гравців до команд та управління даними про секції, програми, заняття та команди. Важливо запровадити локалізацію програмної системи англійською мовою за допомогою бібліотеки i18next, щоб задовольнити

потреби користувачів, які не говорять українською. Це сприятиме розширенню аудиторії користувачів та підвищенню рівня зручності для різних груп.

Однією з важливих функцій є можливість пошуку та фільтрації інформації, що дозволяє користувачам швидко знаходити необхідні дані серед великого обсягу записів. Наприклад, тренери можуть фільтрувати списки гравців за різними критеріями, такими як вік, рівень підготовки або наявність у команді, що значно спрощує процес управління. Автоматизована розсилка сповіщень електронною поштою про зміни в розкладі занять є ще однією важливою функцією. Проведення мануального функціонального тестування та тестування інтерфейсу для забезпечення якості програмного забезпечення дозволить виявити можливі помилки та покращити загальну стабільність системи. Це забезпечить своєчасне інформування гравців та їх опікунів про будь-які зміни у тренувальному процесі, включаючи перенесення занять, зміну часу або місця проведення. Така автоматизація дозволяє зменшити навантаження на тренерів і менеджерів, оскільки їм не потрібно вручну відправляти повідомлення кожному гравцю. Можливості системи з пошуку, фільтрації даних та автоматизації сповіщень значно підвищать рівень організованості та зручності для користувачів. Це дозволить ефективніше планувати участь у тренувальних програмах та забезпечить актуальність інформації, сприяючи більшій дисципліні та відповідальності серед учасників.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Необхідно спроектувати програмну систему збереження інформації стосовно адміністрування футбольних програм і секцій, що виконується тренерами та менеджерами секцій, та створити інформаційну систему, що буде забезпечувати функції, що представлені нижче.

Зокрема, програмна система повинна відображати наступні дані:

- безпосередньо про основні поняття ПЗ: користувачі (тренери, гравці, опікуни, менеджери, адміністратори), програми, секції, заняття, команди;
- безпосередньо про основні поняття ПЗ: користувачі (тренери, гравці, опікуни, менеджери, адміністратори), програми, секції, заняття, команди;
- про пов’язані поняття ПЗ: опікунів гравців, реєстрації гравців на програми, інформацію про гравців у складі команд.

Система повинна підтримувати арифметичну обробку даних у вигляді обчислювальних полів стосовно - згідно наведених мокапів [9]:

- мінімальної та максимальної дати народження гравців, що зареєструвалися на футбольну програму;
- суми вартості всіх програм та середньої вартості всіх програм;
- кількості гравців кожної команди з переліку.

Також, програмна система повинна підтримувати сортування, пошук та фільтрацію даних - згідно наведених мокапів:

- сортувати програми та команди за назвою, гравців у складі команди та зареєстрованих гравців - за ім’ям гравця;
- здійснювати пошук: програм - за назвою; студентів, що зареєстровані на програму, - за назвою;
- фільтрувати опубліковані та неопубліковані програми, а також реєстрації на програми - за датою народження студентів, що зареєстровані на програму.

Додатково, програмна система повинна:

- підтримувати додавання, редагування, та видалення (з підтримкою режиму підтвердження користувачем видалення інформації про поточний об’єкт) даних про користувачів всіх типів, команди, секції, програми, заняття, реєстрації гравців на програми, інформації про гравців у складі команд, інформації про опікунів гравців;
- надавати наступну інформацію: перелік футбольних сесій з вказанням регіону; перелік програм зі стастистиками заповненості, прогресом (кількості занять кожної програми, що залишилися), загальною та середньою ціною, а також датою наступного заняття; перелік реєстрацій (enrollments) гравців на конкретну програму, включно з інформацією про їхніх опікунів; перелік занять конкретної програми, зі зручним інтерфейсом розкладу у виді календаря; перелік команд з інформацією про кількість незайнятих місць, гравців команди та можливістю додавати та виключати гравців, що зареєструвалися на програми тренера, з команди.
- підтримувати можливість формування довільного запиту до БД на мові SQL з підтримкою користувача інформацією стосовно схеми БД;
- підтримувати підготовку та друк наступних звітів: звіт про програми, що містить інформацію про перелік програм з даними про тренування, реєстрації, цінами та датами початку і кінця - з фільтрацією за ознакою опублікованості; звіт про команди, що містить інформацію про перелік команд з даними про їхні назви та прогресом набору в команди (кількість набраних гравців vs максимальна кількість гравців в команді) (Додаток Б);
- система повинна реалізовувати задачу відправки електронних листів гравцям з повідомленням про перенесення розкладу занять.

Нефункціональними вимогами клієнтської частини програмного продукту є наступні:

- доступність - програмний продукт має відповідати вимогам доступності, тобто бути доступним для використання людьми з обмеженими можливостями;
- супроводжуваність - додаток передбачає єдиний стиль, для можливості перевикористання та розширення його надалі;
- переносимість - додаток працюватиме в різних сучасних браузерах (Chrome, Mozilla, Safari, Edge), без обмежень операційної системи.

Таким чином, спроєктована задача (комплекс завдань), що буде вирішуватися програмною системою. Вказані завдання є частиною адміністрування футбольних секцій.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Діаграми функціональності ролей “Тренер” та “Менеджер секцій” програмної системи наведено на рис.3.1-3.2.

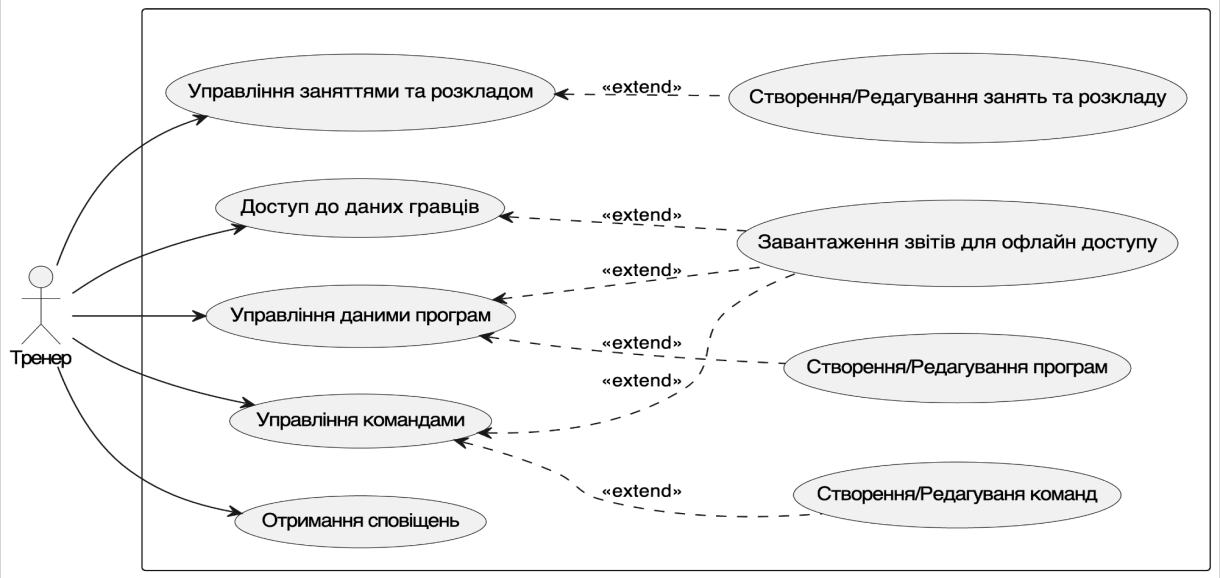


Рисунок 3.1 – USE-CASE діаграма для ролі “Тренер” (рисунок виконаний самостійно)

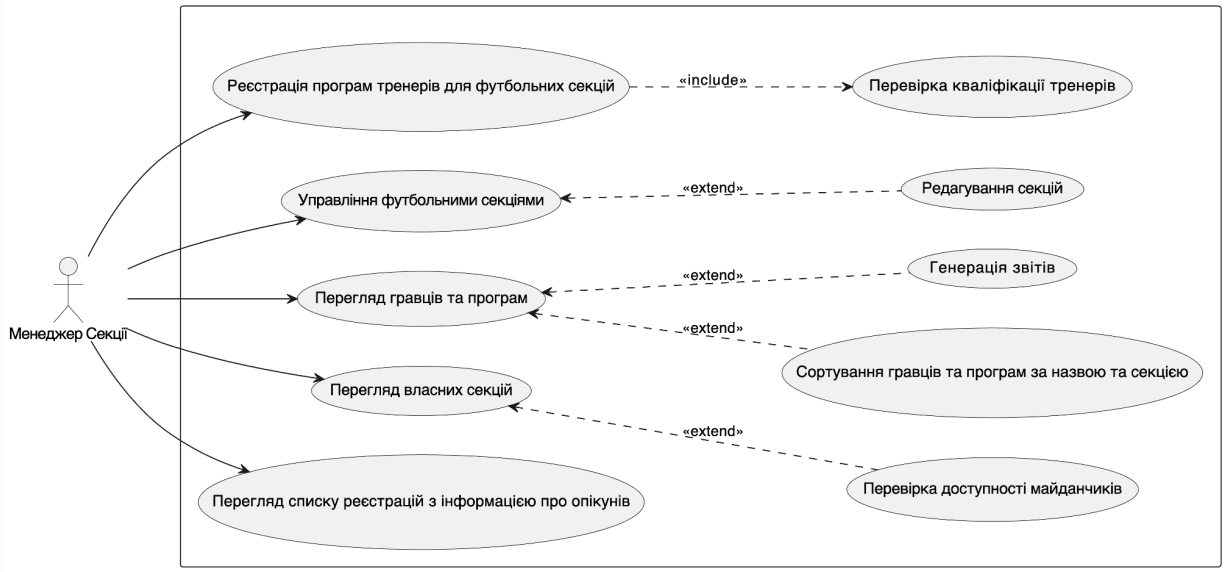


Рисунок 3.2 – USE-CASE діаграма ролі “Менеджер секцій” (рисунок виконаний самостійно)

Варто відмітити, що тренер - це користувач програмної системи; особа, що проводить навчання гравців футболу та працює у закладах освіти, приватних бізнесах, самостійно (підприємці). Програма - структурована навчальна програма, за якою закріплений тренер та яка складається з футбольних занять. Публікація програми - процедура з визначення програми як відкритої для реєстрації, тобто готової для того, щоб гравці мали змогу її бачити та реєструватися на неї. Гравець - користувач; особа, що грає в футбол/навчається грі в футбол за допомогою програмної системи; особа, що приймає рішення. Опікуни/Батьки Гравців - користувач; особа, що відповідає за реєстрацію неповнолітніх гравців, бере участь у виборі тренерів та проводить необхідну комунікацію з організаційних питань.

Зазначимо перелік характеристик для кожного з наведених понять. Поняття секції має наступні властивості: назва, регіон, адреса, призначений менеджер. Поняття “програми” включає в себе назву (наприклад, “Футбольний Фенікс”), опис (наприклад, “Інтенсивна програма для вдосконалення всіх аспектів гри, від фізичної підготовки до психології перемоги.”), обмеження за віком (мінімальний та максимальний вік гравців), ціна в гривнях, ознаку опублікованості, дати старту та початку, дата та час початку та закінчення запису на програму, призначеного тренера, секцію, коментарі (зазвичай, залишені менеджером або тренером), а також вмістимість. Заняття мають зазначені атрибути: асоційовану програму, дату та час проведення. Поняття команди - назву та тренера. Поняття користувача – роль (тренер, гравець, опікун, менеджер, адміністратор), юзернейм, ім'я та прізвище, емейл, номер телефона, контакт та адресу опікуна (якщо існує), дату народження.

Для спрощення роботи тренерів під час роботи з інформацією вони повинні мати можливість: сортувати гравців за ім’ям; сортувати програми та команди за назвою, гравців у складі команди та зареєстрованих гравців - за ім’ям гравця; отримати перелік футбольних сесій з вказанням регіону; отримати перелік реєстрацій гравців на конкретну програму, включно з інформацією про їхніх опікунів; отримати перелік команд з інформацією про кількість незайнятих місць,

гравців команди та можливістю додавати та виключати гравців, що зареєструвалися на програми тренера, з командиж отримати перелік власних програм зі стастистиками заповненості, прогресом (кількості занять кожної програми, що залишилися), загальною та середньою ціною, а також датою наступного заняття.

Оскільки розклад занять може коригуватися тренером, адміністратором або менеджером, є необхідність сповіщати гравців про зміни у розкладі. Запропоновану функціональність зображено за допомогою діаграми станів (рис. 3.3).

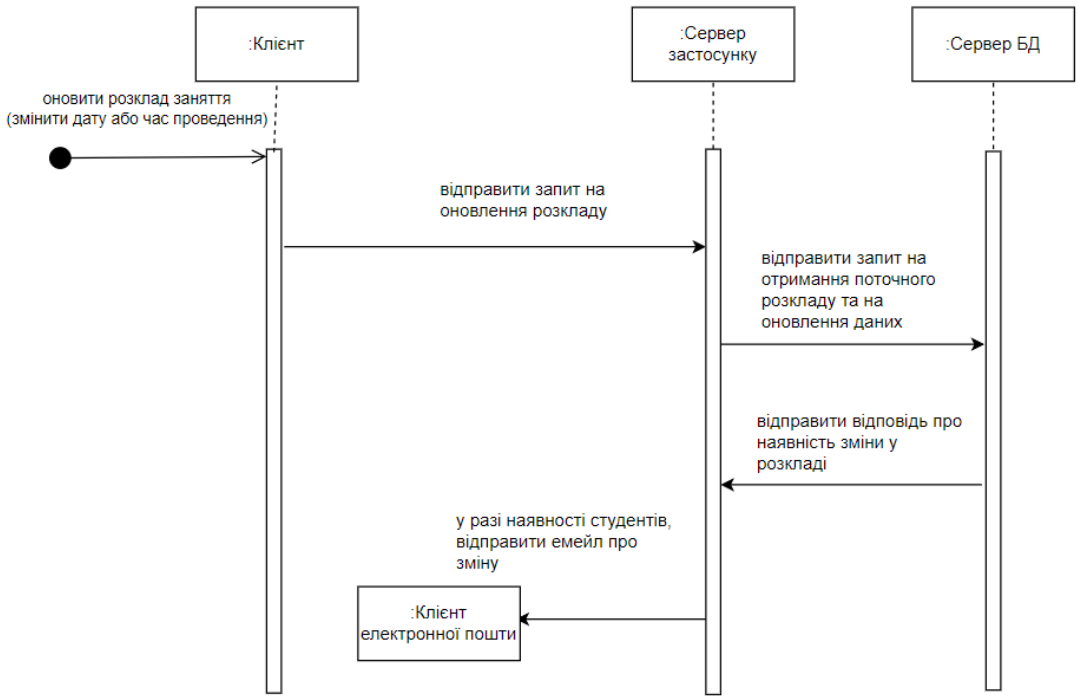


Рисунок 3.3 - Діаграма послідовностей рішення зі сповіщень гравців про зміну розкладу (рисунок виконаний самостійно)

Звіт з команд - це перелік команд з даними про їхні назви та прогресом набору в команди (кількість набраних гравців та максимальна кількість в команді) (див. Додаток Б).

Описання алгоритмічних залежностей показників в програмній системі складається з наступних залежностей: вікі гравця = поточний рік - рік народження; заповеність команди = кількість реєстрацій гравців у складі команди - 15

(максимальна вмістимість); заповненість програми = кількість реєстрацій гравців на програму - максимальна вмістимість програми.

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно ідентифікації: в кожного користувача унікальний юзернейм; в кожної команди унікальна назва.

В проблемній області існує рід обмежень, які можна віднести до обмежень цілісності стосовно зв’язків: кожен гравець може бути зареєстрований у тій самій команді тільки один раз; гравець може бути у складі декількох програм одночасно; гравець може грати за кілька команд (бути у складі одночасно); одна програма керується лише одним тренером; одна програма може бути зареєстрована тільки в одній секції; одна секція керується лише одним менеджером; у гравця може бути декілька опікунів; у опікуна може бути кілька гравців-підопічних.

Таким чином, в рамках цієї роботи було проведено UML проєктування клієнтської частини програмного забезпечення.

3.2 Архітектура клієнтської частини застосунку

Для втілення клієнтської частини даного додатку буде використано середовища розробки JetBrains (WebStorm) [10], мова програмування TypeScript, а також бібліотеку React [11]. Даний технологічний стек дозволяє швидко створити зручний та інтуїтивно зрозумілий додаток.

Для взаємодії клієнтської та серверної частин застосунку обрано підхід RESTful API - абревіатура від REpresentational State Transfer (передача репрезентативного стану) [12,13]. RESTful API - це архітектура інтерфейсу прикладних програм (API), що використовує HTTP-запити для доступу до ресурсів та їхнього використання. Прикладами таких HTTP-запитів є GET, PATCH, POST та DELETE, що дають змогу, відповідно, читати, змінювати, створювати і видаляти ресурси.

REST API на клієнтській стороні буде реалізовано з використанням React та react-query [11]. React також забезпечує наступні переваги для фронтенду: швидкодія (React.js створює настроюваний віртуальний DOM, який працює швидше, ніж звичайний, це підвищує продуктивність додатків і знижує навантаження на браузер); підвищення швидкості розробки завдяки наявності високорівневих функцій та методів, а також компонентній архітектурі; реактивність клієнтського інтерфейсу, що швидко реагує на дії користувача або редагування даних без явного оновлення сторінки.

React на проекті інтегровано з Babel [14] - інструментом для трансполяції кода JavaScript для сумісності зі старими браузерами (дозволяє використовувати функції ECMAScript 6+). Переваги включають об'єднання та оптимізацію файлів JavaScript для ефективної доставки.

TypeScript та JavaScript файли, стилі CSS та інші статичні активи збираються за допомогою технології Webpack [15] - інструменту, що здатний перетворювати файли за допомогою спеціальних завантажувачів. Webpack перетворює різноманітні компоненти проекту на оптимізовану збірку, яка зменшує кількість запитів до сервера і прискорює завантаження сторінки.

3.3 Приклади алгоритмів та методів

Окремої уваги потребує архітектура системи аутентифікації (рис. 3.4). Для аутентифікації користувачів (реєстрація, логін) було обрано технологію “без паролю”, або passwordless, за допомогою системи Passage, розробленої компанією 1Password, яка (система) підтримує біометричну аутентифікацію користувачів [16]. В розпорядженні автора також є бібліотека “passage-react” [17].

Для цілей авторизації фронтенд передаватиме бекенду JWT токен з кожним HTTP запитом; токен зберігається в браузері користувача та діє 15 хвилин [18]. Щойно строк дії токену закінчується, система переводить користувача в анонімний режим (логаут) та переносить на сторінку логіну (рис.3.5).

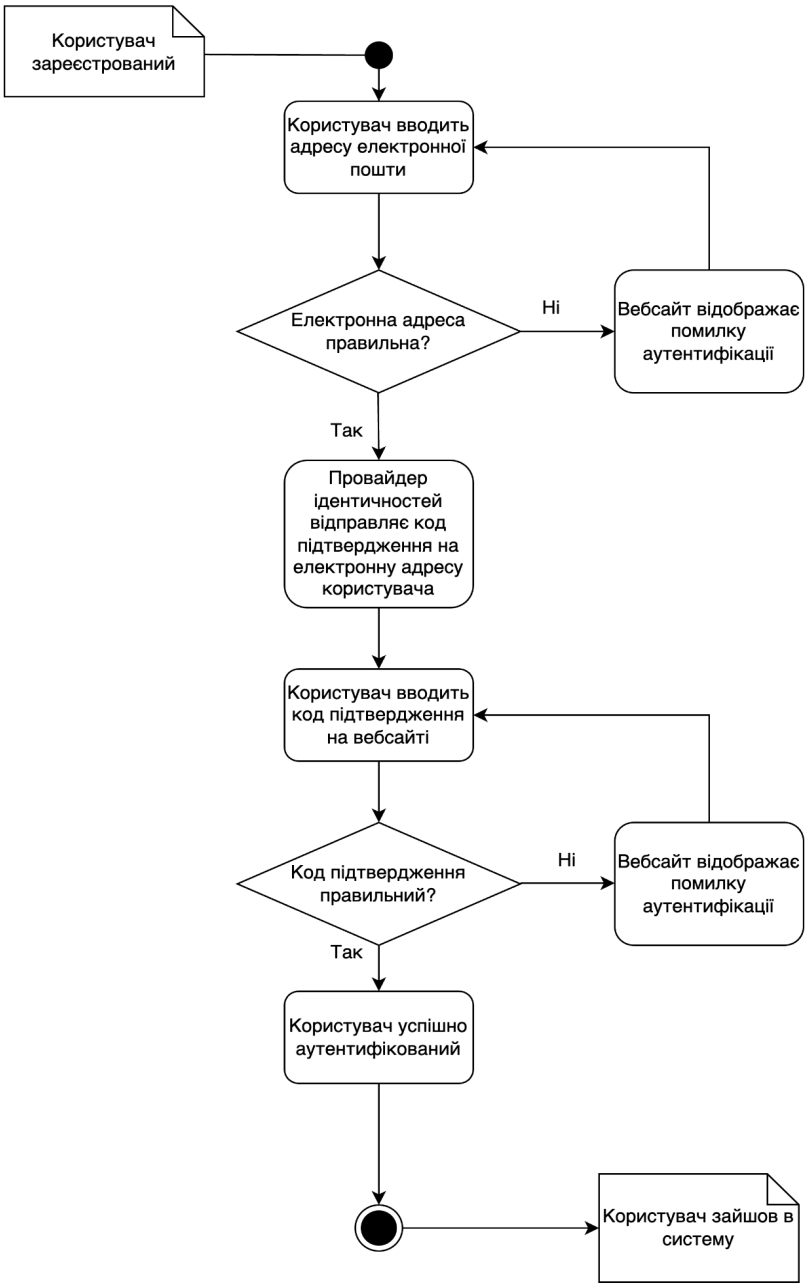


Рисунок 3.4 - Діаграма активностей системи аутентифікації (рисунок виконаний самостійно)

Як вже згадувалося, програмна система підтримуватиме кілька мов (наразі українську та англійську). Інтернаціоналізація та локалізація буде реалізовано повністю у клієнтській частині застосунку за допомогою бібліотеки “i18next” [19]. Для керуваннями залежностями (бібліотеками) фронтенду доцільно використати

Node Package Manager (NPM) [20] - менеджер пакетів JavaScript, який керує залежностями в проектах.

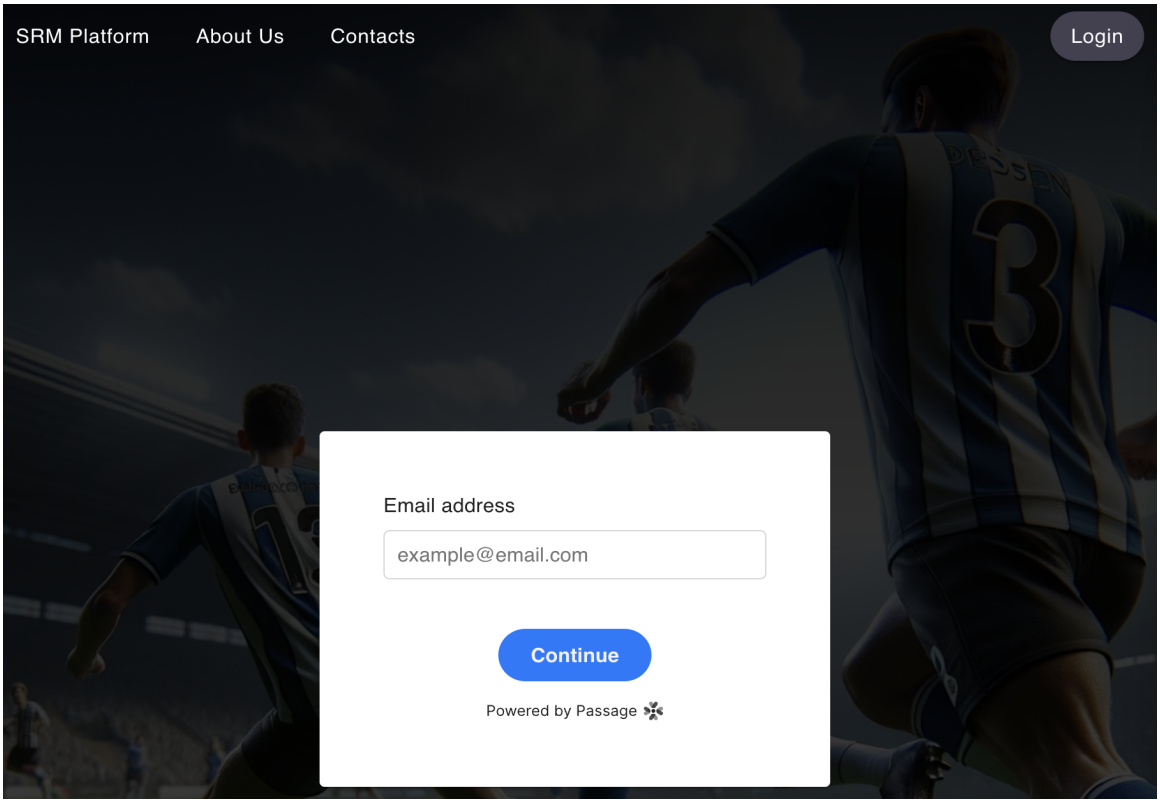


Рисунок 3.5 - Сторінка логіну (рисунок виконаний самостійно)

За допомогою NPM можна керувати встановленням, оновленням та видаленням пакетів. У широкому сенсі NPM є найбільшим у світі реєстром програмного забезпечення. Також за допомогою NPM виконується запуск скриптів для проведення різноманітних операцій з клієнтською частиною застосунку, в тому числі виконання скриптів для отримання фінальних статичних файлів для передачі їх веб-серверу.

Таким чином, було наведено детальний опис програмної реалізації та екосистеми розробки клієнтської частини програмної системи.

В програмній системі буде реалізовано архітектура Flux-Redux з використанням Redux Toolkit (RTK). Це є прогресивним підходом до управління станом додатка. Основна ідея полягає в тому, що весь стан додатка зберігається в

одному централізованому сховищі, а зміни до стану здійснюються лише через диспетчеризовані дії. Redux Toolkit значно спрощує роботу з Redux завдяки зручним утилітам та скороченому синтаксису. У RTK, функція configureStore створює сховище з вбудованими середовищами розробки, такими як Redux DevTools та вбудована підтримка middleware. Слайси в RTK об'єднують логіку дій та редюсерів у компактні модулі, що робить код більш організованим і читабельним. Кожен слайс містить початковий стан, редюсери для визначення, як змінюється стан у відповідь на дії, та автоматично генеровані дії. Також поєднання RTK Toolkit з використанням RTK Query дозволяє ефективно управляти запитами до API та станом серверних даних, що інтегрується з RTK. RTK Query забезпечує кешування, оновлення даних та обробку помилок з мінімальним кодом, що спрощує роботу з асинхронними запитами. Архітектура Flux-Redux з RTK створює чітку структуру для додатків, де компоненти React підписуються на необхідні частини стану і диспетчеризують дії для зміни цього стану, забезпечуючи передбачувану та легку для підтримки архітектуру.

Фрагмент реалізації слайсу сторінки «Користувач» наведено нижче:

```
const userSlice = createSlice({
  name: 'user',
  initialState,
  reducers: {
    setUser: (state, action: PayloadAction<User | undefined>) => {
      state.user = action.payload
      state.authenticated = true
    },
    clearUser: state => {
      removeLocalStorageItem('psg_auth_token')
      removeCookie('psg_auth_token')
      state.user = null
      state.authenticated = false
    },
  },
})
```

Таким чином, використання Redux Toolkit та RTK Query у рамках архітектури Flux-Redux значно підвищує ефективність та зручність розробки сучасних додатків, зменшуючи кількість шаблонного коду та забезпечуючи високу продуктивність і надійність.

3.4 Проєктування користувацького інтерфейсу

Програмна система повинна відображати наступні дані: безпосередньо про основні поняття ПЗ: користувачі (тренери, гравці, опікуни), програми, секції, заняття, команди; про пов’язані поняття ПЗ: опікунів гравців, реєстрації гравців на програми, інформацію про гравців у складі команд.

Простий та зрозумілий інтерфейс (UI/UX) розроблено в Figma [14], спрямований на вирішення задач адміністрування футбольних секцій для всіх категорій користувачів: гравців, тренерів, менеджерів, забезпечуючи мінімалістичний дизайн з чітким розділенням функціоналу.

Система повинна підтримувати арифметичну обробку даних у вигляді обчислювальних полів - згідно наведених мокапів [9]: мінімальної та максимальної дати народження гравців, що зареєструвалися на футбольну програму; суми вартості всіх програм та середньої вартості всіх програм; кількості гравців кожної команди з переліку (рис. 3.6-3.8).

Також програмна система повинна підтримувати сортування, пошук та фільтрацію даних - згідно наведених мокапів: сортувати програми та команди за назвою, гравців у складі команди та зареєстрованих гравців - за ім’ям гравця; здійснювати пошук: програм - за назвою; студентів, що зареєстровані на програму, - за назвою; фільтрувати опубліковані та неопубліковані програми, а також реєстрації на програми - за датою народження студентів, що зареєстровані на програму (рис.3.7-3.12).

Додатково, програмна система повинна: підтримувати виконання наступних запитів до БД, що часто виникають, як-от: отримати перелік футбольних сесій з вказанням регіону; отримати перелік програм зі стастистиками заповненості, прогресом (кількості занять кожної програми, що залишилися), загальною та середньою ціною, а також датою наступного заняття; отримати перелік реєстрацій гравців на конкретну програму, включно з інформацією про їхніх опікунів;

отримати перелік занять конкретної програми, зі зручним інтерфейсом розкладу у виді календаря.

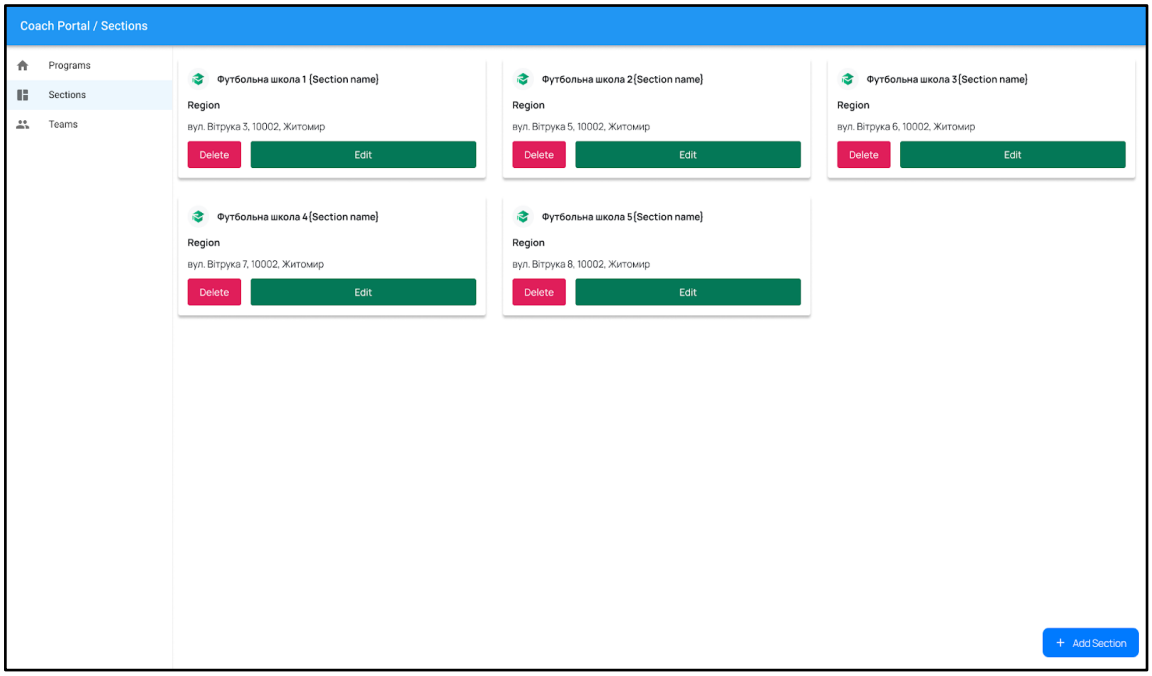


Рисунок 3.6 – Портал Тренерів - Перелік секцій (рисунок виконаний самостійно)

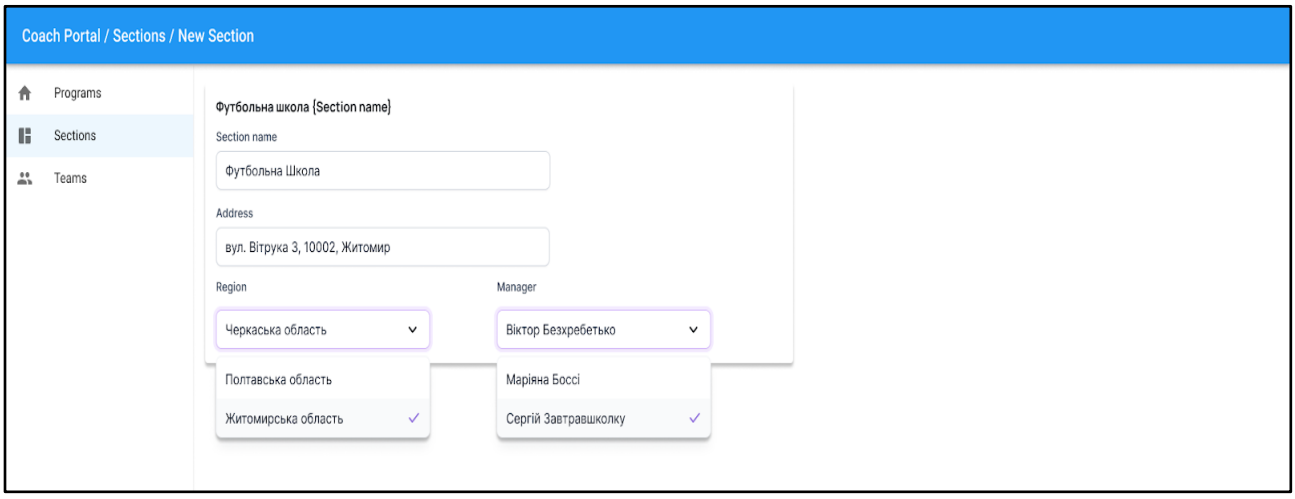


Рисунок 3.7 – Фрагмент порталу Тренерів - Створення Нової Секції (рисунок виконаний самостійно)

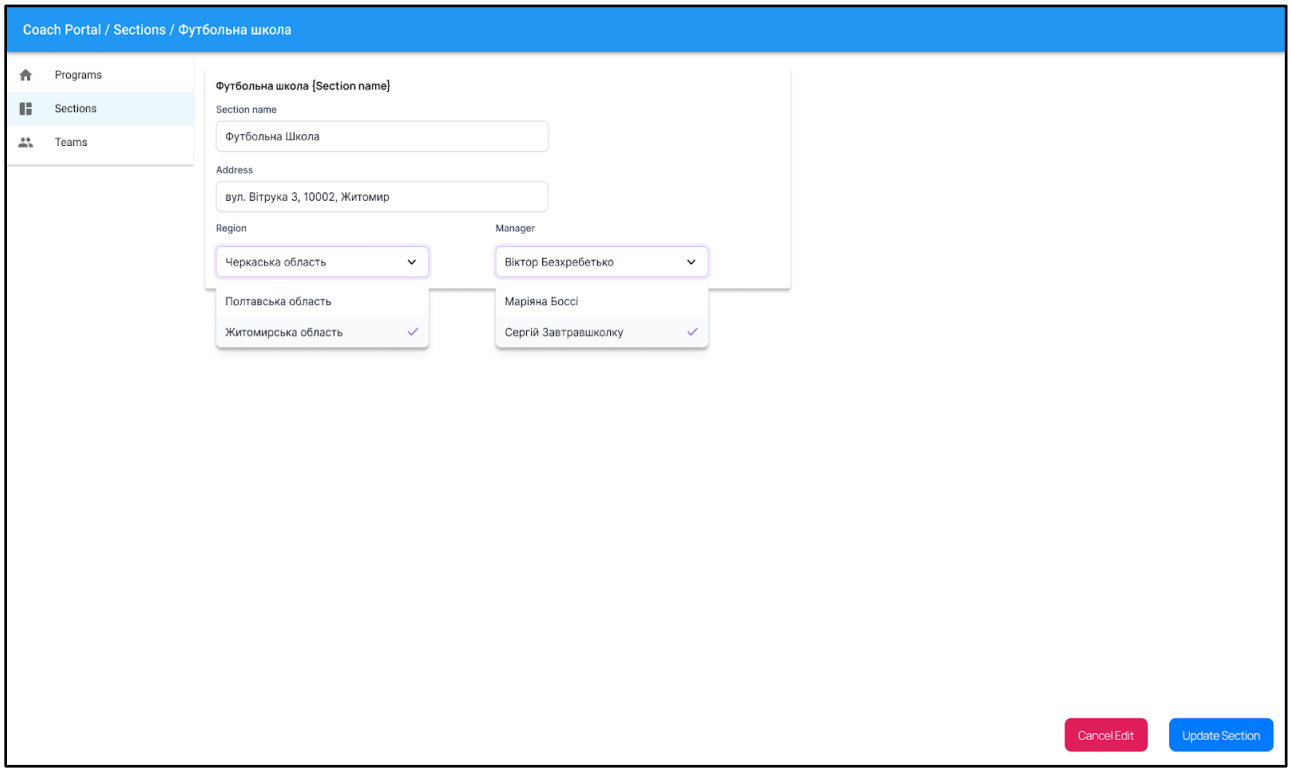


Рисунок 3.8 – Портал Тренерів - Редагування Секції (рисунок виконаний самостійно)

Окрім цього, програмна система дозволятиме отримати перелік команд з інформацією про кількість незайнятих місць, гравців команди та можливістю додавати та виключати гравців, що зареєструвалися на програми тренера, з команди; підтримувати можливість формування довільного запиту до БД на мові SQL з підтримкою користувача інформацією стосовно схеми БД; підтримувати підготовку та друк звіту: про команди, що містить інформацію про перелік команд з даними про їхні назви та прогресом набору в команди (кількість набраних гравців vs максимальна кількість гравців в команді).

В порталі тренерів (див.рис.3.9) підтримуватиметься: сортування за назвою (перший стовпчик - “Name”), пошук за назвою (елемент “Search”), фільтрація опублікованих та неопублікованих програм (вкладинки “Published”, “Unpublished”)

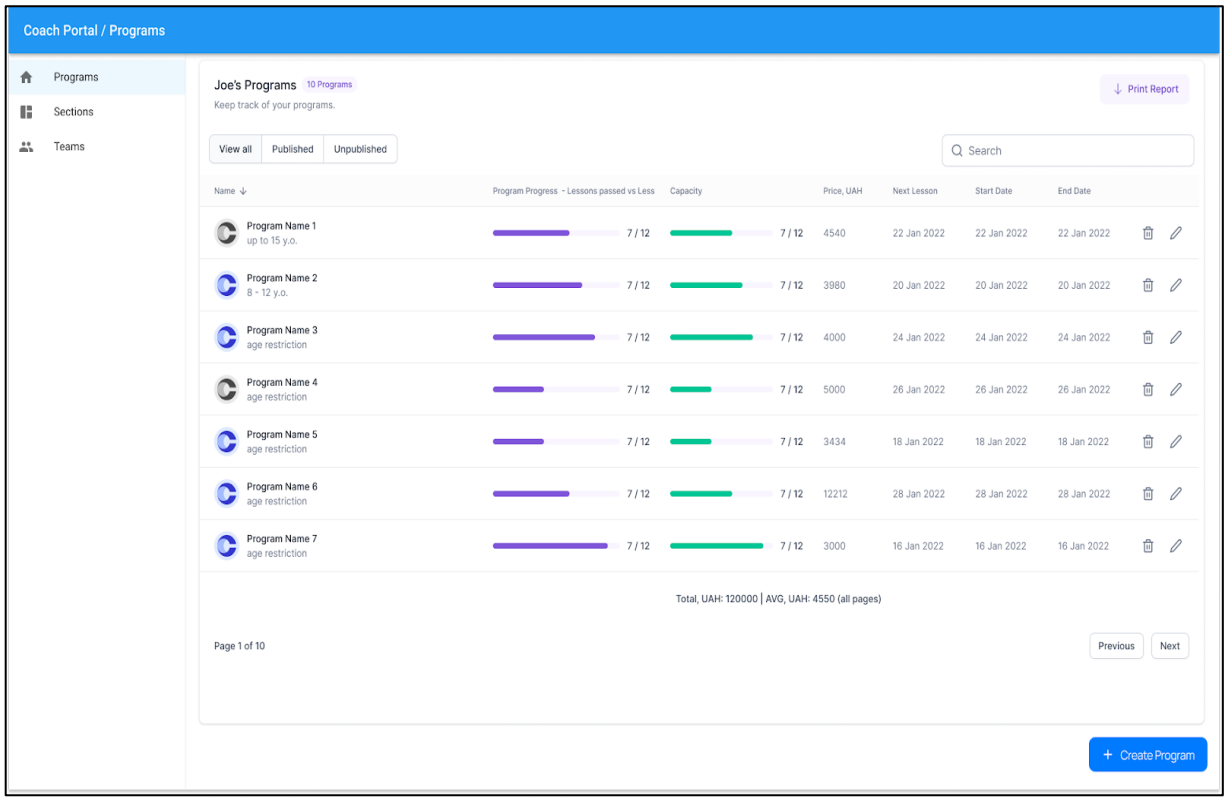


Рисунок 3.9 – Портал Тренерів - Список програм (рисунок виконаний самостійно)

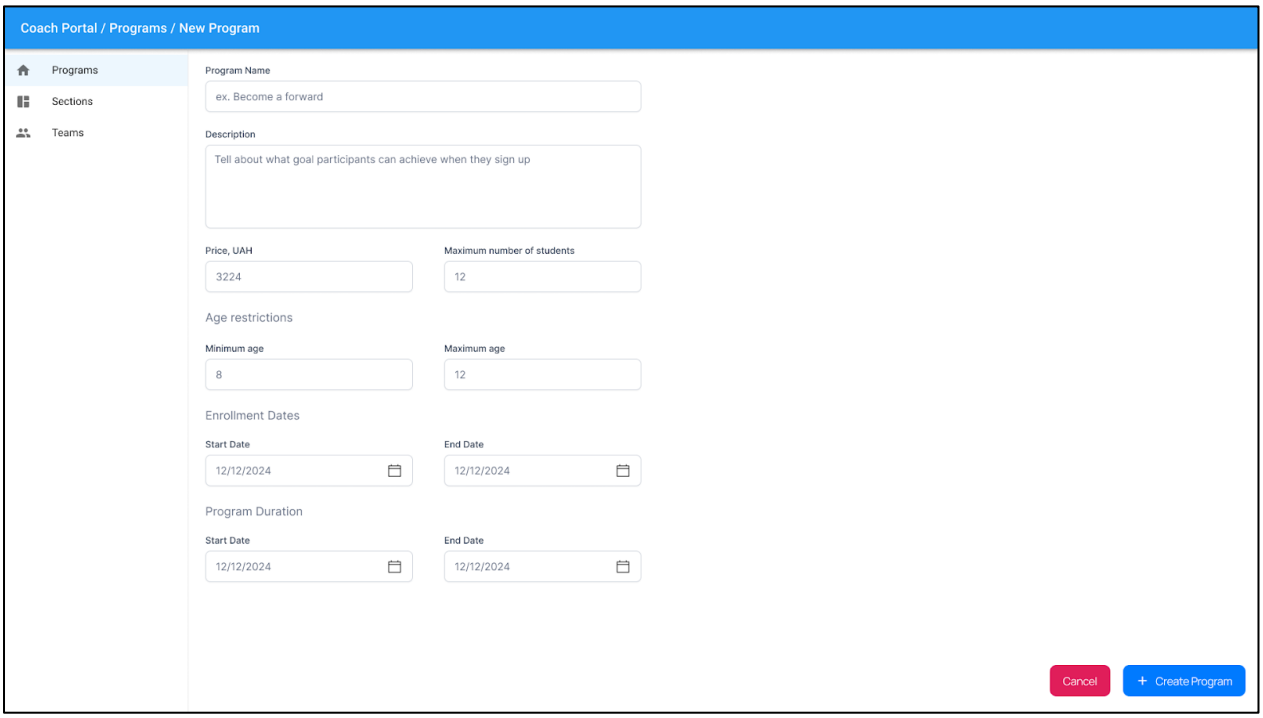


Рисунок. 3.10 – Портал Тренерів - Форма створення програми (рисунок виконаний самостійно)

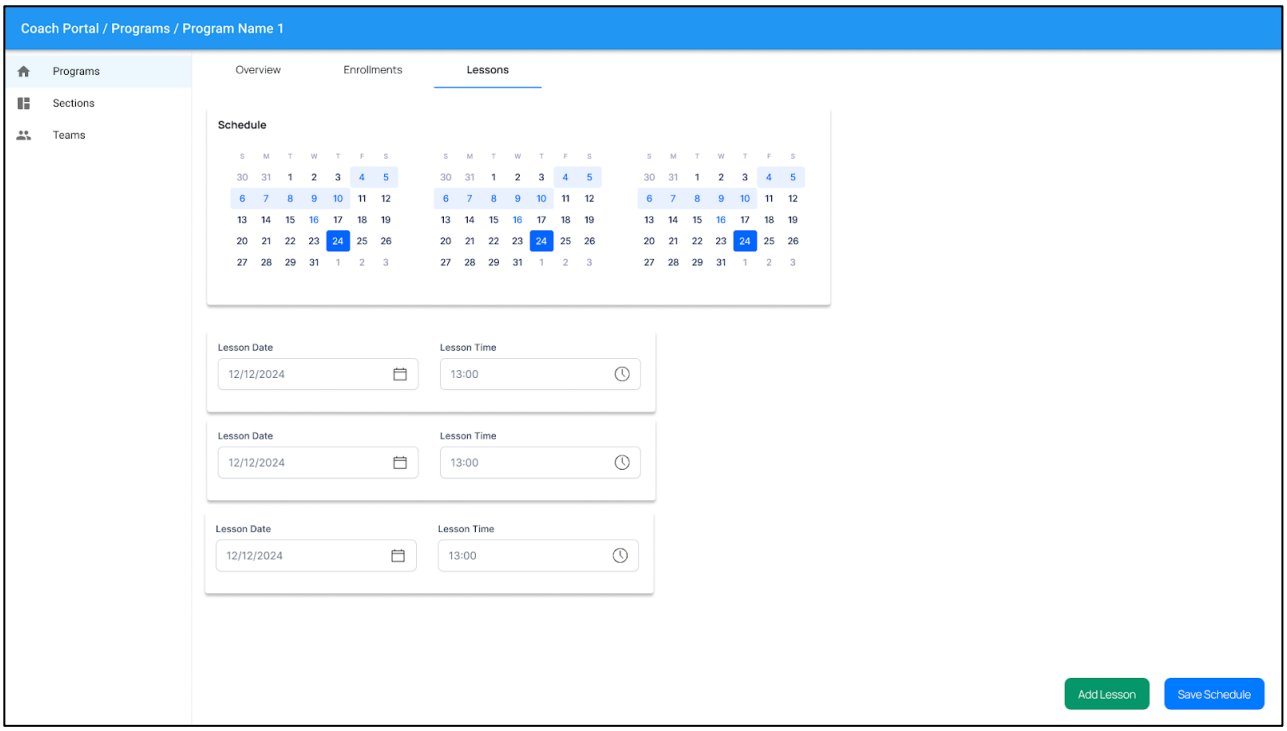


Рисунок 3.11 – Портал Тренерів - Деталі програми - Розклад занять (рисунок виконаний самостійно)

Режимом створення команди підтримуватиметься: сортування команди за назвою (перший стовпчик - “Name”), гравців у складі команди та зареєстрованих гравців - за ім’ям гравця (секції “Team 5”, “Enrolled students” відповідно).

ПЗ буде побудоване за принципом групування по функції або взаємозв'язку. Інформація на екрані забезпечує для користувача можливість перегляду екрану в логічній послідовності; простоту вибору потрібної інформації; можливість ідентифікації пов'язаних груп інформації. Система повинна забезпечувати: розпізнавання виняткових ситуацій (повідомлень про помилки або попередження); можливість визначити, яка дія з боку користувача потрібна (і чи потрібне взагалі) для продовження виконання завдання.

При розробці ПЗ буде дотримано принцип стилістичної цілісності, принцип вирівнювання, принцип використання сталих асоціацій і стереотипів, принцип зручності використання, принцип візуальної ієрархії, принцип сканувальності.

Розроблені мокапи використовують надбання теорії кольору (колірні комбінації є гармонічними).

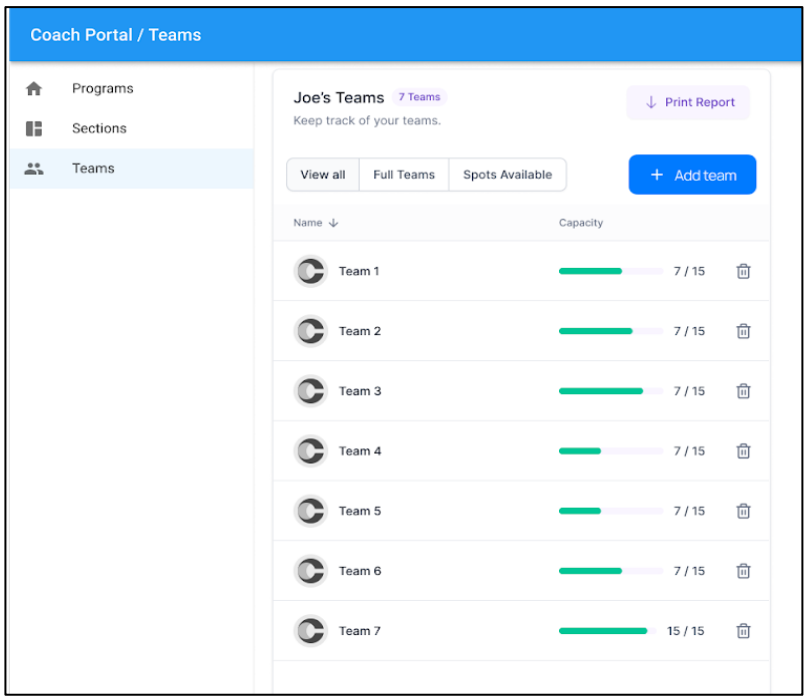


Рисунок 3.12 – Портал Тренерів - Формування Команд. Підтримується: сортування за назвою (перший стовпчик - “Name”) (рисунок виконаний самостійно)

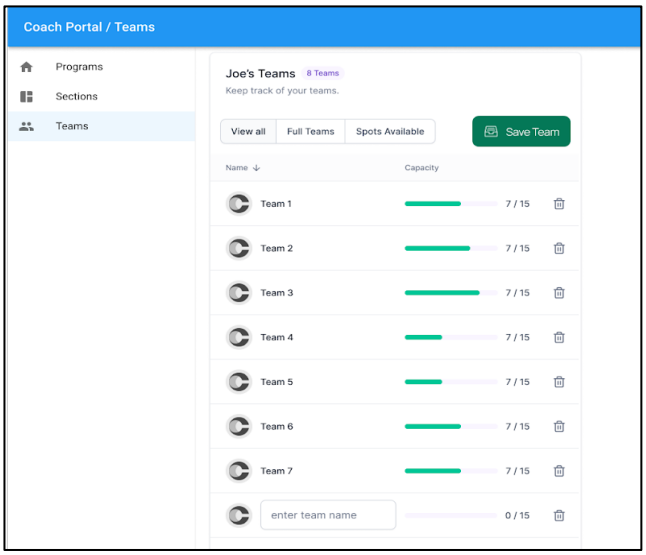


Рисунок 3.13 – Портал Тренерів - Формування Команд - Створення команди. Підтримуєься: сортування за назвою (перший стовпчик - “Name”) (рисунок виконаний самостійно)

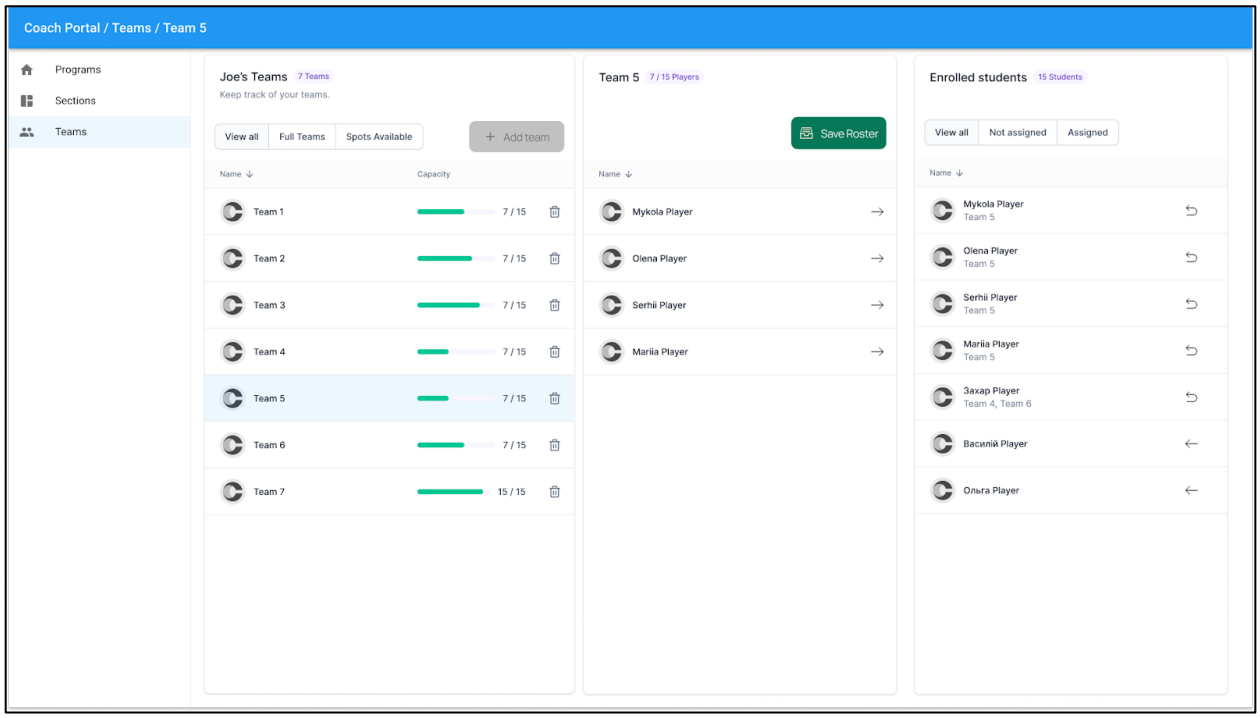


Рисунок 3.14 – Портал Тренерів - Формування Команд - Створення команди (рисунок виконаний самостійно)

Застосунком підтримуватиметься дві мови: українська (за замовчуванням) та англійська (рис 3.15-3.16 відповідно).

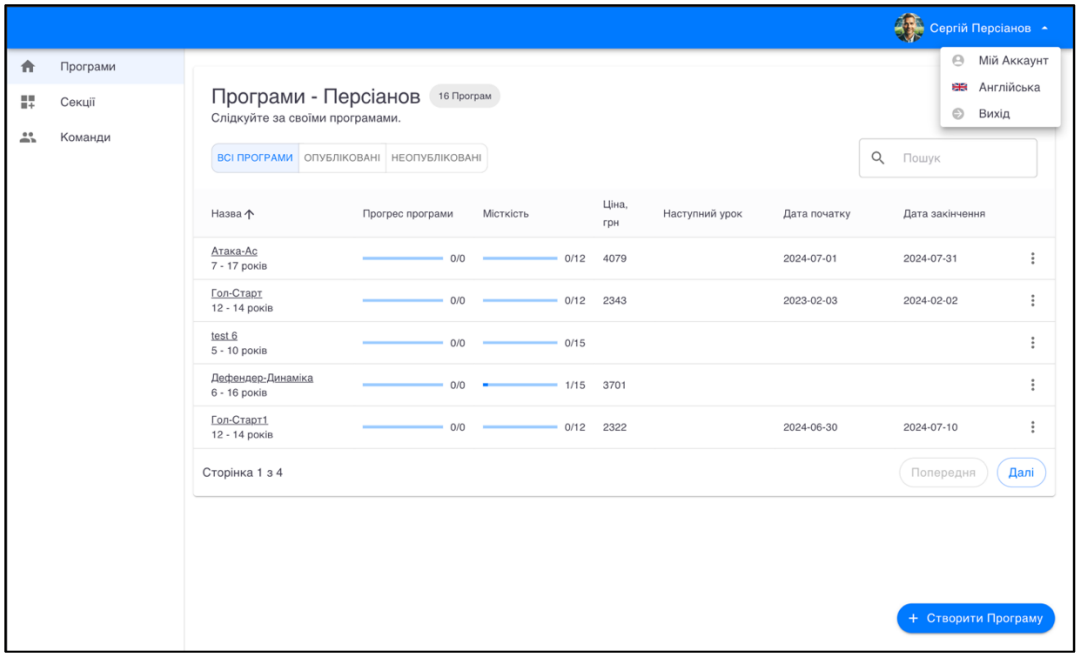


Рисунок 3.15 – Українська версія (рисунок виконаний самостійно)

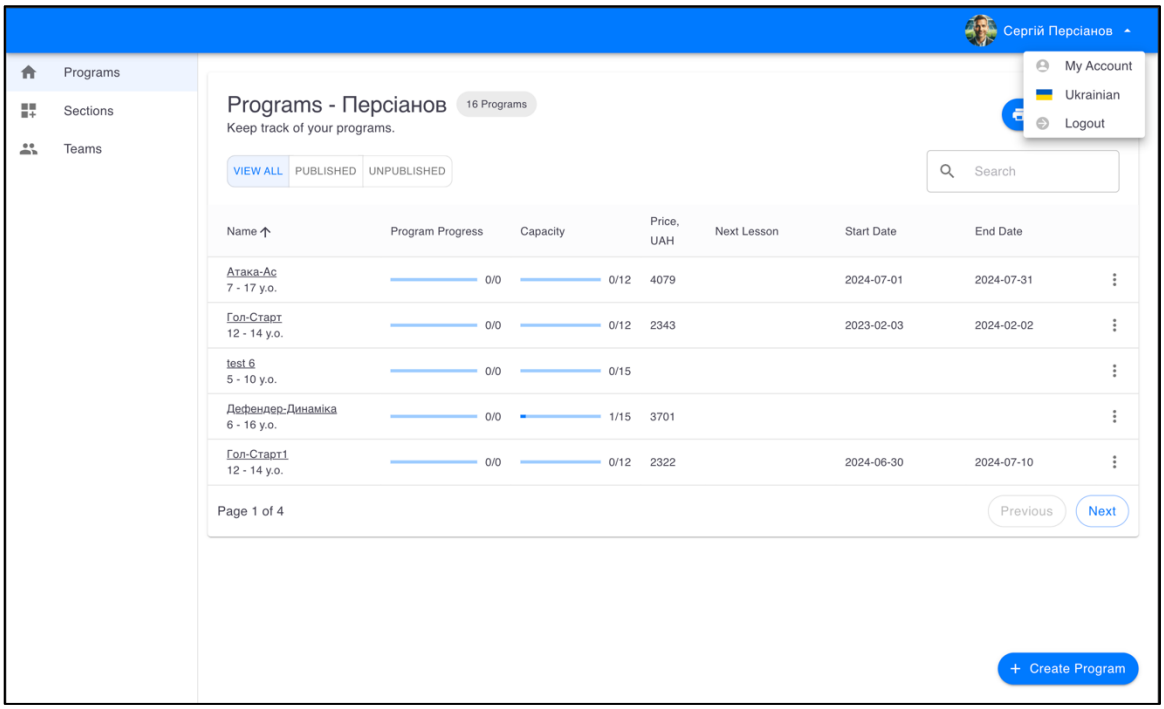


Рисунок 3.16 – Англійська локалізація (рисунок виконаний самостійно)

Інтерфейс перегляду та редагування власного профілю зображено на рис.3.17.

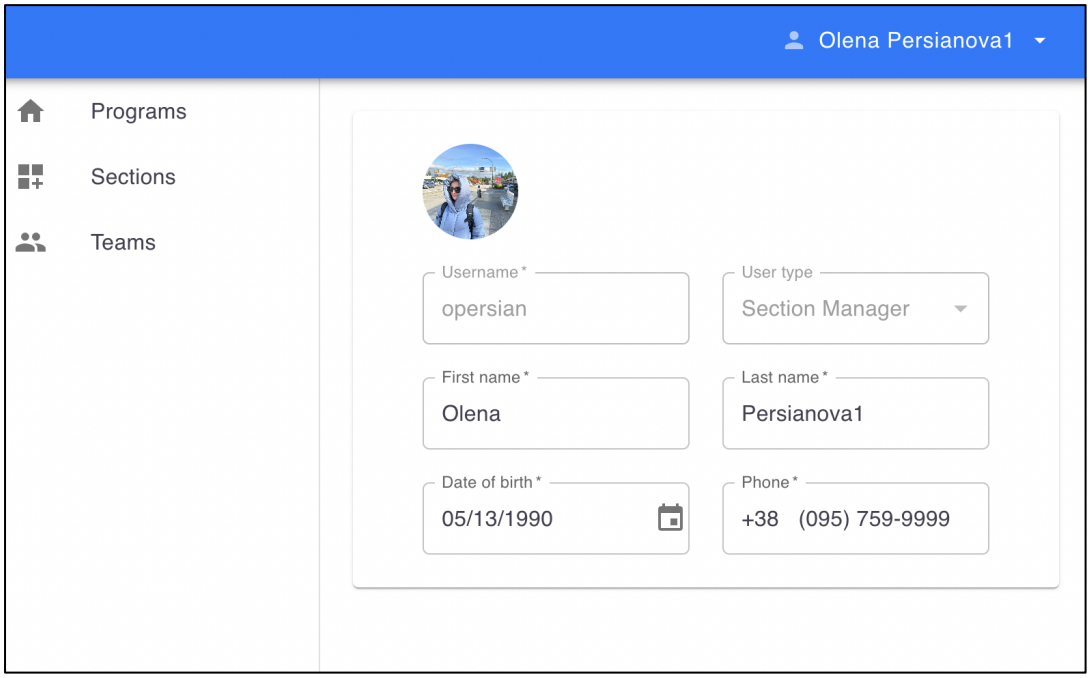


Рисунок 3.17 – Інтерфейс власного профілю (рисунок виконаний самостійно)

Приклади інтерфейсу виводу помилок наведено на рис. 3.18-3.21.

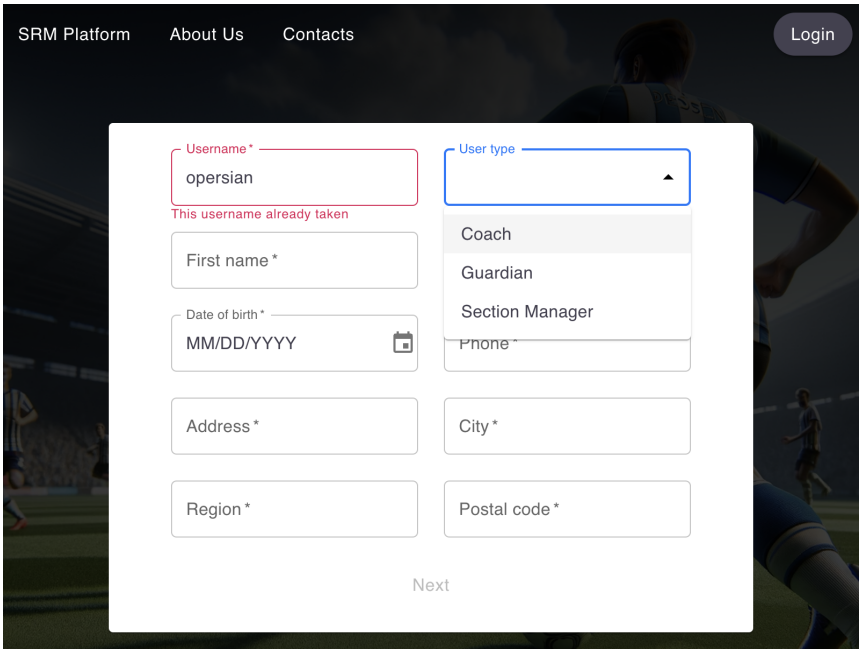


Рисунок 3.18 – Помилка існуючого юзернейму при реєстрації користувача (рисунок виконаний самостійно)

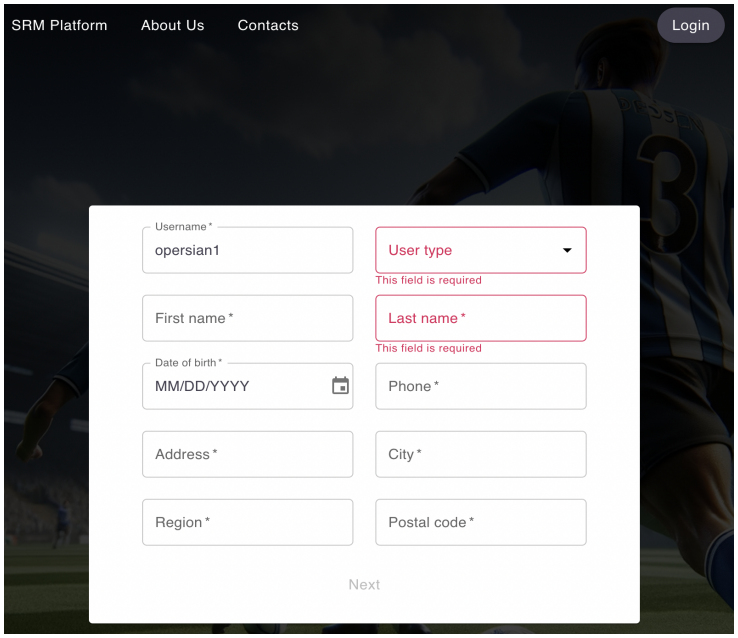


Рисунок 3.19 – Помилка відсутності обов'язкових полів при реєстрації нового користувача (рисунок виконаний самостійно)

Як бачимо, система Figma дозволяє спроєктувати зовнішній вигляд та правила валідації полів користувацького вводу, що є важливою частиною розробки клієнтської частини програмної системи з адміністрування футбольних секцій.

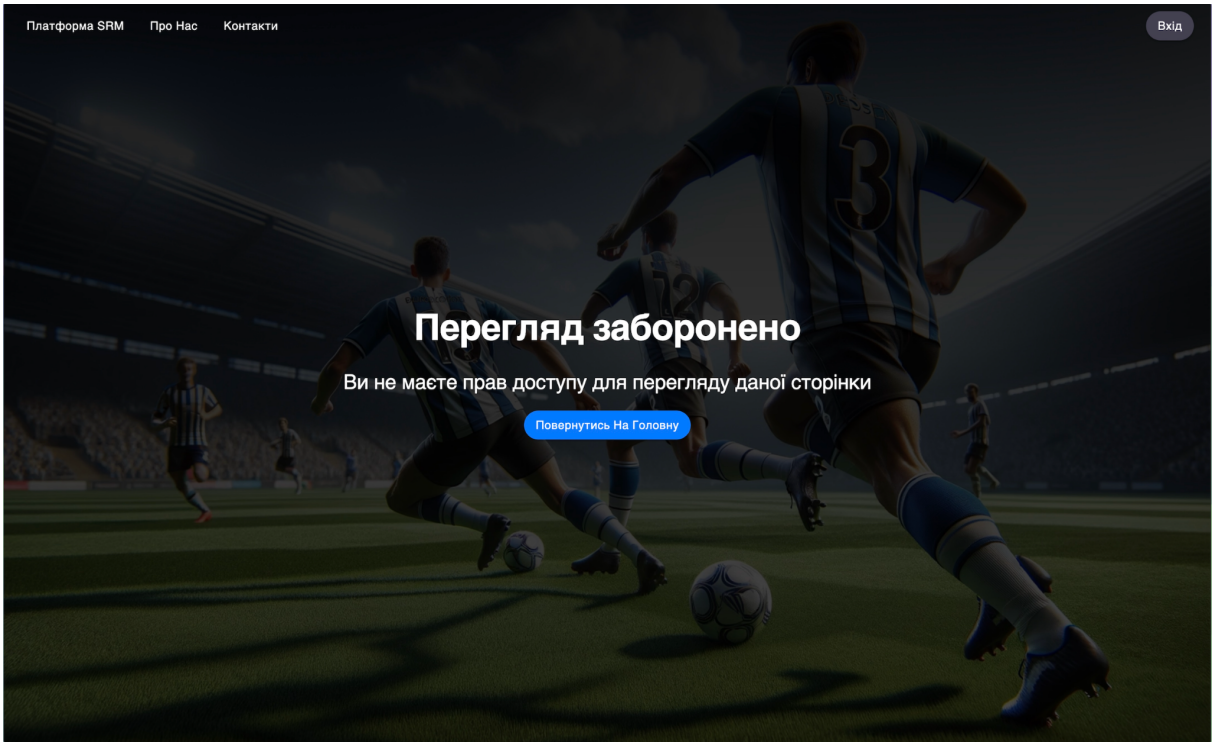


Рисунок 3.20 – Помилка авторизації при спробі переглянути опікуном перелік команд за прямим посиланням (рисунок виконаний самостійно)

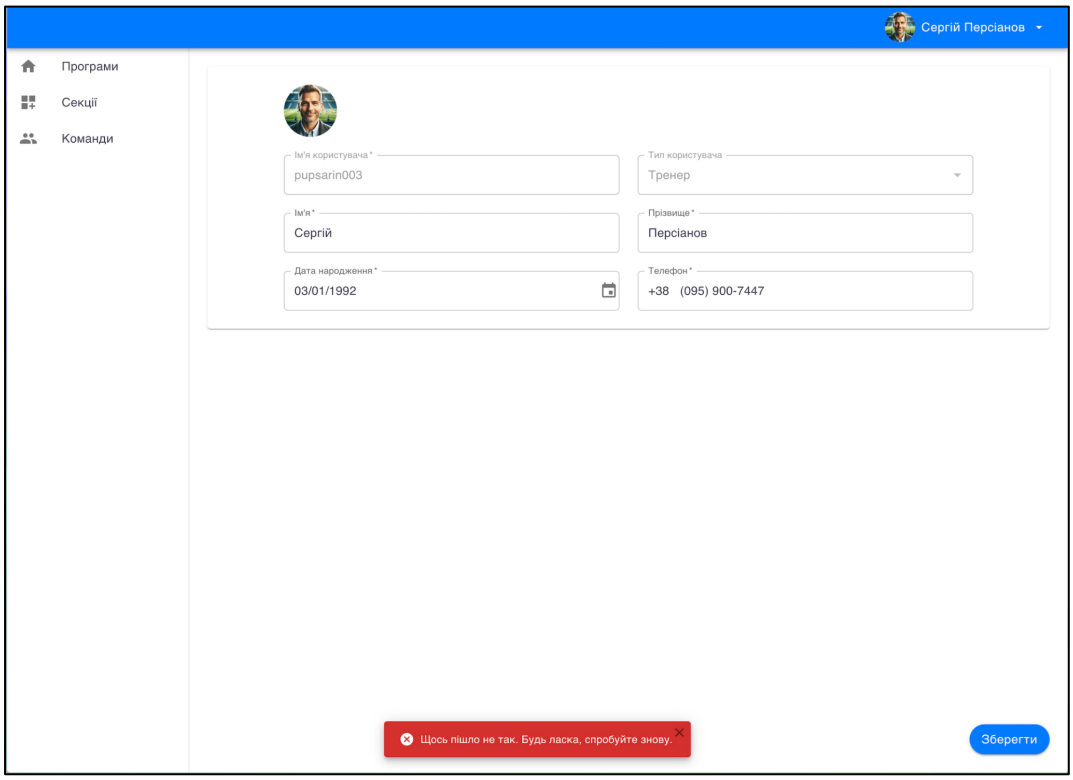


Рисунок 3.21 – Помилка при спробі оновити облікові дані в кабінеті користувача (рисунок виконаний самостійно)

Для створення мокапів використовувався сервіс Figma, що полегшить розробку клієнтської частини програмного застосунку та, зокрема, розробку стилів.

Таким чином, спроєктовано комплекс завдань програмної системи та їхній користувацький інтерфейс, що здебільшого відповідає кращим практикам проєктування UI/UX.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Обґрунтування вибору високорівневого веб-фреймворку

Було обрано React для розробки своїх проектів з кількох важливих причин, які роблять цей фреймворк вигідним і ефективним вибором у порівнянні з Angular та Vue.

Перш за все, React відомий своєю гнучкістю та модульністю. Використовуючи компоненти, розробники можуть легко створювати та повторно використовувати різні частини коду, що сприяє більш організованій і підтримуваній архітектурі застосунків. Це дозволяє швидко адаптуватися до змін вимог та оновлень, що є критичним у сучасному світі швидких технологічних змін. Наприклад, наступний фрагмент коду демонструє створення простого компоненту заголовку та його використання в інших компонентах застосунку:

```
export const Title: React.FC<SRMTypographyProps> = ({
  children,
  isHighlighted,
  isInverted,
  ...otherProps
}) => (
  <Typography
    variant='h3'
    color={getColor({ isHighlighted, isInverted })}
    {...otherProps}
  >
    {children}
  </Typography>
)

...
<form noValidate onSubmit={handleSubmit}>
  <Grid container spacing={3}>
    <Grid item container spacing={3}>
      {isEdit && (
        <Grid item xs={12}>
          <Title>{t('program_form.title')}</Title>
        </Grid>
      )}

...
```

Однією з ключових переваг React є його віртуальний DOM. Використовуючи цю технологію, React мінімізує оновлення реального DOM, що значно підвищує продуктивність застосунків. Коли в додатку відбуваються зміни, React створює віртуальну копію DOM і обчислює найефективніший спосіб оновлення реального DOM. Це дозволяє зменшити кількість операцій оновлення і підвищити швидкість роботи додатку, що особливо важливо для великих та інтерактивних застосунків.

На відміну від Angular, який є повним фреймворком з великим набором вбудованих функцій, React є більш легким і невимушеним у використанні. Angular надає широкий спектр інструментів і директив, які можуть бути корисними, але також додають складності. Крива навчання для Angular є більш крутою, що може бути бар'єром для нових розробників. React, з іншого боку, фокусується на побудові UI-компонентів і дозволяє розробникам обирати інші інструменти та бібліотеки відповідно до потреб проекту. Це дає більше свободи та контролю над процесом розробки.

Vue також є популярним вибором завдяки своїй простоті та низькому порогу входу. Однак, React має значно більшу спільноту та екосистему, що забезпечує кращу підтримку та доступ до різноманітних ресурсів, бібліотек та інструментів. Спільнота React є однією з найбільших у світі фронтенд-розробки, що означає наявність великої кількості готових до використання рішень, плагінів і прикладів, які можуть прискорити процес розробки.

Ще одним важливим аспектом є підтримка з боку великих компаній. React був створений і підтримується Facebook, що гарантує його стабільність і подальший розвиток. Це є важливим фактором для багатьох бізнесів, які шукають довготривалі та надійні технологічні рішення. У порівнянні, Vue не має такого ж рівня підтримки з боку великих корпорацій, що може викликати деякі сумніви щодо його майбутнього розвитку.

React також відрізняється своєю філософією і підходом до розробки. Він заохочує функціональне програмування та використання чистих функцій, що

сприяє написанню більш передбачуваного та легкого для тестування коду. Це дозволяє розробникам створювати додатки, які є більш надійними та простими в обслуговуванні.

4.2 Обґрунтування вибору сервісу аутентифікації

Було обрано Passage by 1Password для авторизації користувачів у проєкті, оскільки цей сервіс пропонує високий рівень безпеки і простоту використання. 1Password, відомий своїми продуктами для зберігання паролів і особистих даних, гарантує, що Passage використовує передові методи шифрування для захисту даних користувачів. Однією з основних переваг Passage є шифрування паролів на стороні клієнта, що означає, що навіть якщо хакери отримають доступ до серверів, вони не зможуть розшифрувати паролі без ключів шифрування, які зберігаються лише на пристроях користувачів. Це значно знижує ризик компрометації даних і підвищує довіру користувачів до безпеки системи.

Auth0 by Okta [21] також є потужним рішенням для управління ідентифікацією та доступом, надаючи широкий набір функцій для багатофакторної аутентифікації, соціальних входів, ідентифікаційного менеджменту та аналітики. Auth0 підтримує різні стандарти безпеки, такі як OAuth2, OpenID Connect та SAML, що робить його універсальним інструментом для забезпечення безпеки у великих корпоративних середовищах. Однак, через великий набір функцій, Auth0 може бути складним у налаштуванні та інтеграції, особливо для невеликих команд або проектів з обмеженими ресурсами. Крім того, витрати на використання Auth0 можуть бути значними для малого бізнесу або стартапів, що змушує шукати більш економічні альтернативи.

Firebase Authentication, створений Google, є ще одним популярним вибором для розробників, особливо тих, хто вже використовує інші сервіси Firebase. Firebase Authentication забезпечує легку інтеграцію з Firebase Database, Firestore, та іншими сервісами, що робить його привабливим для розробки мобільних та веб-

застосунків. Firebase підтримує входи через електронну пошту, соціальні мережі (Facebook, Google, Twitter) та анонімні входи, що робить його зручним для широкого спектру користувачів. Однак, Firebase Authentication має деякі обмеження в налаштуваннях безпеки та управлінні користувачами порівняно з більш гнучкими рішеннями, такими як Auth0 або Passage.

Passage by 1Password виділяється серед інших тим, що він створений з урахуванням потреб користувачів у високій безпеці та простоті використання. На відміну від Firebase Authentication, Passage надає розширені можливості для налаштування безпеки і шифрування даних, що робить його більш надійним вибором для проектів, де безпека є критично важливою. Крім того, інтеграція Passage є достатньо простою, що дозволяє швидко впровадити цей сервіс у додаток без значних витрат часу і ресурсів.

Auth0, хоч і має широкий спектр функцій, може бути занадто складним для проектів, які не потребують усіх можливостей, які він надає. Налаштування Auth0 може вимагати значних зусиль і часу, що може бути недоцільним для невеликих команд. У таких випадках Passage може стати ідеальним компромісом, надаючи необхідний рівень безпеки та зручність без надмірної складності.

Firebase Authentication є хорошим вибором для проектів, де інтеграція з іншими сервісами Firebase є критичною, і де вимоги до безпеки не настільки високі. Firebase забезпечує швидкий старт і легкість у використанні, що робить його привабливим для стартапів та невеликих проектів. Проте, для проектів з вищими вимогами до безпеки і кастомізації, Passage може бути більш придатним вибором.

Отже, вибір Passage by 1Password ґрунтується на поєднанні високої безпеки, простоти інтеграції та налаштування, що робить його відмінним вибором для проектів, які потребують надійного захисту даних користувачів та зручності використання. У той час як Auth0 та Firebase Authentication також мають свої переваги, Passage забезпечує оптимальний баланс між безпекою, функціональністю та простотою, що робить його вигідним вибором для багатьох розробників.

Зручний та інтуїтивно зрозумілий інтерфейс адміністрування зареєстрованого ресурсу системи адміністрування футбольних секцій в Passage/1Password [16] наведений на рис.4.1.

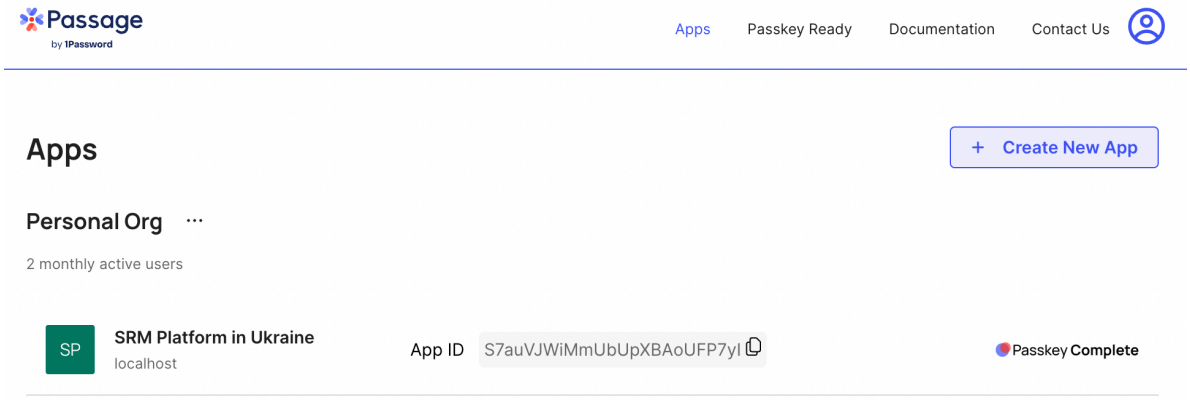


Рисунок 4.1 - Зовнішній вигляд інтерфейсу адміністрування системи аутентифікації Passage/1Password (рисунок виконаний самостійно)

Фрагмент реалізації сторінки «Логін» наведено нижче:

```
export default function Login() {
  const dispatch = useAppDispatch()
  const [getUserByEmail] = useLazyGetUserByEmailQuery()
  const [validateUser] = useValidateUserMutation()
  const { getCurrentUser } = usePassage()
  const handleLogin = useCallback(async () => {
    const userInfo = await getCurrentUser()?.userInfo()
    if (userInfo?.email == null) {
      throw new Error('Passage login failed')
    }
    await validateUser(userInfo.email)
    const response = await getUserByEmail(userInfo.email)
    const user = response?.data?.results[0]
    dispatch(setUser(user))
  }, [dispatch, getCurrentUser, getUserByEmail, validateUser])

  return (
    <LandingLayout>
      <Paper elevation={1} className={classNames['paper']}>
        <PassageTheme />
        <PassageLogin onSuccess={handleLogin} />
      </Paper>
    </LandingLayout>
  )
}
```

4.3 Обґрунтування вибору мови програмування Typescript

Було обрано TypeScript для розробки свого проекту замість JavaScript з кількох вагомих причин, які роблять TypeScript привабливим і ефективним інструментом для сучасної розробки програмного забезпечення.

Перш за все, TypeScript є надмножиною JavaScript, що означає, що він включає всі можливості JavaScript, але додає до них статичну типізацію. Це дозволяє розробникам визначати типи змінних, параметрів функцій, повертаємих значень та інших структур даних, що допомагає запобігати багатьом помилкам ще на етапі компіляції. У випадку з JavaScript, помилки типів можуть бути виявлені лише під час виконання програми, що ускладнює налагодження і тестування, особливо у великих проектах. Наприклад, наступний фрагмент коду демонструє використання інтерфейсів та типів у TypeScript:

```
export interface ApiResponse<T> {
  count: number
  next: string | null
  previous: string | null
  results: T[]
}

export interface Id {
  id: number
}

export interface User extends Id {
  [PROP_USERNAME]: string
  [PROP_FIRST_NAME]: string
  [PROP_LAST_NAME]: string
  [PROP_EMAIL]: string
  [PROP_USER_TYPE]: UserTypes
  [PROP_PHONE_NUMBER]: string
  [PROP_DOB]: string
  [PROP_PHOTO]: string
  [PROP_SHIPPING_ADDRESS]?: string
}
```

Статична типізація, яку надає TypeScript, значно підвищує передбачуваність і стабільність коду. Розробники можуть легко зрозуміти, які типи даних очікуються в різних частинах програми, що спрощує процес розробки і знижує ймовірність

виникнення помилок. Це особливо важливо для великих команд, де кілька розробників працюють над одним проектом і потребують чіткого визначення інтерфейсів та взаємодій між компонентами.

Крім того, TypeScript забезпечує кращу підтримку сучасних функцій JavaScript. Він дозволяє розробникам використовувати нові можливості ECMAScript, такі як асинхронні функції, генератори, модулі та багато іншого, навіть якщо ці функції ще не підтримуються всіма браузерами. TypeScript компілює код у JavaScript, який є сумісним зі старішими версіями ECMAScript, що забезпечує зворотну сумісність і підтримку широкого спектру платформ.

Ще однією значною перевагою TypeScript є його інтеграція з сучасними інструментами розробки. Більшість популярних редакторів коду, таких як Visual Studio Code, WebStorm та інші, мають потужні інструменти для роботи з TypeScript, включаючи автодоповнення, перевірку типів у реальному часі та рефакторинг коду. Це значно підвищує продуктивність розробників і дозволяє швидше знаходити та виправляти помилки.

TypeScript також забезпечує покращену підтримку об'єктно-орієнтованого програмування (ООП). Він дозволяє використовувати класи, інтерфейси, абстрактні класи та інші концепції ООП, що сприяє створенню більш структурованого та організованого коду. Це особливо корисно для великих проектів, де необхідно підтримувати складну ієрархію класів і залежностей між ними.

Підтримка TypeScript з боку великих компаній і спільноти також є важливим фактором. TypeScript був розроблений і підтримується компанією Microsoft, що гарантує його стабільність і подальший розвиток. Крім того, багато великих компаній, таких як Google, Airbnb, Slack та інші, використовують TypeScript у своїх проектах, що свідчить про його надійність і ефективність.

TypeScript також надає розширені можливості для роботи з сучасними фреймворками і бібліотеками. Наприклад, Angular, один з найпопулярніших

фреймворків для розробки веб-застосунків, був створений з урахуванням використання TypeScript. Це забезпечує глибоку інтеграцію і покращену підтримку інструментів розробки, що робить розробку на Angular з TypeScript більш ефективною та продуктивною.

4.4 Обґрунтування вибору бібліотеки керування станом застосунку

Було обрано Redux Toolkit для управління станом застосунка замість MobX з кількох ключових причин, що роблять цей інструмент привабливим і ефективним для сучасної розробки.

Redux Toolkit, який розроблений командою Redux, був створений для спрощення роботи з Redux, що часто вважається складним і громіздким у своїй базовій формі. Однією з основних переваг Redux Toolkit є те, що він забезпечує набір інструментів для автоматизації багатьох типових завдань, таких як створення дій і редукторів, а також налаштування стану. Це значно зменшує кількість шаблонного коду, який потрібно писати, і дозволяє розробникам зосередитися на бізнес-логіці додатку. Використовуючи createSlice, розробники можуть легко створювати редуктори і дії, що робить процес написання коду більш інтуїтивним і менш схильним до помилок.

Однією з основних особливостей Redux Toolkit є його підтримка асинхронних операцій через createAsyncThunk. Це дозволяє обробляти асинхронні запити в додатку з мінімальними зусиллями, забезпечуючи чітку і зрозумілу структуру для управління побічними ефектами. У порівнянні з традиційним підходом Redux, який часто потребує використання додаткових бібліотек, таких як redux-thunk або redux-saga для управління асинхронними операціями, Redux Toolkit надає все необхідне "з коробки".

З іншого боку, MobX використовує реактивний підхід до управління станом, що означає, що будь-які зміни в стані автоматично відображаються в компоненті, який використовує цей стан. Це може бути дуже зручним, оскільки розробникам не

потрібно явно передавати стан або викликати функції для його оновлення. Проте, ця "магічність" MobX може ускладнювати розуміння і налагодження коду, особливо в великих проектах. Крім того, MobX може стати складним у підтримці через його імпліцитну природу оновлень стану, що може призвести до непередбачуваних наслідків і складних для виявлення помилок.

Redux Toolkit також забезпечує кращу інтеграцію з сучасними інструментами розробки. Завдяки своїй чіткій структурі і явному підходу до управління станом, він дозволяє легко використовувати інструменти для налагодження, такі як Redux DevTools (рис.4.2), що значно полегшує процес розробки і діагностики проблем. Це особливо важливо для великих команд, де можливість легко відстежувати і розуміти стан додатку є критичною для ефективної співпраці і підтримки коду.

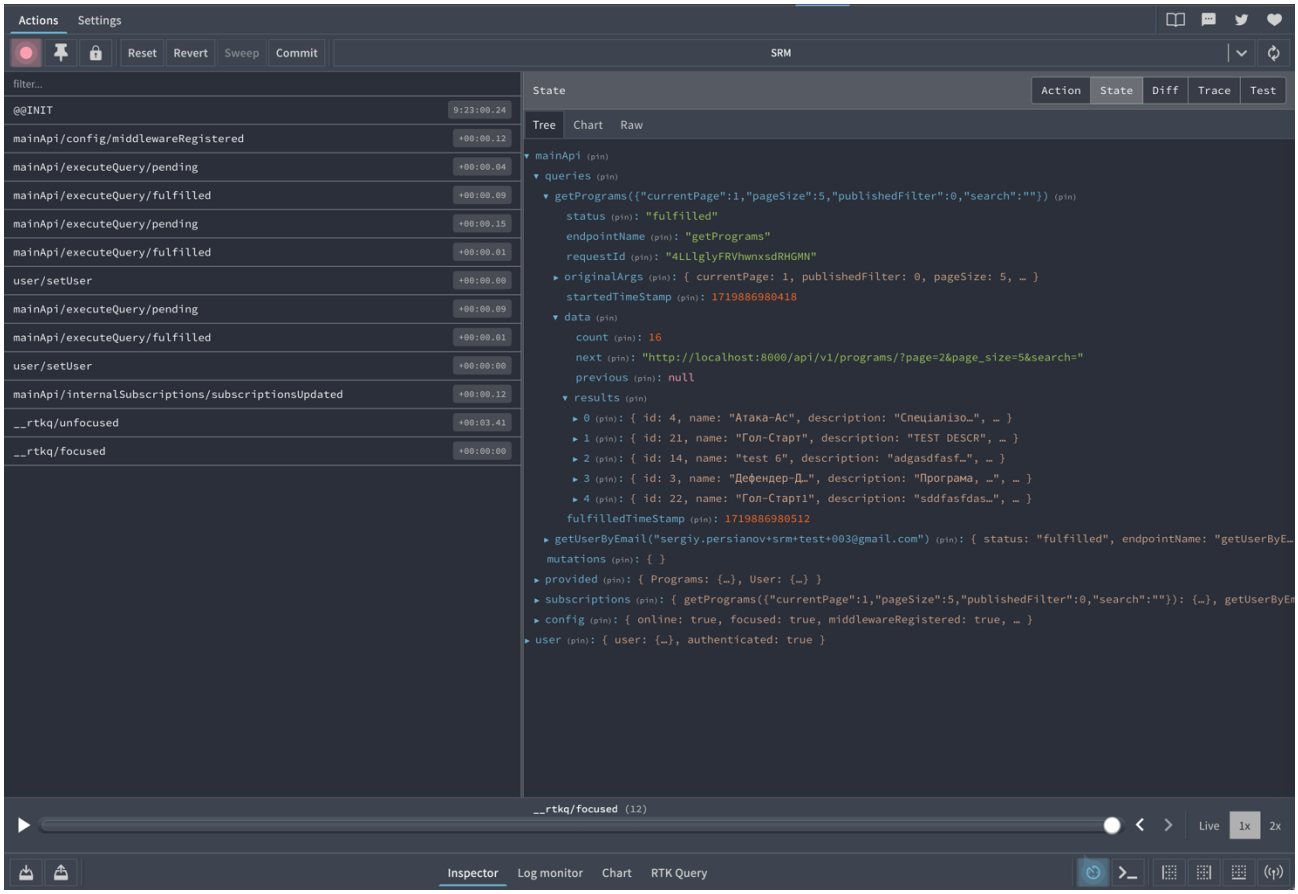


Рисунок 4.2 – Історія дій згенерованих Redux Toolkit відображена у Redux DevTools (рисунок виконаний самостійно)

Ще однією перевагою Redux Toolkit є його сумісність з TypeScript. TypeScript, який забезпечує статичну типізацію і покращену перевірку типів на етапі компіляції, є популярним вибором серед розробників для створення надійного і підтримуваного коду. Redux Toolkit надає відмінну підтримку TypeScript, забезпечуючи типізацію дій, станів і редукторів, що значно покращує зручність роботи з кодом і зменшує ймовірність помилок.

MobX також підтримує TypeScript, але його реактивна природа може ускладнювати використання типів, особливо коли йдеться про складні структури даних і взаємодії між ними. Це може створити додаткові труднощі при роботі з великими проектами і складними бізнес-логіками, де точність і передбачуваність є критичними.

Крім того, Redux Toolkit надає готові шаблони для створення застосунків з використанням сучасних практик і підходів. Це включає в себе підтримку іммутабельності стану, яка є однією з основних концепцій Redux, що допомагає уникати непередбачуваних змін стану і полегшує тестування. MobX, з його імпліцитною реактивністю, може створювати складнощі з підтримкою іммутабельності і ускладнювати процес тестування.

4.5 Інтеграція та використання Sass і Material UI

В рамках даної роботи було реалізовано інтеграцію стилів за допомогою бібліотеки Material UI та препроцесора Sass, що дозволило забезпечити консистентність стилів і підвищити ефективність розробки інтерфейсу користувача. Основним джерелом правди і стилів у проекті виступає Sass, а Material UI використовується для швидкої побудови компонентів з урахуванням цих стилів.

У проекті було застосовано Sass як основний інструмент для визначення та організації стилів. Це дозволило створити єдиний набір змінних для кольорів, типографіки, розмірів, відступів та інших стилістичних параметрів, які можна

легко використовувати в будь-якому місці проекту. Наприклад, у файлі зі змінними Sass містяться всі основні кольори, розміри шрифтів, відступи та брейкпойнти:

```
$blue-dark: #007AFF;
$grey-dark: #444150;
$red: #E01E5A;
$green: #87be41;
$h-1-font-size: 3.125rem;
$h-2-font-size: 1.75rem;
$breakpoints: (
        sm: 30rem,
        md: 49rem,
        lg: 64rem,
        xl: 96rem,
);
```

Цей підхід дозволив централізовано керувати стилями і легко змінювати їх при необхідності.

Material UI був використаний для побудови компонентів, що значно прискорило розробку інтерфейсу. Було реалізовано інтеграцію з Sass, щоб компоненти Material UI використовували ті ж самі стилі та змінні, що і інші елементи додатка. Для цього було створено тему Material UI, яка використовує змінні з Sass:

```
import * as variables from 'common/styles/muiTheme.module.scss';

export const muiTheme = createTheme({
  palette: {
    primary: { main: variables.primary },
    secondary: { main: variables.secondary },
    error: { main: variables.error },
  },
  typography: {
    h1: { fontSize: variables.h1FontSize },
    h2: { fontSize: variables.h2FontSize },
  },
})
```

Основним джерелом правди для стилів у проекті є Sass. Усі стилі, визначені в Sass, імпортуються та використовуються в темі Material UI. Це забезпечило єдине джерело істини для стилів і дозволило легко підтримувати їх консистентність. Наприклад, у файлі muiTheme.module.scss змінні Sass були експортовані як модулі, які потім імпортувалися у JavaScript для налаштування теми Material UI:

```
// common/styles/muiTheme.module.scss
@use 'sass:map';
@import 'common/styles/foundation/variables';

:export {
  primary: $blue-dark;
  secondary: $grey-dark;
  error: $red;
  h1FontSize: $h-1-font-size;
  h2FontSize: $h-2-font-size;
  mobileBreakpoint: map.get($breakpoints, 'sm');
}
```

Компоненти Material UI використовують стилі та змінні, визначені у Sass, що дозволяє зберігати єдиний стиль у всьому додатку. Наприклад, компоненти кнопок, таблиць та інших елементів інтерфейсу були налаштовані відповідно до стилів, визначених у Sass:

```
components: {
  MuiButton: {
    styleOverrides: {
      root: { borderRadius: variables.buttonRadius },
    },
  },
  MuiTableHead: {
    styleOverrides: {
      root: { background: variables.tableHeaderBg },
    },
  },
}
```

Це забезпечило, щоб всі компоненти мали однаковий вигляд і відповідали вимогам дизайну.

У проекті було реалізовано інтеграцію стилів за допомогою Sass та Material UI. Sass виступає основним джерелом правди для стилів, забезпечуючи централізоване керування змінними та стилями. Material UI дозволяє швидко створювати компоненти, використовуючи ці стилі, що забезпечує консистентність і уніфікованість інтерфейсу користувача. Це рішення підвищило ефективність розробки і дозволило легко підтримувати та змінювати стилі у всьому проекті.

4.6 Огляд реалізації клієнстької частини за допомогою екоситсеми React

В рамках даної роботи було реалізовано клієнтську частину додатка за допомогою різноманітних бібліотек екосистеми React. Це забезпечило гнучкість, масштабованість і зручність використання додатка. Основними використаними бібліотеками стали react-i18next для інтернаціоналізації, @reduxjs/toolkit для управління станом, react-router-dom для маршрутизації, notistack для показу повідомлень та react-final-form для валідації форм.

Для забезпечення інтернаціоналізації було використано бібліотеку react-i18next. Вона дозволила підтримувати декілька мов у додатку, автоматично визначаючи мову користувача та завантажуючи відповідні переклади. Інтернаціоналізація була налаштована таким чином, що ресурси для української та англійської мов зберігалися в окремих файлах, а бібліотека автоматично переключалася між ними в залежності від мови користувача. Це дозволило значно покращити користувацький досвід, забезпечивши можливість роботи з додатком на зручній для користувача мові. Приклад ініціалізації react-i18next:

```
i18n
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    resources: {
      en: { translation: translationEN },
      ua: { translation: translationUA },
    },
    lng: 'ua',
    fallbackLng: 'ua',
    interpolation: { escapeValue: false },
  })
```

Для управління станом додатка використовувалася бібліотека @reduxjs/toolkit. Це забезпечило централізоване зберігання стану додатка та ефективне керування даними. Було створено слайси для різних частин додатка, таких як команди та програми, що дозволило легко розширювати та підтримувати додаток. Редюсери та дії були налаштовані для оновлення стану на основі подій користувача. Завдяки використанню @reduxjs/toolkit, код став більш організованим

і легким для підтримки, що дозволило швидко реагувати на зміни вимог і забезпечити стабільну роботу додатка. Фрагмент слайсу команд:

```
const teamsSlice = createSlice({
  name: 'teams',
  initialState: { teams: {}, chosenTeam: null },
  reducers: {
    updateTeamsWithPlayers: (state, action) => {
      state.teams = action.payload
    },
    chooseTeam: (state, action) => {
      state.chosenTeam = action.payload
    },
  },
})
```

Ще однією важливою частиною реалізації було використання @reduxjs/toolkit/query для роботи з API. Це дозволило легко інтегрувати запити до сервера та автоматично оновлювати стан додатка на основі отриманих даних. Було створено базову конфігурацію API та налаштовано різні ендпоінти для отримання та оновлення даних. Це значно спростило роботу з серверними даними, забезпечивши ефективне і безпечне оброблення запитів та відповідей. Використання @reduxjs/toolkit/query дозволило зменшити обсяг ручного коду і зосередитися на логіці додатка, забезпечивши швидку і легку інтеграцію нових API.

```
const mainApi = createApi({
  reducerPath: 'api',
  baseQuery: fetchBaseQuery({ baseUrl: '/api' }),
  endpoints: builder => ({
    getTeams: builder.query({
      query: () => 'teams',
    }),
    getTeamById: builder.query({
      query: id => `teams/${id}`,
    }),
    createTeam: builder.mutation({
      query: team => ({
        url: 'teams',
        method: 'POST',
        body: team,
      }),
    }),
  }),
})
```

Маршрутизація в додатку була реалізована за допомогою бібліотеки react-router-dom. Це дозволило створити багатосторінковий додаток з чітко визначеними маршрутами для кожної сторінки та компонента. Було налаштовано маршрути для загальнодоступних та приватних сторінок, а також створено компоненти для керування доступом до них. Було створено маршрути для таких сторінок, як "Головна", "Про нас", "Контакти", "Програми", а також спеціальні маршрути для тренерів та опікунів. Це забезпечило гнучкість і масштабованість додатка, дозволивши легко додавати нові сторінки та функціонал у майбутньому. Скорочений код роутеру:

```
const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path='/' element={<App />}>
      <Route path='/login' element={<PublicRoute element={<Login />} />} />
      <Route path='/cabinet' element={<PrivateRoute element={<Cabinet />}
/>} />
      <Route
        path='/programs'
        element={<PrivateCoachRoute element={<Programs />} />}
      />
      <Route
        path='/guardian/programs'
        element={<PrivateGuardianRoute element={<GuardianPrograms />} />}
      />
      <Route index element={<NavigateToDefault />} />
      <Route path='*' element={<Navigate replace to='/not-found' />} />
    </Route>
  )
)
```

Для показу повідомлень у додатку використовувалася бібліотека notistack. Вона забезпечила зручний спосіб інформування користувачів про різні події, такі як успішні дії, помилки або інші важливі події. Було налаштовано провайдер SnackbarProvider для централізованого керування повідомленнями, а також створено кастомний компонент для показу повідомлень. Це дозволило забезпечити єдиний стиль і поведінку всіх повідомлень у додатку, підвищивши користувацький досвід і надавши користувачам важливу інформацію в зручний і ненав'язливий спосіб.

```
const NotistackProvider = ({ children }) => (
  <SnackbarProvider
    anchorOrigin={{ horizontal: 'center', vertical: 'bottom' }}
    action={key => (
      <IconButton onClick={() => closeSnackbar(key)}>
        <CloseIcon />
      </IconButton>
    )}
  >
    {children}
  </SnackbarProvider>
)
```

Валідація форм у додатку була реалізована за допомогою react-final-form. Це дозволило створювати та валідувати форми з використанням кастомних валідаторів. Було розроблено функції валідації для перевірки обов'язкових полів, дат, чисел та інших даних, що забезпечило надійну перевірку введених користувачами даних. Це забезпечило точність і цілісність даних, що вводяться користувачами, зменшило кількість помилок і покращило загальний досвід роботи з додатком. Використання react-final-form дозволило легко адаптувати і розширювати валідаційні правила в залежності від змін вимог до бізнес-логіки додатка.

```
const useValidationHelpers = () => {
  const { t } = useTranslation()

  const required = useCallback(
    value => (value ? undefined : t('error_messages.required')),
    [t]
  )

  const validateDate = useCallback(
    value =>
      dayjs(value).isValid() ? undefined :
t('error_messages.invalid_date'),
    [t]
  )

  return { required, validateDate }
}
```

Для забезпечення інтеграції всіх цих бібліотек було створено компонент OuterProviders, який обгортає додаток необхідними провайдерами. Це дозволило

централізувати налаштування і забезпечити доступ до всіх функцій у будь-якому місці додатка. Компонент OuterProviders забезпечив єдиний контекст для всієї програми, включаючи стан, теми, інтернаціоналізацію, а також доступ до повідомлень і маршрутизації. Це значно спростило структуру додатка і зробило код більш зрозумілим і підтримуваним.

```
const OuterProviders = ({ store = appStore, children }) => {
  const locale = i18n.language
  dayjs.locale(locale === 'ua' ? 'uk' : 'en')

  return (
    <Provider store={store}>
      <ThemeProvider theme={muiTheme}>
        <NotistackProvider>
          <I18nextProvider i18n={i18n}>
            <PassageProvider
              appId={process.env.PASSAGE_APP_ID!}
              lang={i18n.language}
            >
              <ActionProvider>
                <LocalizationProvider
                  dateAdapter={AdapterDayjs}
                  adapterLocale={locale}
                >
                  {children}
                </LocalizationProvider>
              </ActionProvider>
            </PassageProvider>
          </I18nextProvider>
        </NotistackProvider>
      </ThemeProvider>
    </Provider>
  )
}
```

Інтеграція та використання бібліотек react-i18next, @reduxjs/toolkit, @reduxjs/toolkit/query, react-router-dom, notistack, react-final-form та інших допомогли створити гнучкий і масштабований додаток з багатофункціональним інтерфейсом користувача. Це рішення підвищило ефективність розробки, забезпечило зручний користувацький досвід і дозволило легко адаптувати додаток до змін вимог та масштабування в майбутньому.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Проведемо мануальне, або ручне, тестування функціональності програмної системи з адміністрування футбольних секцій.

Проведемо тестування функціональності «Реєстрація користувачів в системі». Кроки тесту: відкрити застосунок в анонімному режимі, обрати опцію «Зареєструватися», заповнити текстові дані профілю, обрати роль (тренер, опікун, менеджер секцій), обрати опцію «Зареєструватися», ввести надісланий системою Passage код підтвердження, який приходить на електронну адресу, перейти в авторизований режим користування застосунком. Очікуваний результат: користувач успішно зареєстрований та переправлений на головну сторінку застосунку в авторизованому режимі (замість опції «Увійти» в меню відображається опція «Вийти»). Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Редагування профілю». Кроки тесту: відкрити застосунок в авторизованому режимі, перейти на сторінку редагування профілю, змінити фамілію, завантажити картинку профілю, зберегти. Очікуваний результат: зміни профілю успішно збережені, по перевантаженню сторінки відображаються оновлені дані профілю, на сторінці редагування профілю заборонено редагування електронної адреси, ролі та імені користувача (юзернейму). Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Завантаження сертифікату Української асоціації футболу (УАФ) [8] тренером на сторінці профілю» (що по суті є різновидом кейсу з оновлення профілю). Кроки тесту: відкрити застосунок в авторизованому режимі, перейти на сторінку редагування профілю, завантажити приклад сертифікату УАФ (файл з розширенням PDF). Очікуваний результат: сертифікат УАФ успішно збережений (відображається по перевантаженню

сторінки) та доступний до завантаження. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Вихід і вхід». Кроки тесту: відкрити застосунок в авторизованому режимі, обрати опцію меню «Вийти», обрати опцію «Увійти», ввести електронну адресу, ввести надісланий Passage код підтвердження. Очікуваний результат: після виходу з системи користувача переводить на головну сторінку в анонімному режимі (в меню відображається опції «Увійти» та «Реєстрація»), а після входу користувач повертається в авторизований режим (в меню відображається опція «Вийти»). При тестуванні даного кейсу було знайдено та виправлено баг відображення опцій «Реєстрація» та «Увійти» (в анонімному режимі не відображалася опція «Реєстрація», див. рис.5.1).

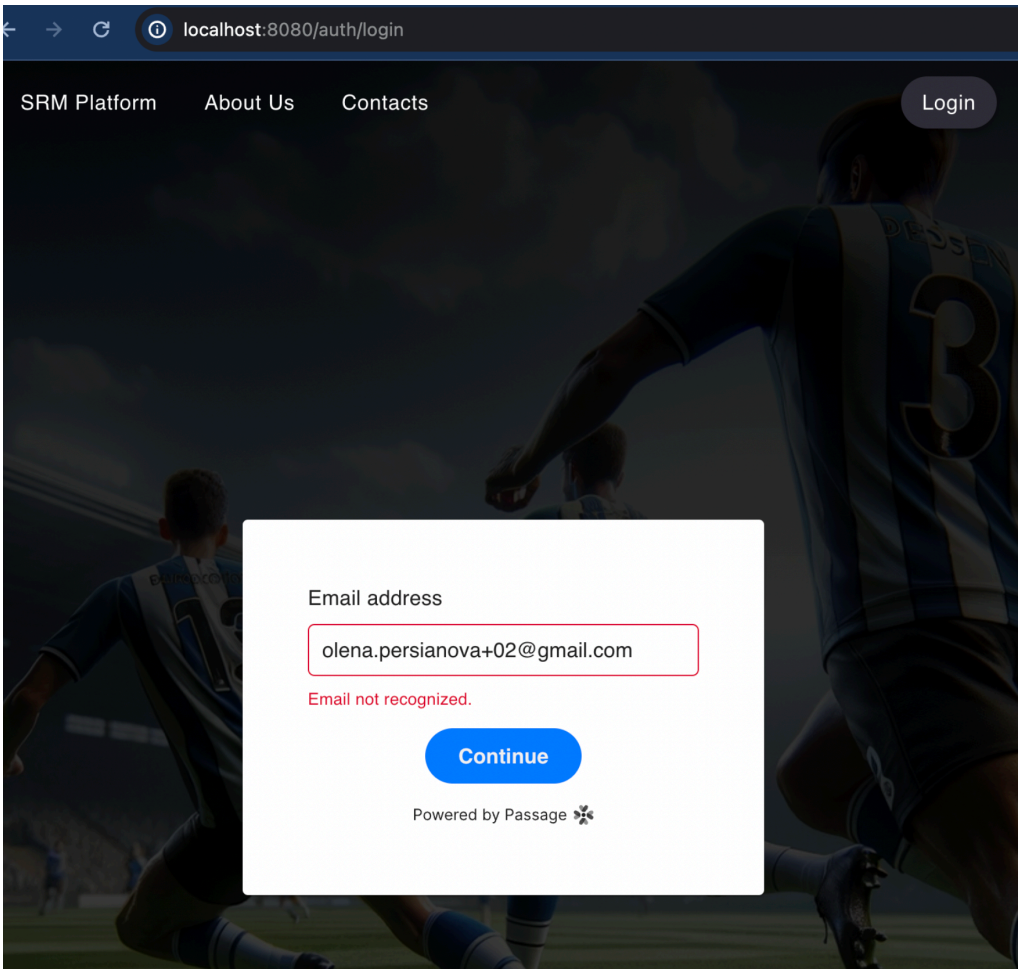


Рисунок 5.1 – Баг анонімного режиму: відсутність опції «Реєстрація» в меню (рисунок виконаний самостійно)

Також, було знайдено баг з відображення опції «Увійти» та «Вийти»: як для анонімного, так і для авторизованого користувачів при переході на головну сторінку відображалася опція «Увійти», хоча для авторизованого користувача має показуватися опція «Вийти». Після виправлення дефектів кроки тесту було повторено. В результаті, всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Зміна мови інтерфейсу». Кроки тесту: перейти в інкогніто режим браузеру, зайти в систему, змінити мову на англійську, а потім - змінити мову знову на українську. Очікуваний результат: за замовченням, зміст відображається українською мовою (в інкогніто режимі браузеру веб-сайт показує україномовний контент); зміна мови доступна в авторизованому режимі, і при переході на англійську мову мова всього контенту змінюється відповідно; повернення до української мови змінює текстовий контент веб-сайту з англійської на українську. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Створення, редагування та перегляд футбольної секції менеджером секцій». Кроки тесту: зайти в систему як менеджер секцій, перейти на сторінку переліку футбольних секцій, ввести необхідну інформацію про секцію, зберегти, відредагувати назву, зберегти, вийти з системи, зайти в систему як інший менеджер секцій та перейти на сторінку футбольних секцій. Очікуваний результат: футбольна секція успішно збережена та оновлена, автор-менеджер бачить секцію у переліку секцій, але інший менеджер не бачить дану секцію у переліку секцій. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Створення, публікація, редагування та перегляд програми тренером». Кроки тесту: зайти в систему як тренер, перейти на сторінку програм, обрати опцію створення програми, ввести необхідну інформацію, створити програму, редагувати програму (змінити назву),

вийти з системи, увійти в систему як інший тренер та перейти на сторінку програм. Очікуваний результат: програма успішно збережена та оновлена, автор-тренер бачить програму у переліку секцій, але інший тренер не бачить дану програму у переліку секцій. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Перегляд власних програм тренером». Кроки тесту: зайти в систему як тренер, перейти на сторінку програм, обрати опції сортування, фільтрації та пошуку програм програми, роздрукувати перелік. Очікуваний результат: тренер бачить всі свої програми та має змогу сортувати, фільтрувати, здійснювати пошук та друкувати звіт з переліком власних програм. У процесі тестування було виявлено баг у функціональності друку звіту з власних програм: звіт не фільтрував програми, якщо обрано ознаку фільтрування за ознакою опублікованості. Після виправлення багу всі кроки тесту були виконані наново, і врешті-решт було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Призначення футбольної секції програмі менеджером секцій». Кроки тесту: зайти в систему як менеджер секцій, перейти на сторінку секцій, призначити програму певній секції. Очікуваний результат: програма була успішно призначена на футбольну секцію, тренером програми отримано сповіщення на електронну адресу про призначення футбольної секції. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат (рис.5.2), отже, тест пройдено.

Проведемо тестування функціональності «Призначення опікуна гравцю в системі самим опікуном». Кроки тесту: зайти в систему як опікун, перейти на сторінку власних гравців, створити профіль гравця, вийти з системи, зайти в систему як інший опікун. Очікуваний результат: профіль гравця успішно створений та опікуном гравця призначений авторизований користувач-опікун, профіль гравця відображається в переліку гравців опікуна, однак не відображається в переліку

гравців іншого опікуна. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

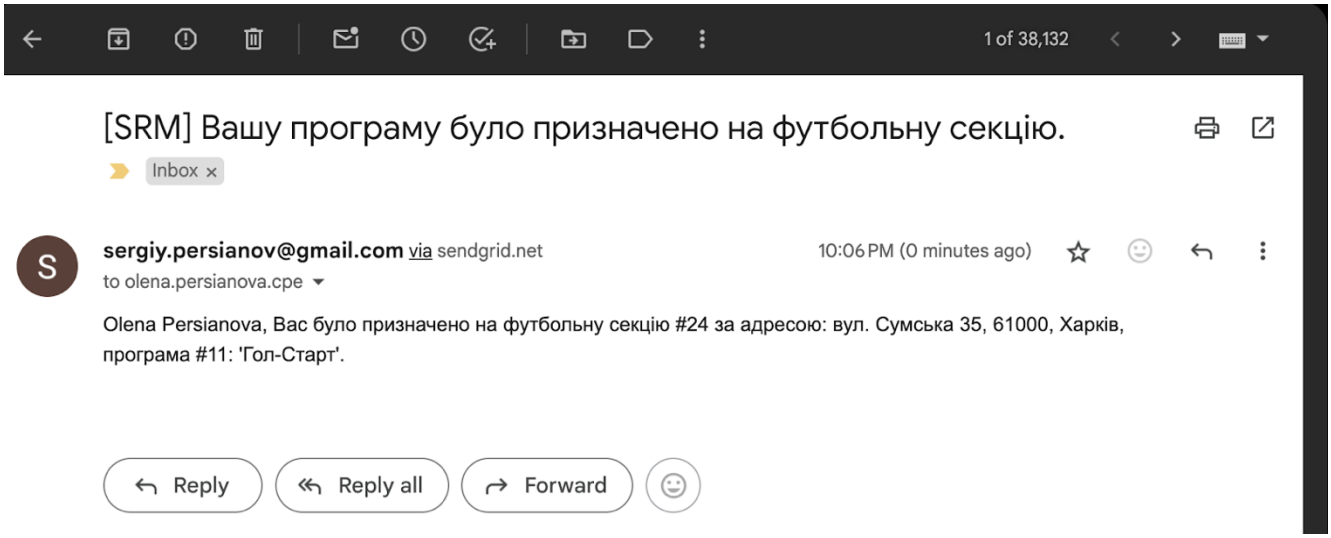


Рисунок 5.2 – Вигляд повідомлення на електронну адресу тренера про призначення футбольної секції програмі (рисунок виконаний самостійно)

Проведемо тестування функціональності «Реєстрація гравця на програму опікуном» та «Перегляд реєстрацій на власну програму тренером». Кроки тесту: зайти в систему як опікун, перейти на сторінку програм, зареєструвати свого гравця на програму, вийти з системи, зайти в систему як тренер програми, на яку щойно зареєструвалися, та перейти на сторінку перегляду реєстрацій на програму. Очікуваний результат: гравець успішно зареєстрований на програму і тренер бачить дану реєстрацію. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Створення та редагування команди тренером». Кроки тесту: зайти в систему як тренер, перейти на сторінку переліку команд, обрати опцію створення команди, ввести необхідну інформацію, зберегти, змінити назву команди, зберегти. Очікуваний результат: команда успішно створена та відредагована . Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Створення та редагування розкладу занять тренером». Кроки тесту: увійти в систему як тренер, перейти на сторінку програми та її розкладу, створити кілька занять, змінити час заняття. Очікуваний результат: заняття програми успішно створені та заняття успішно відредаговано, опікуном отримано сповіщення на електронну адресу про зміну розкладу. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат (рис.5.3), отже, тест пройдено.

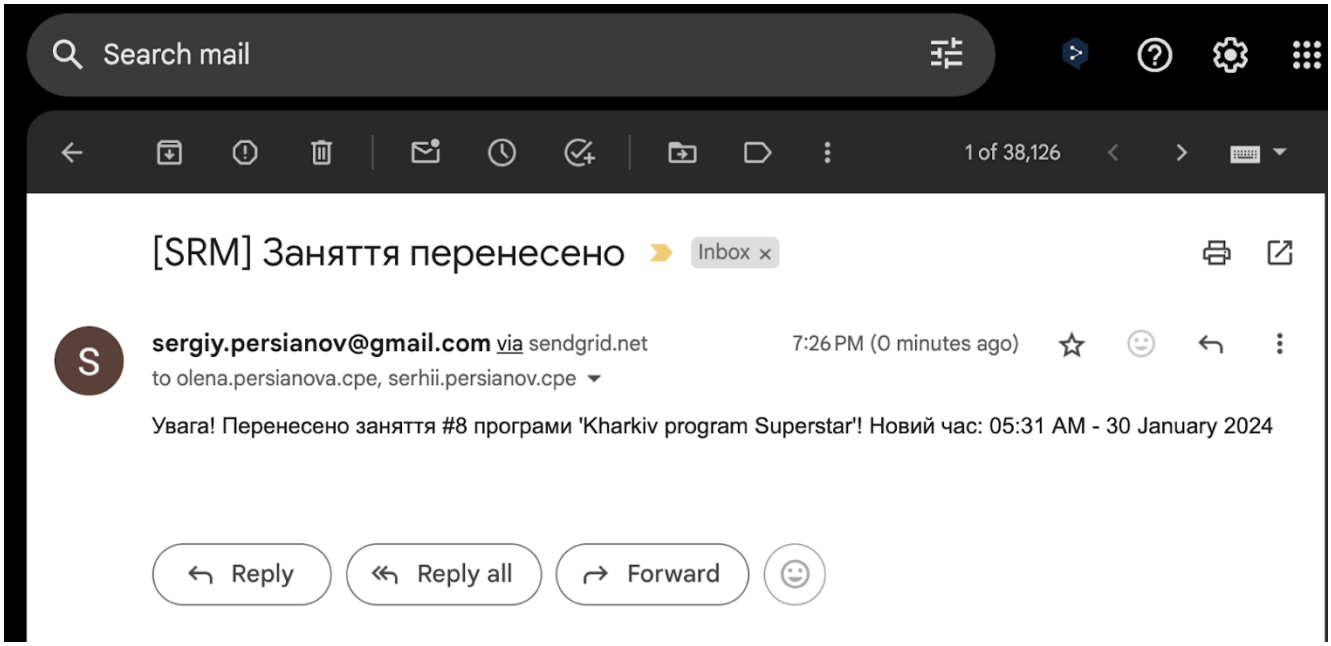


Рисунок 5.3 – Вигляд повідомлення на електронну адресу про зміну розкладу занять (рисунок виконаний самостійно)

Проведемо тестування функціональності «Перегляд розкладу занять гравця їхнім опікуном». Кроки тесту: зайти в систему як опікун, перейти на сторінку програми та її розкладу. Очікуваний результат: опікун може бачити розклад програми, на якій займаються їхні гравці. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування функціональності «Перегляд команд, що займаються у власній секції менеджером секцій». Кроки тесту: зайти в систему як менеджер секцій, перейти на сторінку секцій та їхніх команд. Очікуваний результат: опікун в

змозі бачити команди, що займаються в їхніх секціях, а також здійснювати ссортування, фільтрацію за ознакою наявності вільних місць, а також друк звіту з команд, що займаються у секції менеджера. Всі необхідні кроки були виконано успішно та було отримано очікуваний результат, отже, тест пройдено.

Проведемо тестування інтерфейсу на предмет відповідності адаптивного дизайну. Адаптивний дизайн є критично важливим аспектом сучасних веб-застосунків, оскільки користувачі використовують різноманітні пристрої для доступу до інтернет-ресурсів. Тестування було проведене на різних пристроях та розмірах екранів, щоб забезпечити коректне відображення та функціонування інтерфейсу на всіх можливих платформах.

Для реалізації адаптивного дизайну використовувалася комбінація SASS міксинів і бібліотеки Material UI. Це дозволило гнучко налаштувати стилі для різних розмірів екранів, забезпечуючи оптимальний вигляд інтерфейсу на всіх пристроях.

Наприклад, форма реєстрації (рис. 5.4), була ретельно перевірена на різних мобільних пристроях, включаючи смартфони з розмірами екранів до 30rem (480px). Використання міксину @mixin mobile дозволило адаптувати всі необхідні поля форми так, щоб вони залишалися доступними та легко читаються користувачем. Елементи керування, такі як кнопки та текстові поля, були розташовані таким чином, щоб користувач міг без проблем взаємодіяти з ними, використовуючи сенсорний екран мобільного пристрою.

Сторінка “Перелік програм” (рис. 5.5), також пройшла тестування на предмет адаптивності. Було перевірено, чи всі програми відображаються в зрозумілому та компактному вигляді, забезпечуючи зручний доступ до інформації. Користувачі можуть переглядати, сортувати та фільтрувати програми, не втрачаючи при цьому функціональність інтерфейсу. Використання міксину @mixin tablet дозволило налаштувати відображення сторінки для екранів розміром до 49rem (768px).

Важливим аспектом було зберегти читабельність тексту та зручність навігації навіть на невеликих екранах.

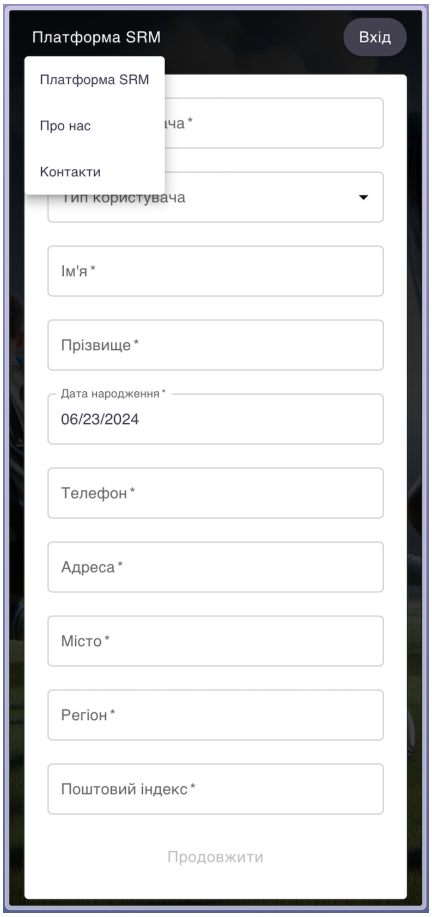


Рисунок 5.4 – Вигляд форми реєстрації на мобільному пристрої (рисунок виконаний самостійно)

Деталі програми на мобільному пристрої, як показано на рисунку 5.6, також були ретельно протестовані. Всі основні дані про програму, такі як розклад занять, опис та інші важливі деталі, відображаються чітко і структуровано. Це дозволяє користувачам швидко знаходити необхідну інформацію та здійснювати взаємодію з програмою навіть на невеликих екранах. Використання міксинів @mixin laptop та @mixin desktop забезпечило оптимальне відображення для екранів розміром до 64rem (1024px) та 96rem (1536px) відповідно.

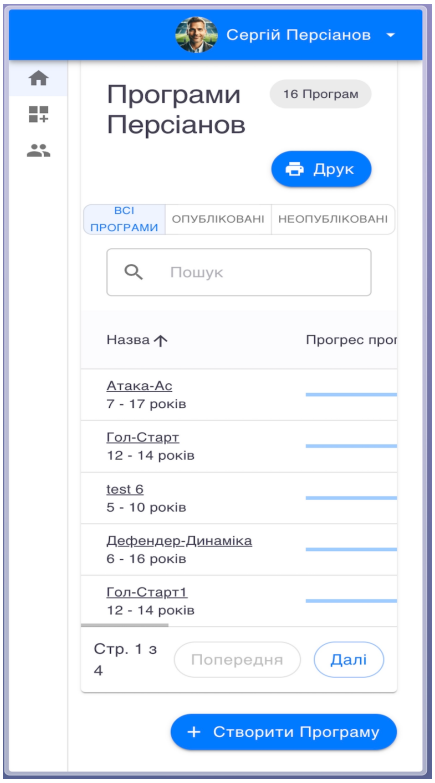


Рисунок 5.5 – Вигляд сторінки “Перелік програм” на мобільному пристрої (рисунок виконаний самостійно)

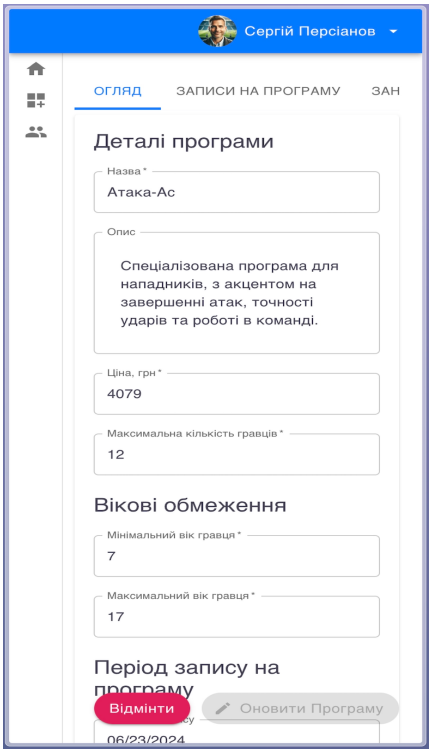


Рисунок 5.6 – Вигляд сторінки “Деталі програми” про на мобільному пристрої (рисунок виконаний самостійно)

Для забезпечення максимальної адаптивності, були використані сучасні методи веб-розробки, такі як медіа-запити в CSS, що дозволяють змінювати стилі в залежності від розміру екрану. Крім того, була застосована технологія гнучкої верстки (flexbox), яка дозволяє створювати динамічні макети, що автоматично адаптуються до будь-якого розміру екрану. Завдяки цьому, інтерфейс залишається функціональним і зручним незалежно від того, чи використовується десктопний комп'ютер, планшет або смартфон.

Десктопні версії інтерфейсу також були піддані тестуванню. На великих екранах інтерфейс забезпечує максимально зручну навігацію та використання всіх доступних функцій. Розділи, такі як “Перелік програм” і “Деталі програми”, відображають більше інформації одночасно, що дозволяє користувачам швидше знаходити потрібні дані.

Особлива увага приділялася тестуванню планшетних версій інтерфейсу. Планшети часто використовуються для роботи та розваг завдяки їхньому зручному розміру екрана. Тому було важливо забезпечити, щоб інтерфейс був однаково зручним і на планшетах з роздільною здатністю до 49rem (768px). Під час тестування перевірялися всі основні функції, такі як навігація, заповнення форм та перегляд списків, щоб переконатися, що вони працюють бездоганно і не викликають у користувачів жодних труднощів.

Крім того, тестування включало перевірку роботи інтерфейсу в різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge (табл.5.1). Це дозволило переконатися, що інтерфейс однаково добре працює незалежно від обраного користувачем браузера.

Було перевірено коректність відображення всіх елементів, роботу інтерактивних функцій та швидкість завантаження сторінок. Завдяки цьому користувачі можуть використовувати програму на будь-якому браузері, не турбуючись про можливі технічні проблеми. Варто відмітити, що програмна система працює найкраще в найсучасніших версіях браузерів.

Таблиця 5.1 – Перелік протестованих браузерів (таблиця виконана самостійно)

Браузер	Операційна система	Найстаріша протестована версія	Найновіша протестована версія
Safari	macOS	14.1.2	17.5
Safari	iOS		17.5
Edge	Windows	91.0.864.59	126.0.2592.81
Edge	macOS	91.0.864.59	126.0.2592.81
Edge	Linux	91.0.864.59	126.0.2592.81
Edge	iOS		126.2592.86
Edge	Android		125.0.2535.96
Chrome	Windows	91.0.4472.124	126.0.6478.126
Chrome	macOS	91.0.4472.124	126.0.6478.126
Chrome	Linux	91.0.4472.124	126.0.6478.126
Chrome	Android		126.0.6478.122
Chrome	iOS		126.0.6478.108
Firefox	Desktop	89.0.2	127.0.2
Firefox	iOS		127.1
Firefox	Android		127.0

Результати тестування підтвердили, що інтерфейс програми відповідає вимогам адаптивного дизайну і забезпечує коректне відображення та функціонування на різних пристроях та розмірах екранів. Це забезпечує користувачам зручний доступ до всіх функцій програми незалежно від обраного ними пристрою, що значно підвищує загальний рівень задоволеності від використання програмного забезпечення.

Таким чином, було проведено функціональне тестування застосунку з увагою на правила авторизації за ролями та тестування інтерфейсу, успішність якого підтверджує відповідність програми стандартам якості програмного забезпечення.

ВИСНОВКИ

Результатом роботи став став проєкт клієнтської частини інтегрованої системи для управління відносинами у спортивній галузі. Застосунок надає можливість ефективної організації футбольних навчальних програм, а також забезпечує зручні інструменти для пошуку та зв'язку між тренерами та спортсменами. Особлива увага в процесі розробки була приділена оптимізації зберігання та обробки інформації, що забезпечує високу якість та надійність даних.

Проєктування та програмна реалізація цього проекту дозволили автору не тільки закріпити знання, отримані під час навчання, але й розширити компетенції в області сучасних веб-технологій, зокрема в роботі з REST API, використанні екосистеми бібліотеки React, а також в освоєнні архітектури Redux для керування станом застосунку. Це досвід підкреслює важливість адаптивності та готовності до застосування новітніх технологій у вирішенні програмних задач.

Спроєктований та реалізований проєкт являтиме собою повнофункціональну програмну систему, орієнтовану на ефективне управління футбольними тренувальними секціями. Проект буде реалізований на основі глибокого аналізу об’єктно-орієнтованого програмування, що демонструє комплексний підхід до розробки програмного забезпечення у спортивній індустрії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sports fan relationships with soccer teams . International Journal of Digital Culture and Electronic Tourism. 2(3):213. 2018. URL: https://www.researchgate.net/publication/323765878_Sports_fan_relationships_with_soccer_teams (дата звернення 21.05.24)

2. Сервіс пошуку тренерів “Trener” (Україна). URL: https://trener.ua/uk/kiev (дата звернення 23.05.24)

3. Сервіс замовлення послуг “Кабанчик” (Україна). URL: https://kabanchik.ua/ua/kyiv/category/posluhy-treneriv (дата звернення 04.07.24)

4. Сервіс пошуку репетиторів “BUKI” (Україна). URL: https://buki.com.ua/tutors/fitnes-trener/ (дата звернення 20.06.24)

5. Сервіс пошуку та доставки товарів і послуг “OLX” (Україна). URL: https://www.olx.ua/uk/uslugi/obrazovanie/ (дата звернення 30.05.24)

6. Сервіс пошуку навчальних закладів “Edusearch” (Україна). URL: https://edusearch.com.ua/ (дата звернення 01.07.24)

7. Центр ліцензування української футбольної асоціації. URL: https://clffu.org.ua/ (дата звернення 02.06.24)

8. Українська асоціація футболу. URL: https://uaf.ua (дата звернення 04.06.24)

9. Мокапи графічного інтерфейсу користувача ПЗ SRM в Figma. URL: https://www.figma.com/file/nJlzmeYeWSwN8xMYtrOPkK/Diploma (дата звернення 14.06.24)

10. JetBrains IDEs. URL: https://www.jetbrains.com/ (дата звернення 24.05.24)

11. React: The library for web and native user interfaces. URL: https://react.dev/ (дата звернення 22.05.24)

12. OpenAPI Specification: The world's most widely used API description standard. URL: https://www.openapis.org/ (дата звернення 26.05.24)

13. Representational State Transfer (REST). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. URL: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення 17.06.24)

14. Babel: JavaScript compiler. URL: https://babeljs.io/ (дата звернення 18.06.24)

15. Webpack: bundle your assets. URL: https://webpack.js.org/ (дата звернення 19.06.24)

16. Unlock a passwordless future with passkeys: Passage by 1Password. URL: https://passage.1password.com/ (дата звернення 12.06.24)

17. Passage React SDK: Integrate Passage into your React application. URL: https://docs.passage.id/embedded-login/examples-by-framework/react (дата звернення 12.06.24)

18. JSON Web Tokens (JWT). URL: https://jwt.io/ (дата звернення 25.06.24).

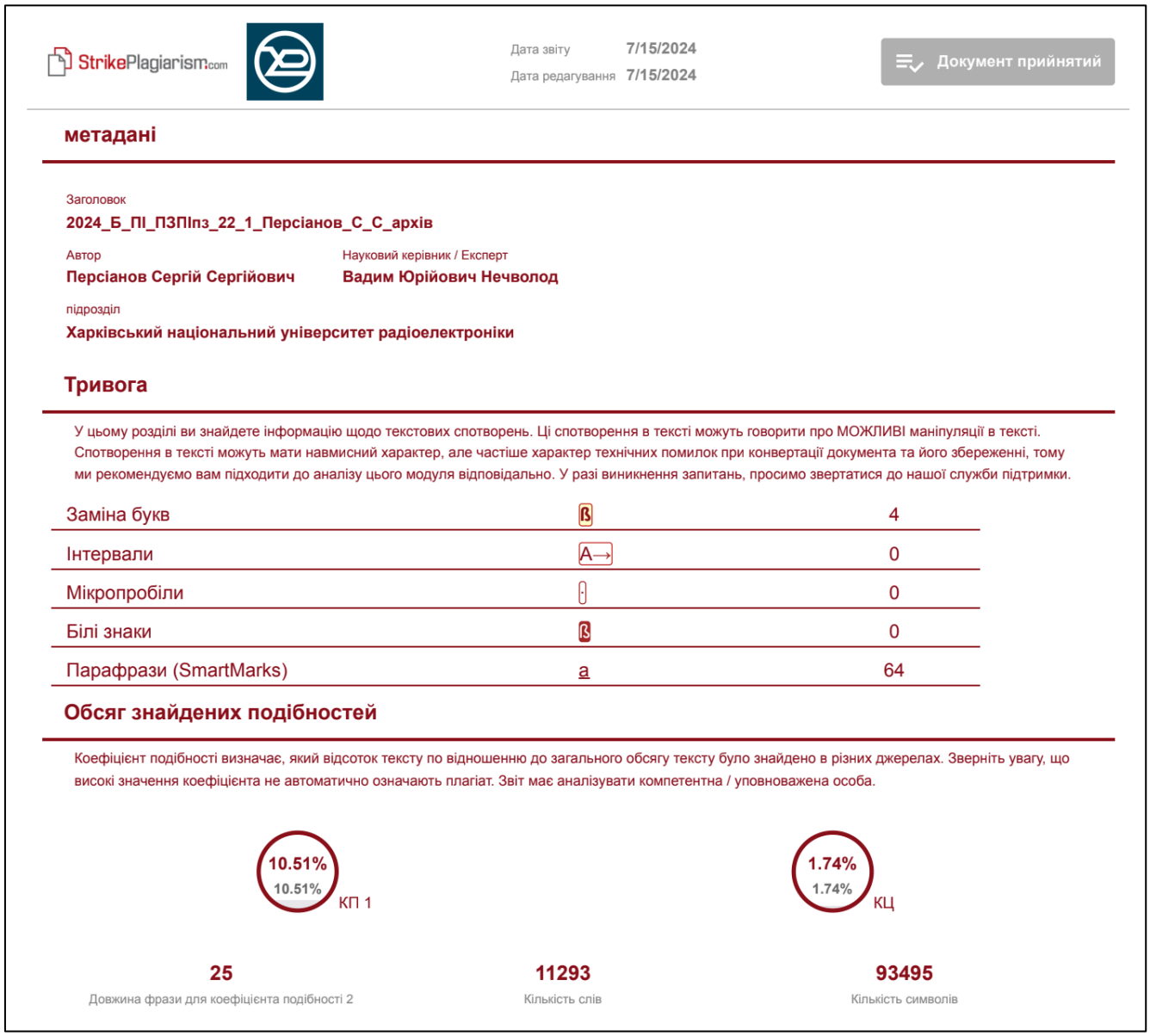
19. i18next is an internationalization-framework. URL: https://www.i18next.com/ (дата звернення 27.06.24)

20. NPM: Build amazing things. URL: https://www.npmjs.com/ (дата звернення 29.06.24)

21. Auth0 by Okta. URL: https://auth0.com/docs/ (дата звернення 03.07.2024)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/15/2024
Дата редагування 7/15/2024

Документ прийнятий

метадані

Заголовок
2024_Б_ПІ_ПЗПІпз_22_1_Персіанов_С_С_архів

Автор
Персіанов Сергій Сергійович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		4
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	64

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

10.51% 10.51% КП 1

1.74% 1.74% КЦ

25
Довжина фрази для коефіцієнта подібності 2

11293
Кількість слів

93495
Кількість символів

ДОДАТОК Б

Приклад звіту

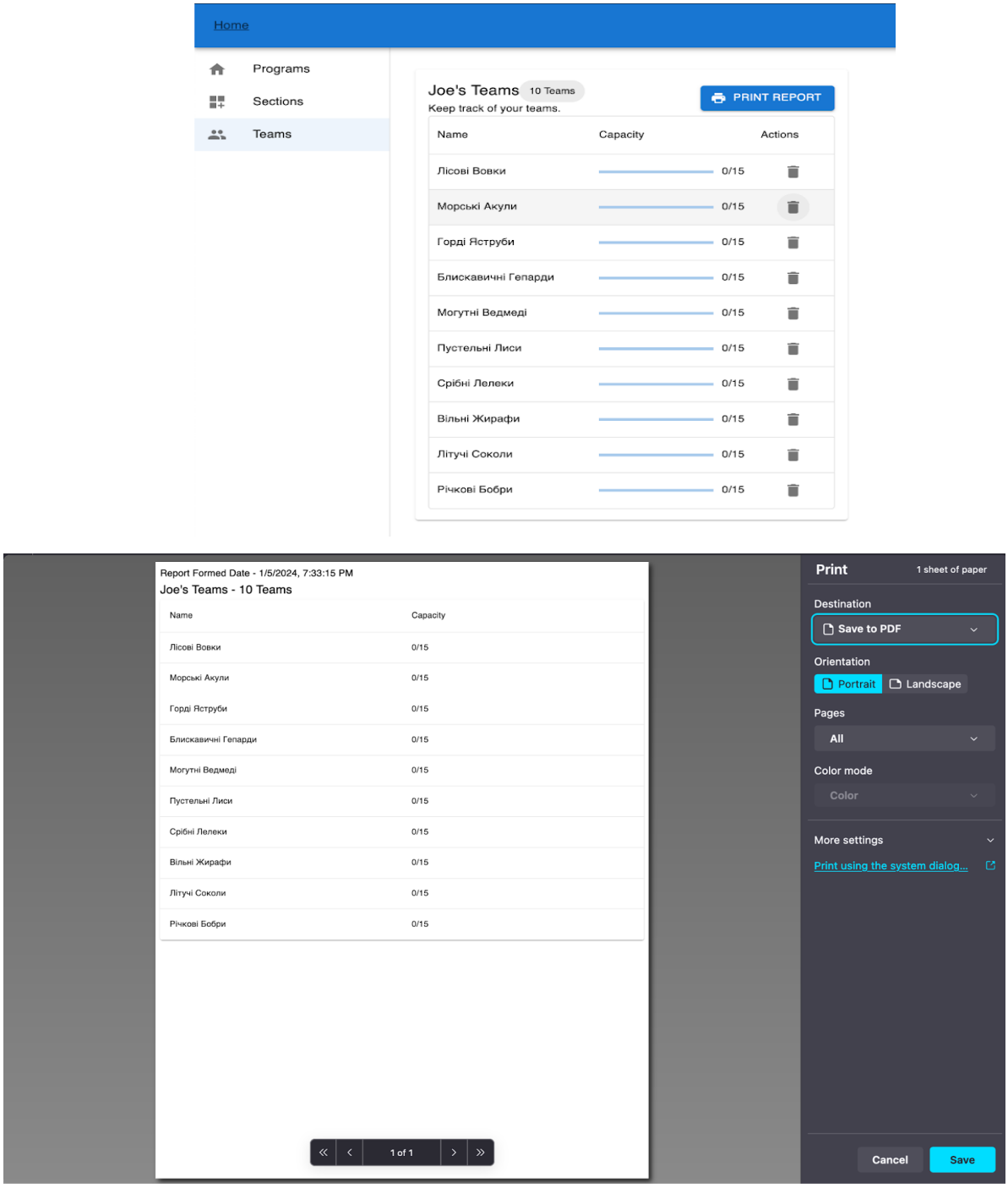


Рисунок А.1 - Перелік команд з даними про їхні назви та прогресом набору в команди (кількість набраних гравців vs максимальна кількість в команді) (рисунок виконаний самостійно)

ДОДАТОК В

Слайди презентації

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Кваліфікаційна робота
Програмна система з адміністрування футбольних секцій.
Front-end

Виконав:
студент групи ПЗПІпз-22-1
Персіанов С.С.

Керівник:
Доц. кафедри ПІ
Валенда Н.А.

2024

Мета роботи

Створення клієнтської частини (англ. front-end) програмної системи з адміністрування футбольних секцій, що підтримуватиме режими тренера, менеджера футбольних секцій, та опікуна.

2

Аналіз існуючих рішень

Характеристика	Конкурент				
	Trener	Кабанчик	Buki	Olx	Edusearch
Сфокусованість на набутті навичок (компетентнісний підхід)	+ (частково)	-	-	-	-
Зручний календар з планування занять	-	-	-	-	-
Загальний рівень складності UI / UX	помірний	високий	високий	помірний	помірний
Підтримка локалізації	+ (2 мови)	+ (2 мови)	+ (3 мови)	+ (2 мови)	-

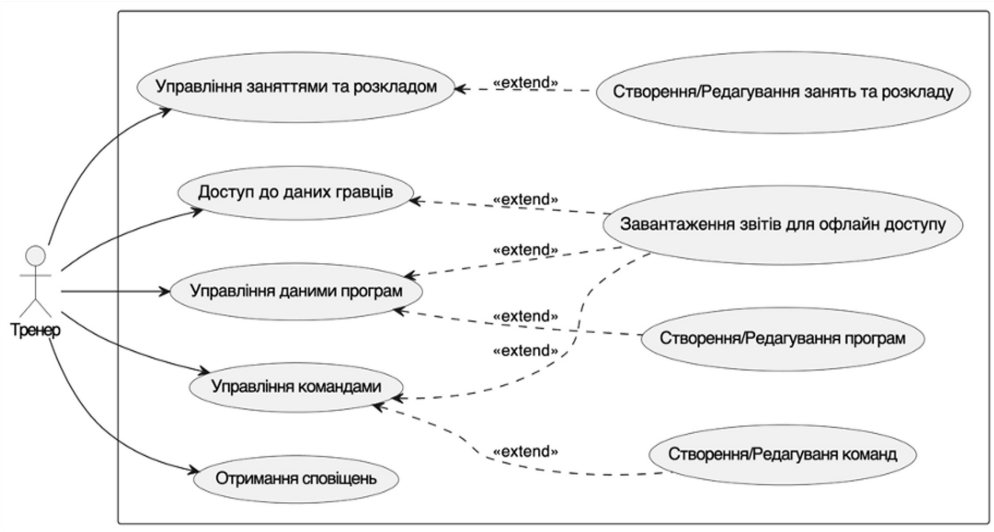
3

Постановка задачі

- Розробити клієнтську частину (реактивний інтерфейс) програмної системи з адміністрування футбольних секцій за допомогою бібліотеки ReactJS та її екосистеми;
- спроєктувати зрозумілий, гнучкий та ефективний користувацький інтерфейс за допомогою інструменту Figma;
- запровадити локалізацію програмної системи англійською та українською мовами за допомогою бібліотеки i18next;
- провести мануальне функціональне тестування та тестування інтерфейсу.

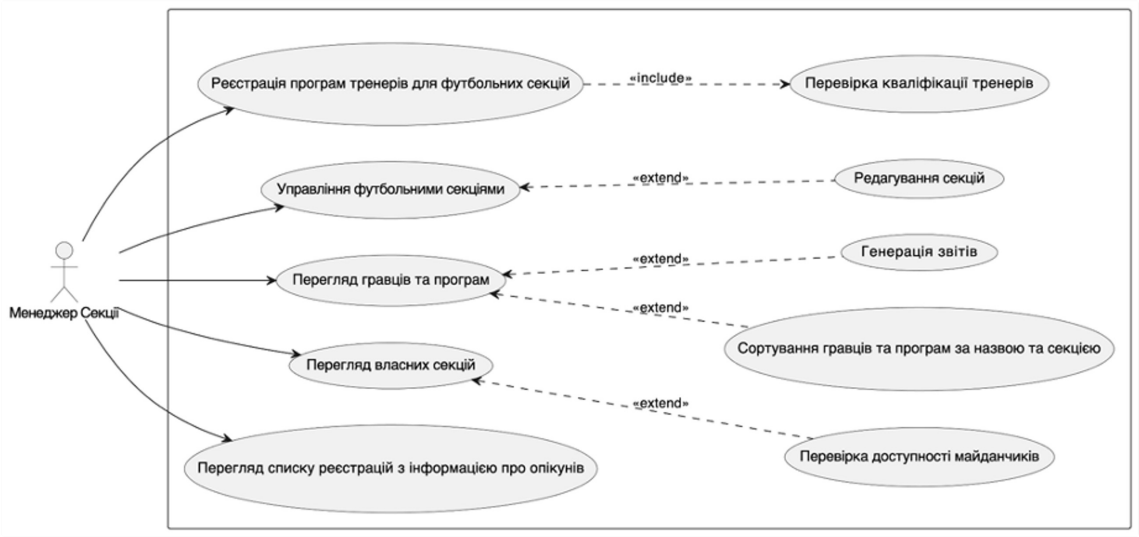
4

USE-CASE діаграма ролі “Тренер”



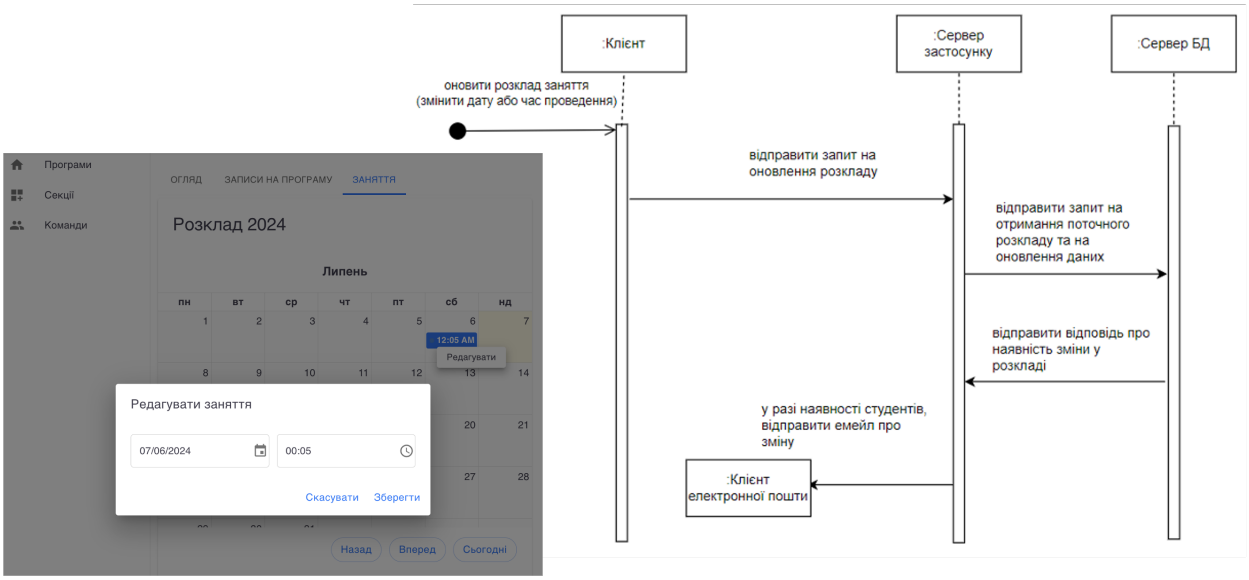
5

USE-CASE діаграма ролі “Менеджер секцій”



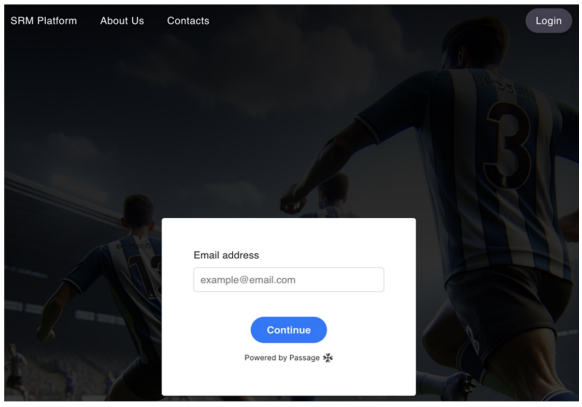
6

Діаграма послідовностей рішення зі сповіщень гравців про зміну розкладу

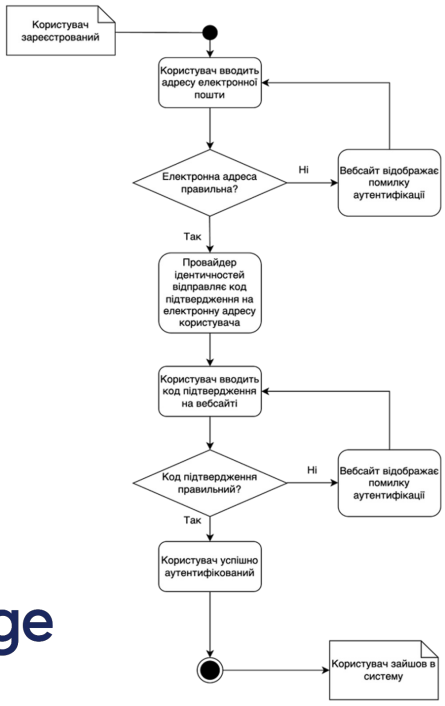


7

Системи аутентифікації: діаграма активностей, скрин сторінки логіну



Passage by 1Password



8

Використані технології

React JS

Redux

RTK Query

TypeScript

react-i18next

webpack

BABEL

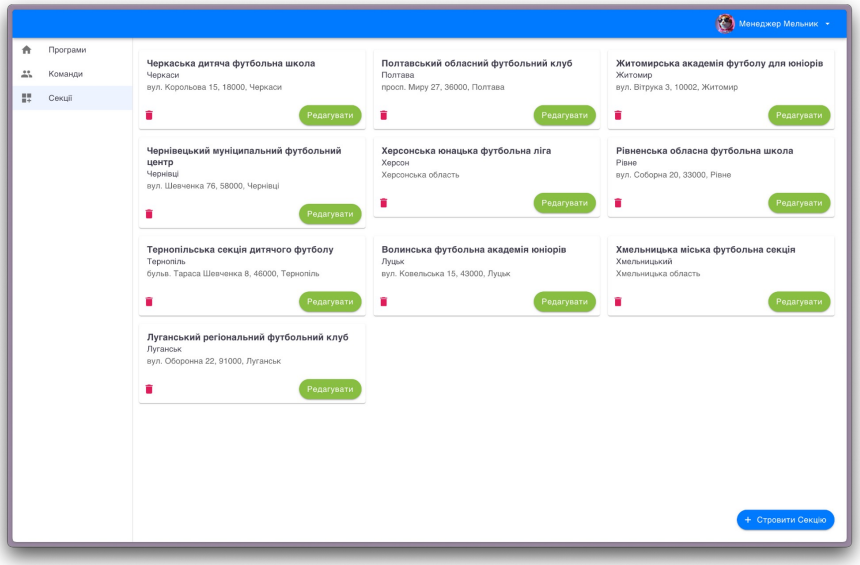
Figma

Passage by 1Password

Sass

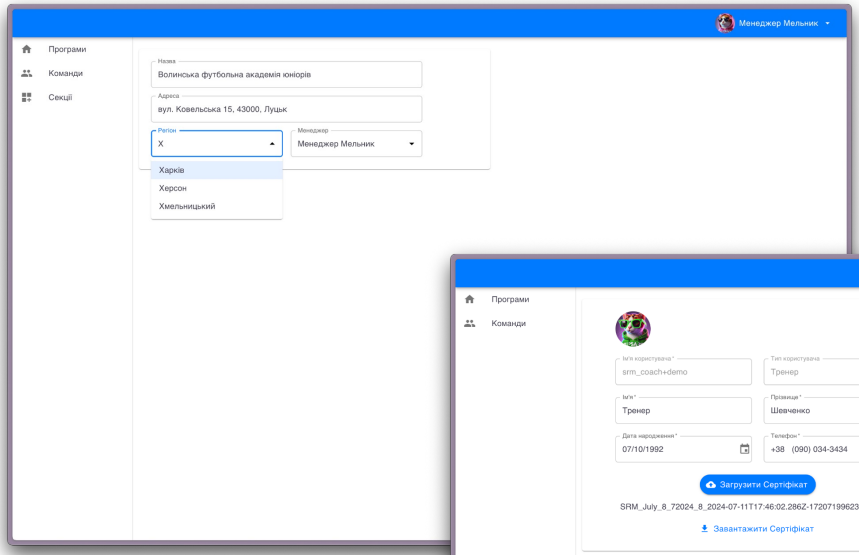
9

Скріншоти інтерфейсу - Перелік секцій

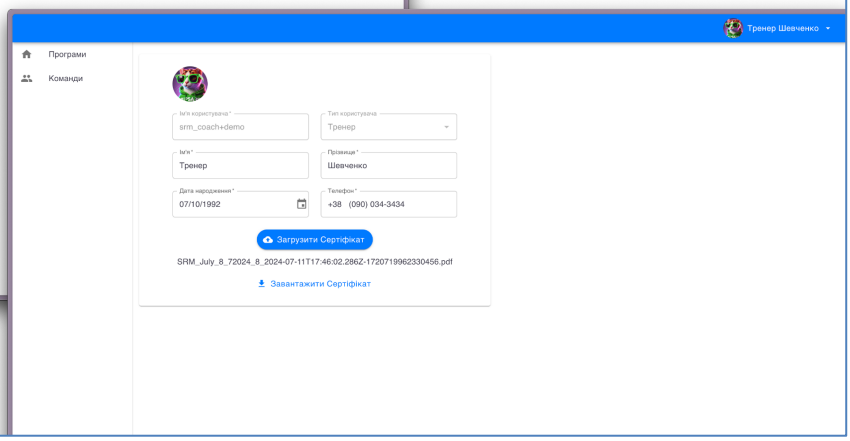


10

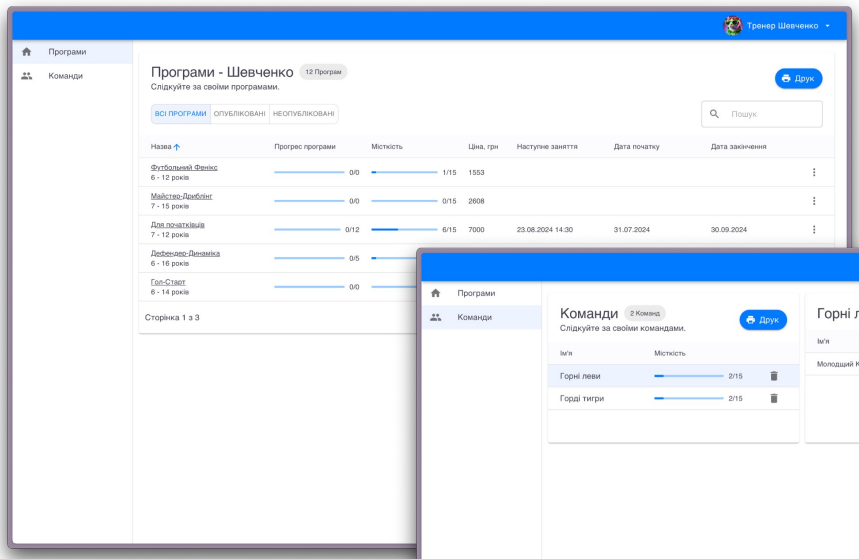
Створення нової секції для тренерів



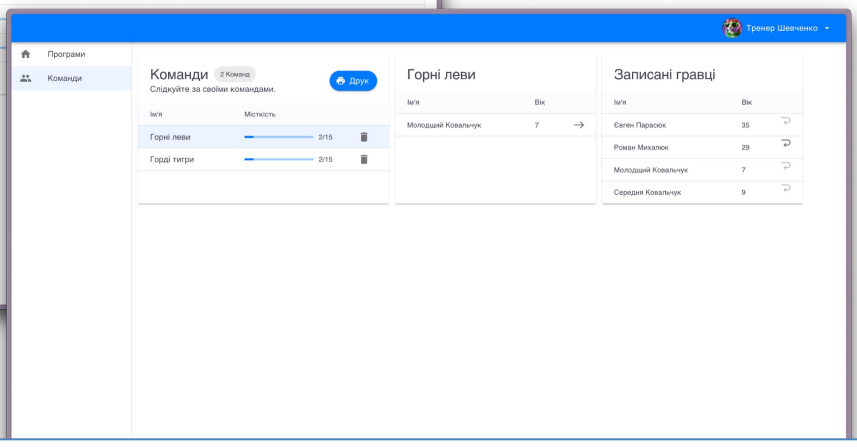
Кабінет тренера.



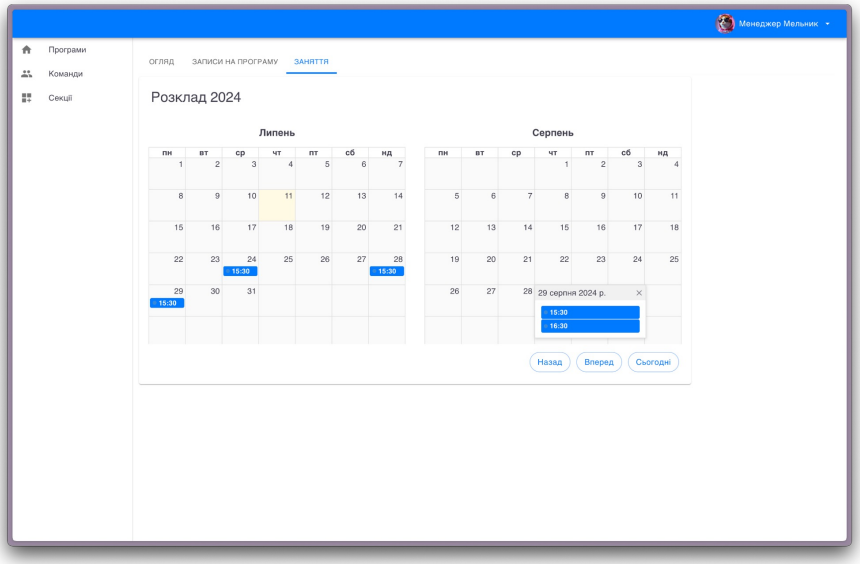
Список програм



Управління командами

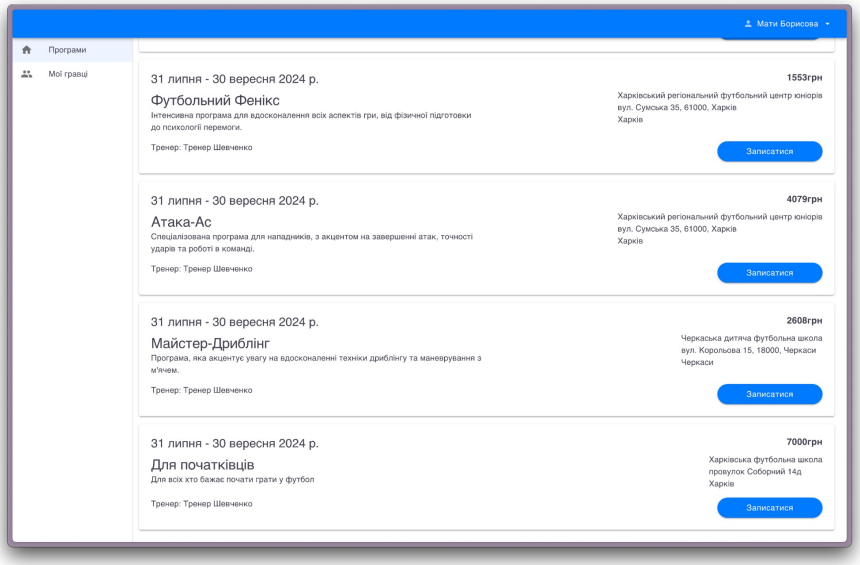


Деталі програми - Розклад занять



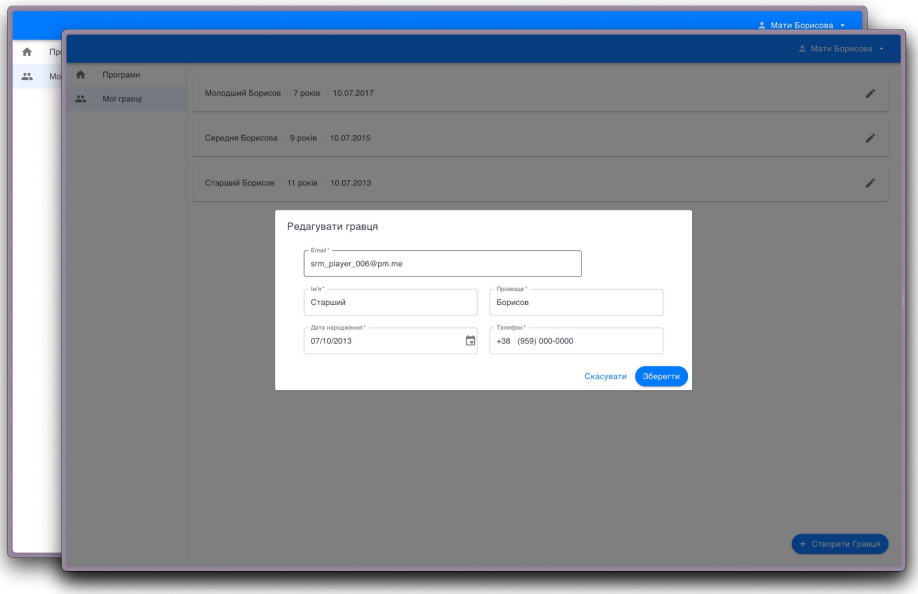
13

Перелік програм для опікуна



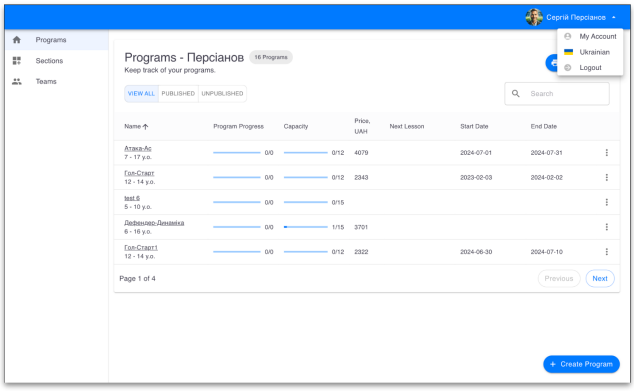
14

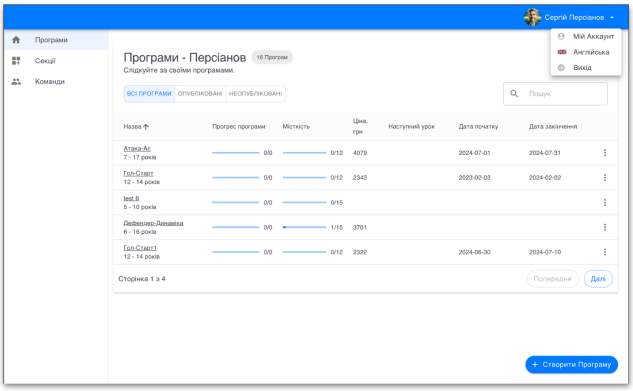
Управління гравцями опікуна



15

Скріншоти інтерфейсу. Локалізація.





16

Фрагменти коду з прикладом функцій валідації полів введення

```
export default function useValidationHelpers() {
 const { t } = useTranslation()
...
const validateDateGreaterThanField = useCallback(
 (otherFieldName: string): FieldValidator<string> =>
   (value, allValues) => {
     const dateValue = dayjs(value)
     const otherFieldValue = dayjs(
       (allValues as Record<string,
string>)[otherFieldName]
     )
     if (!dateValue.isValid()) {
       return t('error_messages.invalid_date')
     }
     if (dateValue.isBefore(otherFieldValue)) {
       return t('error_messages.must_be_greater_than', {
otherFieldName })
     }
     return undefined
   },
 [t]
)
...
```

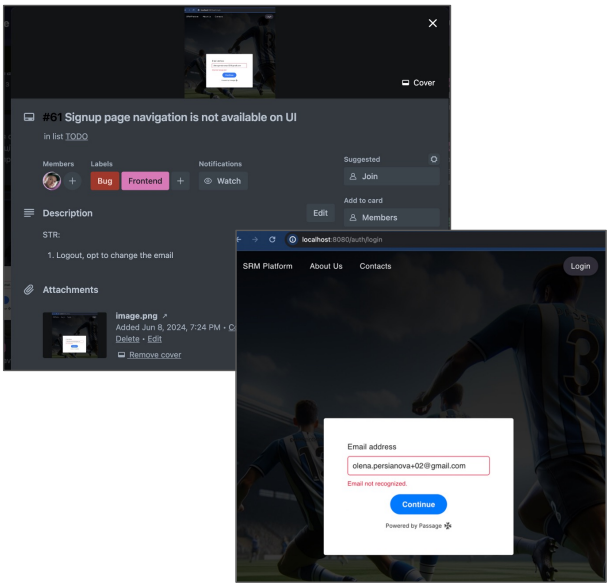
```
...
const validateNumberInRange = useCallback(
 (min: number, max: number): FieldValidator<string>
=>
   value => {
     const numberValue = Number(value)
     if (isNaN(numberValue)) {
       return t('error_messages.invalid_number')
     }
     if (numberValue < min || numberValue > max) {
       return t('error_messages.out_of_range', {
min, max })
     }
     return undefined
   },
 [t]
)

...
```

17

Функціональне тестування (мануальне)

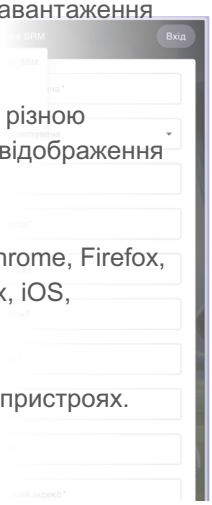
Trello

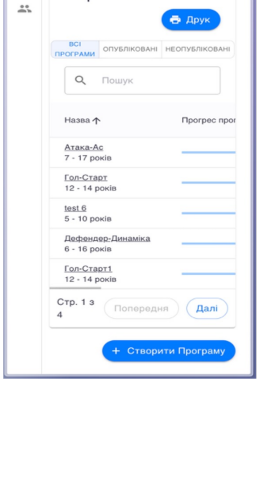
- У випадку виявлення багу, у системі Trello створювався тікет
- Приклади знайдених та виправлених багів:
  - баг анонімного режиму: відсутність опції «Реєстрація» в меню (тест функціональності «Вихід і вхід»)
  - баг авторизованого режиму: для авторизованого користувача відображалася опція «Увійти» замість «Вийти»
  - баг у функціональності друку звіту з власних програм: звіт не фільтрував програми

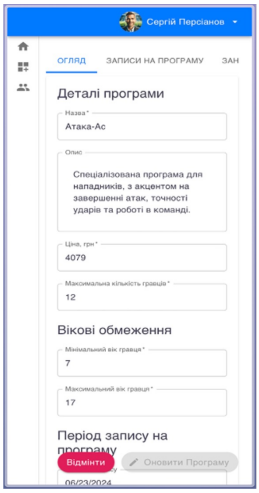


Тестування інтерфейсу

- Проведено ручне тестування реєстрації користувачів, редагування профілю, завантаження сертифікатів, вхід-вихід, зміну мови.
- Адаптивність перевірено на екранах з різною роздільною здатністю для коректного відображення інтерфейсу.
- Тестування проведено в браузерах Chrome, Firefox, Safari, Edge на Windows, macOS, Linux, iOS, Android.
- Інтерфейс працює коректно на різних пристроях.







19

Висновки

- Було спроєктовано користувацький інтерфейс за допомогою інструменту Figma;
- розроблено клієнтську частину (реактивний інтерфейс) програмної системи з адміністрування футбольних секцій за допомогою бібліотеки ReactJS та її екосистеми;
- запроваджено локалізацію програмної системи англійською мовою за допомогою бібліотеки i18next;
- проведено мануальне функціональне тестування та тестування інтерфейсу.

20