МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ПРОВЕДЕННЯ ЗМАГАНЬ З ПЛАВАННЯ

Спеціальність 122 «Комп'ютерні наук»

122 – БКР – 402.21810314

Виконав студент 4-го курсу, групи 402
В. В. Котляренко
«20» червня 2022 р.

Керівник: канд. техн. наук, доцент
Є. О. Давиденко
«20» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних наук

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____» _____20___р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук

Котляренко Владиславу Валентиновичу.

1. Тема кваліфікаційної роботи «Інформаційна система підтримки проведення змагань з плавання».

Керівник роботи Давиденко Євген Олександрович, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «7» грудня 2021 р. №318

2. Строк представлення кваліфікаційної роботи студентом «28» червня 2022 р.

3. Вхідні (початкові) дані до роботи: попередньо затверджені заяви команд на змагання.

Очікуваний результат: вебплатформа, яка пропонує створення користувачів різних ролей для допомоги в організації та проведенні змагань з плавання. Користувачі мають можливість створювати спеціальні заяви та протоколи, додавати спортсменів, які візьмуть участь у змаганнях. Система дозволяє переглядати рекорди, рейтинги та розряди. При завершенні запливів, організатор

має додати результати до системи, яка за допомогою алгоритму обрахунку очок визначить переможців.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- аналіз вимог та існуючих аналогів;
- створення клієнт-серверної архітектури;
- розробка програмного забезпечення інформаційної системи;
- проведення тестування системи;
- аналіз результатів розробки.

5. Перелік графічного матеріалу: презентація, 76 сторінок основної частини, 29 рисунків, 18 посилань.

6. Завдання до спеціальної частини: «Аналіз вимог до організації заходів техніки безпеки та охорони праці під час роботи за комп'ютером»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Канд. техн. наук, доцент Алексєєва Анна Олександрівна	

Керівник роботи ___канд. техн. наук, доцент Давиденко Є. О._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Котляренко В. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «23» листопада 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Інформаційна система підтримки проведення змагань з плавання.

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	03.10.2021р.	04.10.2021р.	виконано
2.	Огляд літератури за темою роботи	06.10.2021р.	08.10.2021р.	виконано
3.	Аналіз предметної області	15.10.2021р.	15.10.2021р.	виконано
4.	Розробка проєктних рішень	03.11.2021р.	04.11.2021р.	виконано
5.	Моделювання та конструювання ПЗ	05.11.2021р.	10.11.2021р.	виконано
6.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	16.11.2021р.	22.02.2022р.	виконано
7.	Розробка спеціальної частини з охорони праці	14.05.2022р.	28.05.2022р.	виконано
8.	Відгук керівника КРБ	14.06.2022р.	14.06.2022р.	виконано
9.	Оформлення КРБ та презентації	08.02.2022р.	20.05.2022р.	виконано
10.	Попередній захист	31.05.2022р.	31.06.2022р.	виконано
11.	Рецензування	15.06.2022р.	15.06.2022р.	виконано
12.	Завершення оформлення КРБ та презентації	15.06.2022р.	21.06.2022р.	виконано
13.	Захист кваліфікаційної роботи	28.06.2022р.	28.06.2022р.	виконано

Розробив студент Котляренко В. В. _____
(прізвище та ініціали) (*підпис*)

Керівник роботи канд. техн. наук, доцент Давиденко Є. О. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202__ р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402
ЧНУ ім. Петра Могили
Котляренко Владислав Валентиновича
Тема: «Інформаційна система підтримки проведення змагань з плавання»

Дана робота присвячена розробці програмного забезпечення для підтримки та організації проведення змагань з плавання базуючись на основі протоколів та використовуючи алгоритми обрахування очок FINA.

Об’єктом дослідження є процеси підтримки проведення змагань з плавання.

Предметом дослідження є організація клієнт-серверна архітектура, що має взаємодію з базою даних, та можливістю адаптації згідно з потенційно різними вимогами організації змагань.

Метою роботи є підтримка інтерактивних дій щодо створення організаційного підґрунтя проведення змагань з плавання за рахунок розробки вебплатформи.

Пояснювальна записка бакалаврської кваліфікаційної роботи складається зі вступу, чотирьох розділів, висновків та додатків.

У вступі обґрунтовується актуальність теми, описуються поставлена задача, предмет дослідження та об’єкт дослідження.

У першому розділі описується огляд існуючих систем, метою яких ставиться автоматизація процесу проведення змагань та використання функціональності подібної до нашої реалізації, а саме: використання алгоритмів обрахунку результатів, створення єдиного рейтингу, можливість масштабування системи згідно з вимогами змагань. Також, в рамках першого розділу описуються основні проблеми та задачі які вирішує система.

У другому розділі визначаються основні функціональні та інформаційні можливості системи: збір інформації про учасників змагань, їх школи, команди та попередні результати, збір інформації про правила проведення змагань, обробка отриманої інформації, збір інформації про результати змагань, обрахування

переможних балів. Процеси можливих сценаріїв застосування системи при організації проведення змагань наведені у схематичному вигляді за допомогою блок-схем.

У третьому розділі описуються архітектура, проєктування та технічна реалізація системи, а також технології, та алгоритми, що допомогли у створенні програмного забезпечення для автоматизованої підтримки проведення змагань з плавання згідно наданих даних.

У четвертому розділі надається інформація стосовно тестування розробленого програмного забезпечення, аналіз результатів та обчислень до проведення змагань та після. Представлені знімки екрану інтерфейсу вебплатформи та результатів роботи системи.

У висновках проводиться аналіз роботи та отриманих результатів.

Кваліфікаційна робота містить 76 сторінки основної частини, 29 рисунків, 18 посилань.

Ключові слова: *автоматизація, клієнт-серверна архітектура, оптимізація, протоколи, заяви.*

ABSTRACT

to the bachelor's qualification work by the student of group 402 of

Petro Mohyla Black Sea National University

Kotliarenko Vladyslav Valentynovych

"Information system to support swimming competitions"

This project is dedicated to the development of software to support and organize swimming competitions based on protocols and using FINA scoring algorithms.

The object of research is the processes of supporting swimming competitions.

The subject of research is technologies and tools for developing client-server architecture, interaction with the database.

The aim of the work is to develop a web platform for quick and easy creation of protocols of swimming competitions, with the ability to enter and calculate results, as well as viewing a single rating based on the data entered.

The explanatory note of the bachelor's thesis consists of an introduction, four chapters, conclusions and appendices.

The introduction substantiates the relevance of the topic, describes the task, the subject of research and the object of research.

The first section describes the existing systems, which aim to automate the process of competition and use functionality similar to our implementation, namely: the use of algorithms for calculating results, creating a single rating, the ability to scale the system according to competition requirements. Also, the first section describes the main problems and tasks solved by the system.

The second section defines the main functional and information capabilities of the system: collecting information about competitors, their schools, teams and preliminary results, collecting information about the rules of the competition, processing information, collecting information about the results of competitions, calculating winning points. The processes of possible scenarios of application of the system in the organization of competitions are given in schematic form with the help of block diagrams.

The third section describes the architecture, design and technical implementation of the system, as well as technologies and algorithms that helped to create software for automated support for swimming competitions according to the data provided.

The fourth section provides information on testing the developed software, analysis of results and calculations before and after the competition. Screenshots of the web platform interface and system results are presented.

The conclusions analyze the work and the results obtained.

Thesis contains pages – 76 , images – 29, references – 18.

Keywords: *automation, client-server architecture, optimization, protocols, applications.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

JS – JavaScript

ORM – Object-Relational Mapping

UX – user experience

UI – user interface

WS – websockets

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКИ ПРОВЕДЕННЯ ЗМАГАНЬ З ПЛАВАННЯ»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 402.21810314

Виконав студент 4-го курсу, групи 402
_____*В. В. Котляренко*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Керівник: ___*канд. техн. наук, доцент*___
(наук. ступінь, вчене звання)
____*Є. О. Давиденко*
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Кожен день дарує нам нову інформації з технологічного фронту, адже в наш час будь-яка людська праця поступово переходить до аналогів у вигляді автоматизованої програми, що набагато полегшує процес виконання потрібного завдання та суттєво економить час. Протягом всього часу існування спортивних комплексів плавальних басейнів судді змагань самостійно займалися організаційними питаннями, такими як: створення єдиної бази учасників, рейтингу, протоколу, формування запливів, сортування учасників за різними категоріями, обрахування очок FINA, перевірка на здобуття нового розряду тощо. Завдяки розробці даного продукту вищенаведені потреби становлять мінімальні переймання для суддів та організаторів.

Використання сучасної інформаційної системи відкриває нові можливості організації та проведення змагань. Наразі існує багато різноманітних програм та платформ для освітлення та проведення спортивних подій. Незважаючи на різноманітність видів спорту, методи реалізації та архітектура необхідного програмного забезпечення напрочуд схожі між собою та розрізняються лише окремими доступними функціями та візуальною оболонкою.

Інформаційною системою підтримки проведення змагань можуть користуватися люди, які займають посади в організаційній сфері, та безпосередньо спортсмени та тренери з плавання. Кожен з них знайде програмне забезпечення важливим для себе через ряд функцій, що дозволяють швидко досягати свої цілі.

Проаналізувавши низку інтернет-ресурсів [1, 2, 3], зокрема офіційний ресурс Федерації плавання України [4], можна зробити висновок, що, на жаль, допоміжне програмне забезпечення, що використовується задля організації всеукраїнських змагань, не є публічним, та, фактично, закрите від потенційних користувачів у разі, якщо вони не займають певної посади у Федерації плавання України.

Актуальність розробки доступної інформаційної системи для пересічного користувача важко переоцінити, адже в наш час будь-яка людська праця поступово переходить до аналогів у вигляді автоматизованої програми, що набагато полегшує

процес виконання потрібного завдання та суттєво економить час. При інтенсивних проведеннях змагань слід оптимізувати частини мануальною роботи до машинної, тим самим полегшити та прискорити процеси організації.

Об'єктом роботи є процеси підтримки проведення змагань з плавання.

Предметом роботи є клієнт-серверна архітектура, що має взаємодію з базою даних, та можливістю адаптації згідно з потенційно різними вимогами організації змагань.

Метою роботи є підтримка інтерактивних дій щодо створення організаційного підґрунтя проведення змагань з плавання за рахунок розробки вебплатформи.

Для виконання мети поставлені **наступні завдання**.

1. Проаналізувати всі необхідні можливості програми, та обрати відповідне середовище розробки.
2. Створити зручний UI (згодом і UX) для отримання максимального комфорту та чіткості під час використання даного ПЗ користувачем.
3. Розробити всі алгоритми та логіку застосунку.
4. Налаштувати масштабування та інтеграцію під різні платформи.
5. Навчитися програмно використовувати можливості мови програмування та дотримуватись принципів ООП.
6. Повністю протестувати вебплатформу.

Практичне значення отриманих результатів. Розроблена вебплатформа дає можливості: створювати заяви та протоколи змагань, використовувати їх для підтримки проведення змагань з плавань, переглядати рейтинги, рекорди, систему розрядів, раніше проведені змагання, обраховувати результати FINA через спеціальний калькулятор.

Структура кваліфікаційної роботи. Унаслідок дослідження задач було організовану наступну структуру кваліфікаційного проєкту: вступ, 5 розділів, висновки, список використаних джерел із 18 найменувань. Загальний обсяг: 85 сторінок.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ ПІДТРИМКИ ПРОВЕДЕННЯ ЗМАГАНЬ. ПОСТАНОВКА ЗАДАЧІ

Технології розвиваються швидко та безперервно. Вони з кожним днем охоплюють все більше і більше побутових речей в житті людини. Прагнення до оптимізації та автоматизації різних сфер діяльності призводить до практичного прогресу у створені нового програмного забезпечення як під глобальну проблему, так і для локальної цілі з можливостю оновлення або розширення функціональності згідно з новітніми задачами.

1.1 Опис предметної сфери

Автоматизована інформаційна система — це взаємозв'язана сукупність даних, обладнання, програмних засобів, персоналу, стандартних процедур, які призначені для збору, обробки, розподілу, зберігання, представлення інформації згідно з вимогами, які випливають з цілей організації. Сьогодні, у вік інформації, практично кожна інформаційна система використовує комп'ютерні технології. Основними факторами, які вплинули на рішення впровадження інформаційної системи підтримки проведення змагань з плавання, є потреби організаторів, суддів та тренерів, які беруть участь безпосередньо у процесі організації змагань, задля оптимізації процесів, збільшення продуктивності у порівнянні з рутинною, мануальною діяльністю, як створення заяв, протоколів, рейтингів та запливів [6].

Заява для змагань з плавання створюється попередньо та являє собою документ, який ідентифікує кожного спортсмена, що буде брати участь від конкретної команди. У заяві вказуються наступні поля:

- ім’я, прізвище, рік народження, тип участі спортсмена;
- ім’я, прізвище тренера;
- дистанція, стиль плавання, час, тип запливу, попередній заявочний результат, наявний розряд;
- місто, область, країну спортсмена;
- спортивну школу;

– добровільне спортивне товариство.

Типи участі спортсмена у змаганнях з плавання вважаються додатковими показниками перед формуванням протоколів. Учасник, що має показник багатоборства, готовий змагатися різними стилями, та в підсумку мати єдиний сумарний результат та навпаки, учасник, який є поза конкурсу, взагалі не може конкурувати з іншими та виконує заплив виключно заради власних цілей.

Стартовий протокол змагань – це документ, який є сукупністю заяв з усіх команд, що беруть участь у змаганнях. Завдяки даному документу судді мають можливість корегувати та створювати стартові запливи згідно з регламентом. Після формування стартових протоколів учасники отримують інформацію, що стосується їхніх запливів та черги, в якій вони перебувають. При завершені запливів судді виставляють час, та створюють новий документ – протокол.

Протокол змагань – це результуючий документ, що включає в собі інформацію щодо учасників, їхні запливи та результати, та формує рейтинги змагань. Доступність універсальних функцій дозволяють суддям формувати будь-які рейтинги згідно з регламентом змагань. Це можуть бути або виключно дівчата та хлопці, або ж змішаний командний залік, або за результатами конкретного стилю плавання.

Інформаційна система проведення змагань з плавання має ряд переваг. По-перше, організатори та судді мають низку автоматизованих процесів, що дозволяє коректно займатись організацією не втрачаючи дорогоцінного часу на ручну працю. По-друге, зменшується ризик людського фактору, що може впливати на поспішні результати, а також час, протягом якого визначаються результати. Універсальність дозволяє створювати всі існуючи варіації проведення офіційних змагань, а також «неіснуючі», що можуть бути придумані тренером для окремої команди задля вирішення більш локальних задач.

Серед недоліків слід виділити обов’язкову наявність електронної техніки такої, як комп’ютер, ноутбук, планшет, телефон, але враховуючи те, що на телефоні або планшеті частина функціоналу може бути незручною через замалу висоту та

ширину екрану. Присутність інтернет зв’язку також є необхідною мірою для того, щоб мати можливість використовувати вебплатформу.

Результати змагань формуються після обрахунку очок FINA. Таблиця очок FINA дозволяє порівнювати результати різних змагань. Підрахунок очок FINA надає бали за результати плавання, більше балів за виступи світового класу, як правило, 1000 або більше і менше балів за повільніші результати. Точки обчислюються за допомогою кубічної кривої. З часом плавання (T) і базовим часом (B) у секундах, бали (P) обчислюються за наступною формулою:

$$P = 1000 * (B / T)^3 \quad (1.1)$$

Формула використовується для обчислення балів від часу. Усі значення балів обрізаються до цілого числа. У таблицях балів із назвами 2009 року і раніше значення балів округляються. Якщо необхідно обчислити необхідний час (T) для певної кількості балів (P), то для обчислення першої оцінки використовується точна формула. Потім час слід зменшити на одну соту секунди, доки зворотне обчислення з часом все ще приведе до початкової кількості балів. Базовий час визначається щороку на основі останнього світового рекорду, затвердженого FINA. Для короткого курсу (SCM) базовий час визначається з перерізом дати 31 серпня. Для довгого курсу (LCM) базовий час визначається в кінці року (31 грудня). Для «Нарахування балів FINA 2021» це означає, наприклад: час короткого курсу до «31 серпня 2022 року», тривалість курсу до «31 грудня 2021 року». Базові часи публікуються на вебсайті FINA протягом одного місяця після закінчення відповідного періоду. У файлах нижче згадується формула для розрахунку балів FINA, базовий час, використаний у формулі, і бали, які будуть нараховані спортсменам для досягнення певного часу в LCM або SCM.

1.2 Огляд та аналіз наявних аналогів

Існує багато різноманітних систем, які використовуються для підтримки проведення змагань з різних видів спорту, починаючи з приватних програм, дозвіл до яких відсутній у звичайного користувача, закінчуючи публічним програмним

забезпеченням, ціллю якого охопити певний відсоток аудиторії, заінтересувати та допомогти в організації важливих процесів в створені змагань. На жаль, в проєкті не буде розглянуто приватних програм через брак доступу до них, проте буде проаналізовано доступний сегмент інформаційних систем.

Першою на черзі до аналізу є система The LTA Competition Management System, що пропонується для підтримки проведення змагань з тенісу. Система була надана LTA для полегшення процесу онлайн-заявки, щоб допомогти організаторам турнірів по всій Великій Британії керувати тенісними змаганнями. Система забезпечує безперебійну роботу всіх користувачів протягом усього циклу турніру як для організаторів, так і для гравців.

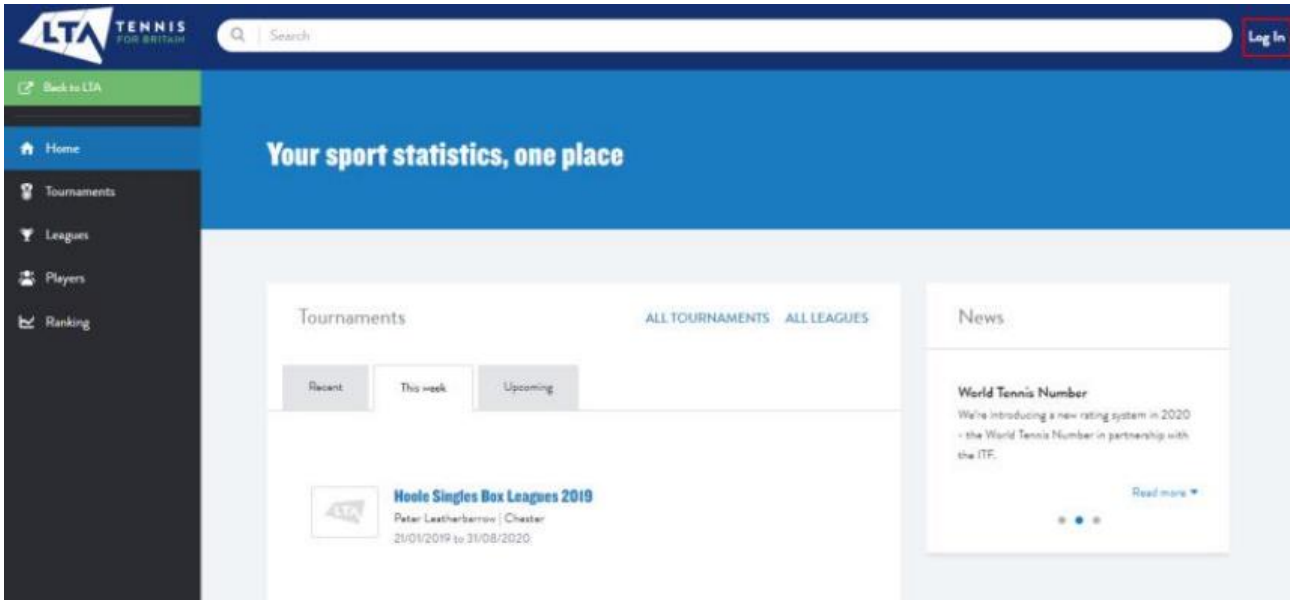

Рисунок 1.1 – Вигляд системи LTA

Процес створення турніру поділяється на декілька кроків. Крок перший – загальна інформація. Спочатку вас попросять надати деякі деталі про турнір.

1. Ім'я.

2. Місце проведення: почніть вводити назву місця, для якого хочете надати доступність зі спадного списку. В списку будуть лише зареєстровані LTA.

3. Дати початку та закінчення – які мають бути в період цього санкціонованого вікна.

4. Оцінка – яка буде визначена оцінками, доступними в цей період санкцій.

5. Стандартні стартові внески для одиночних та парних.

6. Інформація про спонсора.

7. Ім'я організатора турніру.

8. Ім’я арбітра турніру.

Перевірка суддівської ліцензії та її оцінки буде проводитися, щоб переконатися, що вони відповідають мінімальним вимогам до суддівства. Ви можете приховати контактні дані рефері та мати можливість не вказувати ім’я рефері на цьому етапі. Після того, як ви заповнили всі ці поля (які є обов’язковими), натисніть «Далі». Усі змагання можуть бути подані без Рефері, однак вам потрібно буде переконатися, що на змагання буде призначений Рефері з відповідною оцінкою Рефері до затвердження. Крок 2 – Події. На сторінці «Події» буде надано події, які можна запускати залежно від типу програми. Усі події, на які можна подати заявку, будуть перераховані за типом та віковою групою.

Кожна подія закодована:

– BS/GS - одиночні хлопчики/одиночки дівчата;
– BD/GD – парний розряд хлопчиків/парний розряд дівчат;
– MS/WS – чоловічий одиночний розряд/жіночий одиночний розряд;
– MD/WD – чоловічий парний розряд/жіночий парний розряд;
– XD – змішаний парний розряд;
– S – неодружені (будь-якої статі);
– D – парний (будь-якої статі).

Крок 3 – інформація про події:

Для кожної вибраної події буде запропоновано вказати дати початку та закінчення, дату прийняття заяв, терміни закриття та відкликання, а також підтвердити оцінку та вступний внесок. Деякі терміни закриття та відкликання можна буде редагувати, тоді як інші будуть заблоковані, щоб запобігти їх зміні. Така інформація, як інформація про час та попередній розмір розіграшу, буде включена в розділ процесу, який буде надано пізніше. Це гарантує, що користувач надає лише необхідну інформацію на кожному етапі.

Унаслідок аналізу даної інформаційної системи можливо сформулювати наступні переваги та якості продукту:

– гарний та «спокійний» UI;
– універсальний функціонал;
– можливість підтримки проведення змагань.

До негативних показників слід віднести UX, незрозумілість як почати змагання, та на якому етапі верифікації знаходиться протокол змагань.

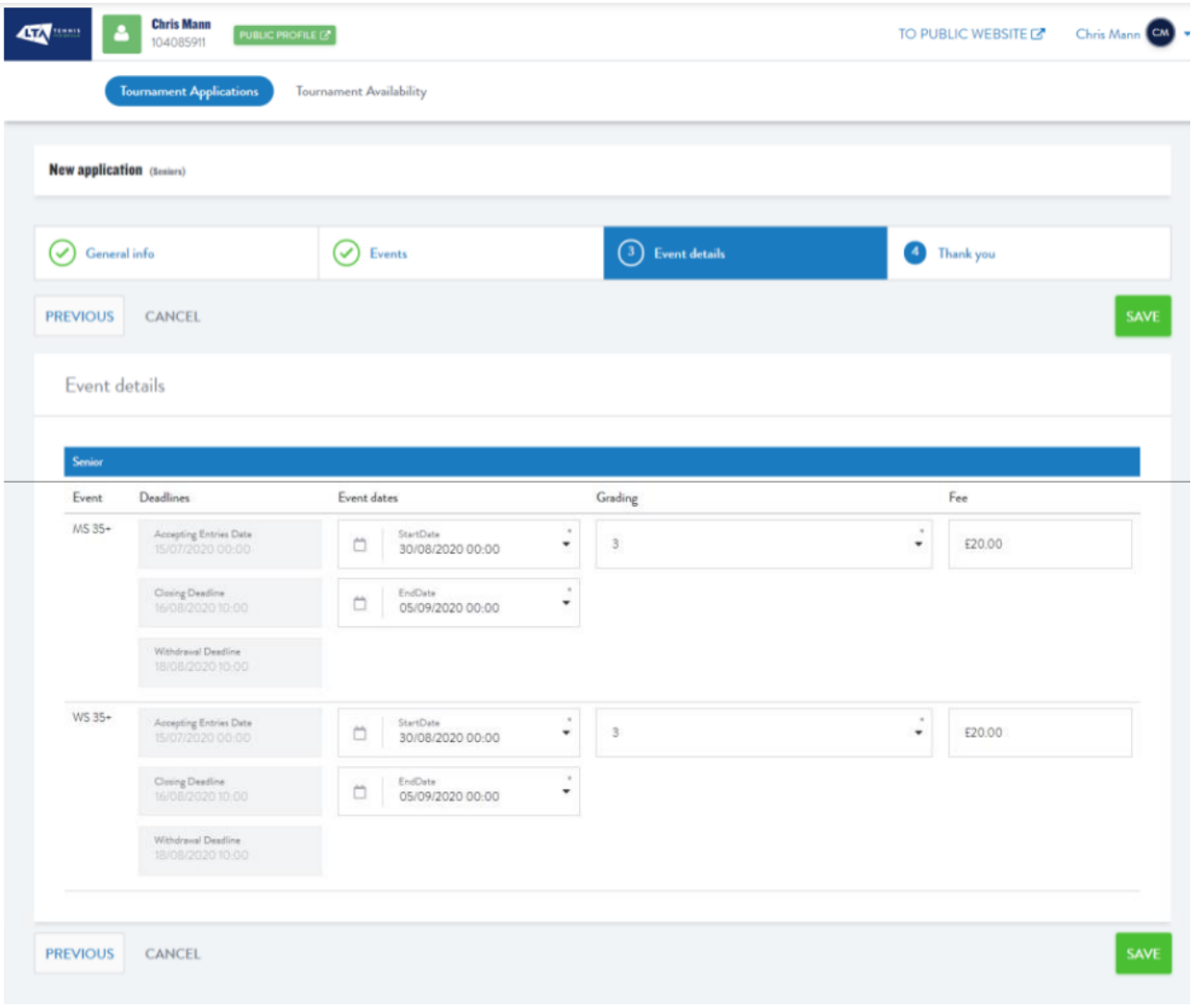

Рисунок 1.2 – Створення протоколу змагання у програмі Competition Management System

Наступний аналог для аналізу – це інформаційна система Friendly Manager. Friendly Manager використовується для підтримки змагань з різних видів спорту, тобто в підґрунтя системи покладено архітектуру кросфункціональності задля

універсальних можливостей. С самого початку застосунок вітає потенційного користувача пропозицією спробувати демо-версію, що є необхідною функцією, адже застосунок не безкоштовний.

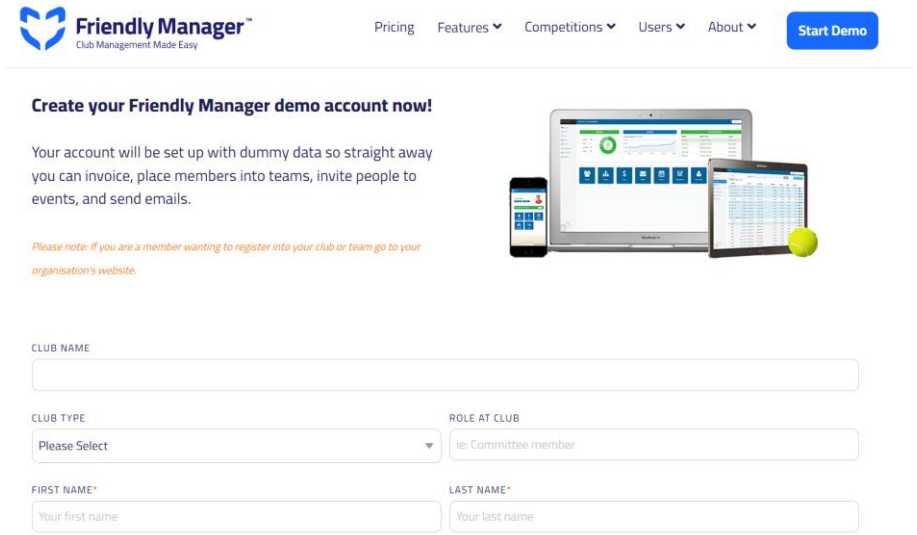

Рисунок 1.3 – Демо-сторінка застосунку Friendly Manager

Головною особливістю даної системи є пропозиції багатьох функцій

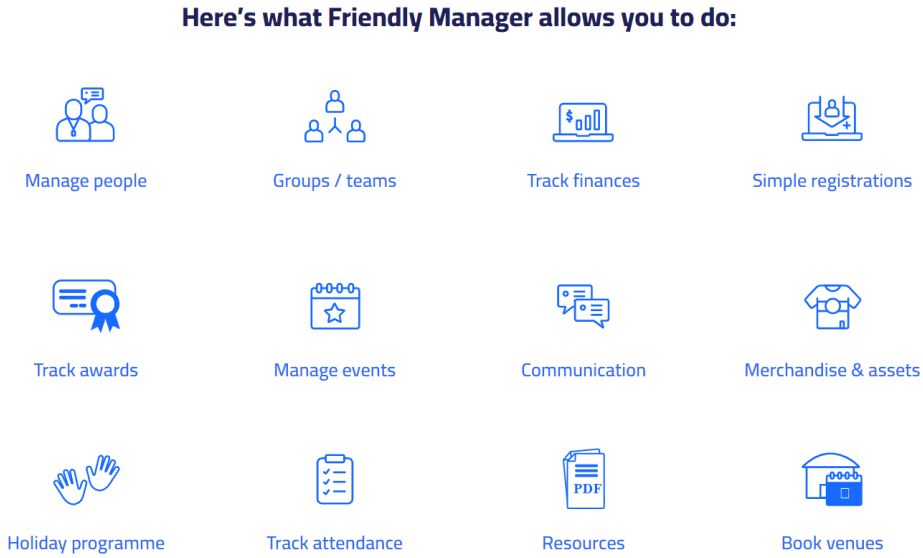

Рисунок 1.4 – Функції застосунку Friendly Manager

Але, в той же час, Friendly Manager є платним, а отже користувачі повинні сплачувати за певний набір функцій, не дивлячись на те, що деякі з них можуть зовсім не знадобиться. Саме цей фактор більше відштовхує від даного застосунку, аніж приваблює.

1.3 Специфікація розробки програмного рішення

Метою роботи є підтримка інтерактивних дій щодо створення організаційного підґрунтя проведення змагань з плавання за рахунок розробки вебплатформи.

Для виконання мети поставлені наступні завдання:

1. Проаналізувати всі необхідні можливості програми, та обрати відповідне середовище розробки.
2. Створити зручний UI(згодом і UX) для отримання максимального комфорту та чіткості під час використання даного ПЗ користувачем.
3. Розробити всі алгоритми та логіку застосунку.
4. Налаштувати масштабування та інтеграцію під різні платформи.
5. Навчитися програмно використовувати можливості мови програмування та дотримуватись принципів ООП.
6. Повністю протестувати вебплатформу.

Програмний застосунок повинен відповідати наступним вимогам:

1) Під час роботи із заявами, протоколами та результатами проводиться автоматичне збереження даних.

2) Унаслідок тестування не було знайдено помилок чи негативних попереджень.

Програмний застосунок має простий UI та може використовуватися користувачами будь-якого рівня, що не мають спеціальних професійних знань в роботі з проведення змагань та мають елементарні навички роботи з ПК.

Програмний застосунок може використовуватись за потребою і не має обмежень, пов’язаних з середовищем роботи чи підключенням до глобальної мережі інтернет.

Для реалізації даної інформаційної системи було обрано за основу клієнт-серверну архітектуру. Оптимальними технологіями для розробки стали React та Nest, що використовують мови JavaScript та TypeScript відповідно.

React – JavaScript-бібліотеки для створення користувацьких інтерфейсів.

Однією з особливостей Реакту є ефективність. Оновлення всього DOM, щоб зробити вебсторінку «реактивною» – вважається вкрай неефективним, оскільки споживає занадто багато ресурсів. Тому, власне, при зміні вебсторінки (наприклад, в результаті запиту або дії користувача) React оновлює віртуальний спеціальний DOM. React зберігає дві версії віртуального DOM – оновлений віртуальний DOM та його резервну копію, створену до оновлення. Після оновлення React порівнює обидві версії між собою, щоб знайти змінені елементи, а потім – оновлює частину реального DOM, що виключно змінилася.

Іншою перевагою Реакту є висока продуктивність. Одна з життєво важливих цілей будь-якого стартапу —– написати вебзастосунокк швидким та чуйним, забезпечити найкраще обслуговування клієнтів. Віртуальна DOM, на відміну від реального DOM, займає мало місця та швидко оновлюється, тим самим підвищуючи продуктивність програми. Віртуальна DOM дозволяє сторінці негайно отримувати відповіді від сервера та відображати оновлення. Наприклад, Facebook застосовує технологію віртуального DOM для оновлення чатів та стрічок користувачів без перезавантаження сторінки.

Висновки до розділу 1

Використання нових інформаційних технологій у різних сферах діяльності людини призвело до розробки програмного забезпечення для оптимізації та автоматизації процесів. Створення програмних продуктів створюється насамперед через цілі вирішення певних задач, що постають перед людьми.

Програмне забезпечення для створення змагань — це інструмент, який можна використовувати для організації та проведення змагань. Оскільки навколишній світ підіймає планку на модернізації, то все більше і більше сфер діяльності змінюються та отримують нові технології, що покращують та оптимізовують процеси.

Щоб полегшити проведення змагань, тренери та організатори починають використовувати програмне забезпечення, що націлене на вирішення проблем, пов’язаних з організацією процесів таких, як створення та заповнення заяви, протоколів, стартових запливів та результатів. Для вирішення даних проблем створюються нові програмні застосунки, що можуть повністю викреслити мануальне проведення змагань.

У першому розділі було розглянуті деякі програми, ціллю яких було покращення взаємодію між людиною та організаційними процесами. Одними з найпопулярніших рішень є застосунки Friendly Manager та Competitions Management Systems. Більшість інформаційних систем підтримки проведення змагань є схожими між собою, мають платні функції, або складний інтерфейс, що буде відштовхувати потенційних клієнтів.

Завдання розробити інформаційну систему для проведення змагань з плавання є безумовно актуальним та необхідним, так як ставить перед собою вирішення всіх питань, що непокоять суддів, тренерів, організаторів. Вебплатформа має гарний та цілком зрозумілий інтерфейс, а також відповідає вимогам, що необхідні для повної модернізації даної сфери, та покращення якості проведення змагань.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Моделі для вирішення задачі

Кожного разу, коли є потреби в організації процесів роботи над програмним забезпеченням, менеджер проєкту має вибір з-поміж декількох десятків моделей розробки.

Модель розробки програмного забезпечення — це набір процесів і методологій, реалізованих для розробки проєкту. Існує багато типів моделей життєвого циклу розробки програмного забезпечення, які компанії, команди, або ж самостійні розробники активно можуть використовувати для досягнення поставлених цілей.

Перелік важливих факторів, які впливають на вибір моделей задля подальшого корегування правил та прямого використання на проєкті:

- процеси проведення тестування;
- готовність презентувати виконану роботи;
- необхідна функціональність;
- надійність;
- точність;
- простота використання;
- рівень технічної складності.

Обрана модель надзвичайно важлива в розробці програмного забезпечення з багатьох причин. Це буде диктувати напрямок і результати проєкту з самого початку. Починаючи з однієї моделі, потім її не можна змінити на іншу, адже це може мати різноманітні наслідки, які можуть вплинути на працездатність команди та ведення проєкту.

Саме тому часто використовуються різні моделі розробки програмного забезпечення для різних проєктів. Масштаб зусиль і терміни, безсумнівно, будуть

факторами, особливо при роботі з клієнтами, які вимагають миттєвих результатів. Коли справа доходить до рівня майстерності, моделі можуть змінюватися.

Розглядаючи понад 50 моделей розробки програмного забезпечення варто пам’ятати, що кожна з них має шанс або досягти поставної мети, або навпаки перешкодити прогресу виконання завдання. Кожен розробник програмного забезпечення знає скільки часу витрачається на помилку, та скільки потенційно це може коштувати проєкту. Також слід мати на увазі, що процеси організації можуть погіршитися, якщо в результаті аналізу буде обрана невідповідна модель розробки програмного забезпечення.

Аналізуючи все сказане, варто визначати моделі програмного забезпечення, які мають повністю покривати потреби продукту. Фактори, які необхідно дійсно враховувати під час зважування різних моделей програмного забезпечення в програмній інженерії:

Таблиця 2.1 – Фактори вибору моделі програмного забезпечення.

Фактор	**Пояснення**
Часові рамки	це, в основному, кількість часу, відведеного на різні аспекти, наприклад: коли слід отримати перші результати, скільки часу дається на завершення завдань, чи розглядається продовження проєкту, створення наступних версій тощо.
Умови	обмеження, компенсації та бюджет, які входять до проєкту, а також пункти щодо термінів та необхідні докази прогресу.
Розмір	наскільки великий проєкт, скільки людей залучено та яка його роль на підприємстві.
Рівень інженерних навичок	кількість та формат залучення спеціалістів різних рівнів (junior, middle, senior).

Кінець таблиці 2.1.

Обсяг проєкту	скільки сфер діяльності буде охоплено, наскільки універсальні програми та наскільки масштабним буде вплив на конкретну галузь.
Мета	мета проєкту, його особливості, потенційні можливості застосування та конкретні галузі, в яких він буде використовуватися.
Мова програмування	які мови програмування будуть використовуватися для проєкту.

Це ті фактори, які повинні вплинути на рішення, коли виконується вибір серед типів моделей життєвого циклу розробки програмного забезпечення. Це рішення потрібно приймати обережно, оскільки пізніше буде неможливим легко його змінити, особливо коли прогрес вже досяг певної точки, після якої потрібно буде починати з нуля, якщо будуть внесені значні зміни.

Як було наведено вище, на даний момент офіційно визнано 50 моделей розробки програмного забезпечення, включаючи моделі інкрементного та ітераційного процесу розробки. Хоча, враховуючи мінливу природу галузі, існують багато моделей, які ще не були офіційно розкриті, або які є настільки нішовими, що підходять лише для дуже специфічних проєктів.

Виходячи з вищенаведених фактів, є потреба в аналізі найбільш бажаних варіантів для багатьох розробників програмного забезпечення та IT-проєктів, які також є найбільш зручними для користувачів.

Agile – найбільш широко використовувана модель розробки програмного забезпечення в галузі завдяки тому, що це неймовірно динамічний і гнучкий процес управління проєктами. Незалежно від того, скільки людей працює над проктом, співпрацювати легко через те, що легко адаптуватися до змін, що вносяться ринком або клієнтами.

На відміну від багатьох інших моделей у цьому списку, методологія agile фактично народилася з іншої моделі, яку проаналізуємо згодом. Передбачалося, що

вона буде набагато універсальнішою і менш схильнішою до жорсткості, що дозволить програмістам реагувати на мінливі вимоги за потреби.

Серед найбільш значущих аспектів цієї конкретної моделі є Agile Manifesto, який містить чотири основні принципи. Фактично, це структура, яка робить гнучку методологію такою, якою вона є. Крім того, є також 12 ключових принципів, які служать якорями, які тримають проєкти на правильному шляху.

Чотири основні цінності:

– особи та взаємодія з процесами та інструментами;
– працююче програмне забезпечення з доступною документацією;
– співпраця з клієнтами протягом переговорів по контракту;
– швидке реагування на зміни згідно з планом.

Дванадцять ключових принципів:

– задоволеність клієнтів завдяки ранній та безперервній доставці програмного забезпечення;
– врахування мінливих вимог протягом процесу розробки;
– регулярні випуски працюючого програмного забезпечення;
– співпраця між зацікавленими сторонами бізнесу та розробниками;
– підтримка, довіра, мотивація залучених співробітників;
– міжособова взаємодія;
– основним показником прогресу є програмне забезпечення;
– гнучкі процеси для підтримки стабільного темпу розвитку;
– увага до технічних деталей так дизайну підвищує маневреність;
– простота;
– самоорганізаційні команди заохочують чудові архітектури, вимоги та дизайну;
– регулярні міркування про те, як працювати ефективніше.

Waterfall – одна з найперших моделей розробки програмного забезпечення. Дана модель є гарною точкою для порівняння того, наскільки далеко зайшло

кодування. По суті, це каскад фаз, де потрібно завершити, перш ніж рухатися далі. Це означає, що якщо ви хочете протестувати певний етап, спочатку всі етапи розробки програмного забезпечення мають бути виконані.

При роботі з моделлю waterfall немає накладень, і все робиться лінійно. Наведена модель працює наступним чином:

- аналіз вимог;
- проєктування системи;
- реалізація;
- тестування;
- розгортання;
- технічне обслуговування.

V Model – модель, що працює на паралельній фазі розробки, яка включає верифікацію та валідацію, які потім можна проілюструвати на діаграмі з літерою V. З одного боку знаходиться верифікація, яка йде разом з різними іншими фазами, а з іншого – валідація. Ця конкретна модель також позначена самими фазами, які поділяються на основі різних категорій.

Верифікація охоплює наступні фази:

- аналіз бізнес-потреб;
- проєктування системи;
- архітектурне проєктування;
- модульний дизайн.

Після виконання вищенаведених фаз наступає фаза кодування, де, фактично, починають закодовувати системні модулі. На даному етапі по-справжньому можливо оцінити вибір мови програмування.

Валідація обмежується наступним переліком фаз:

- unit тестування;
- інтеграційне тестування;
- тестування системи;

– приймальне випробування;

Incremental Model – це процес розробки, де модулі розбиваються на численні окремі блоки. Дана методологія передбачає охоплення чотирьох фаз вимог, проєктування, кодування та тестування, щоб сформувати один крок. Кожен з цих приростів буде діяти як частина один одного, додаючи функції, допоки не буде виконано поставленої цілі. Це робить його більш втомливим, ніж більшість інших моделей розробки програмного забезпечення, але також забезпечує більше контролю.

RAD Model – модель швидкої розробки застосунків є відгалуженням інкрементальної моделі, де в основному йде робота над компонентами окремо. Після цього вони будуть зібрані разом, щоб створити робочий прототип.

Таким чином, ця конкретна модель ідеально підходить, коли є ціль створити зразки, щоб швидко показати клієнтам, що вони потім зможуть прокоментувати. Більше того, кожен компонент можна призначити різним командам для подальшої збірки. Це можна зробити, просто дотримуючись етапів моделі розробки програмного забезпечення RAD, а саме:

– бізнес-моделювання;
– моделювання даних;
– моделювання процесів;
– генерація застосунків;
– тестування та оборот.

Iterative Model. Ця методологія, яку часто плутають з моделлю інкрементальної розробки програмного забезпечення, передбачає спочатку створення простої основи, перш ніж будувати на ній. При цьому рівень складності зростає, поки не буде досягнуто кінцевий продукт.

Існує інший спосіб поглянути на цю модель розробки програмного забезпечення як на двовимірну спіраль, що перекривається, де лінії регулярно повертаються назад, перш ніж рухатися вперед. Кожен крок завершується,

аналізується, коригується та перевіряється, перш ніж рухатися далі. Основні кроки ітеративної моделі:

- планування та вимоги;
- аналіз і проєктування;
- реалізація;
- тестування;
- оцінка.

Spiral Model – модель розробки програмного забезпечення, орієнтована більше на управління ризиками. Ця модель розробки програмного забезпечення є однією з найважливіших методологій у бізнесі. Поділ фаз тут можна розбити на такі чотири квадранти:

- визначення цілей та альтернативних рішень;
- виявлення та вирішення ризиків;
- розробка наступної версії продукту;
- перегляд та планування наступного етапу.

Prototyping Model – модель прототипування, що спеціально призначена для створення прототипу в робочому стані, який потім можна надати клієнтам для зворотного зв'язку. Це циклічний процес, який включає в себе робочий зразок, тестування клієнтом, отримання зворотного зв'язку, який потім використовується для покращення прототипу. Це робиться до тих пір, поки не буде отримано бажаний кінцевий продукт, але на відміну від інших моделей розробки програмного забезпечення, це дуже залежить від того, що говорять інші люди. Безумовно, є що сказати про те, щоб звернути увагу на ідеї, які клієнти можуть запропонувати в цьому типі розробки. Однак є ризик, що в кінцевому підсумку нікому не сподобається продукт, коли буде опубліковано остаточний результат.

Вибір між різними моделями програмного забезпечення в програмній інженерії є важливим кроком у розробці продукту. Моделі надають шляхи та засоби досягнення мети під час роботи над проєктом і дають чіткі вказівки, методи, як досягти мети, а також рішення, коли необхідно вирішити проблеми. Завдяки

найкращим моделям розробки програмного забезпечення, наведеним вище, є можливість проаналізувати кожну модель, достатньо підготуватись та зануритись в проєкт. Саме модель Agile пропонує найбільш ефективні методи розв'язання задач, які сформовані для інформаційної системи підтримки проведення змагань з плавання. Функціональність, універсальність та гнучкість є пріоритетними цілями для унаслідування на проєкті.

2.2 Технології розробки системи

Основні цілі розробки інформаційної системи підтримки проведення змагань з плавання передбачають використання зручного та універсального програмного забезпеченням, з єдиною базою даних, та з можливістю доступу через мережу інтернет. Враховуючи вимоги та специфікацію проєкту, було обрано за основу клієнт-серверну архітектуру, яка може покрити всі потреби замовника, а також потенційно легко масштабуватися у зв'язку з можливими майбутніми оновленнями, створенням нових версій.

Рисунок 2.1 – Візуальна схема клієнт-серверної архітектури

Клієнт-серверна архітектура передбачає використання бази даних, тобто сховище усіх необхідних даних для повноцінної роботи програми. Перед вибором

бази даних слід переглянути основні характеристики технологій тому, що кожна БД може мати свої плюси та мінуси в залежності від специфіки проєкту. У рамках створення інформаційної системи підтримки проведення змагань з плавання була обрана база даних PostgreSQL.

PostgreSQL – це потужна система об'єктно-реляційних баз даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші робочі навантаження даних. Витоки PostgreSQL сягають 1986 року в рамках проєкту POSTGRES в Каліфорнійському університеті в Берклі і має понад 30 років активної розробки на базовій платформі.

PostgreSQL заслужив міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти відкритих вихідних кодів, які стоять за програмним забезпеченням, щоб постійно надавати продуктивні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, має ACID-сумісність з 2001 року і має потужні доповнення, такі як популярний розширювач геопросторової бази даних PostGIS. Не дивно, що PostgreSQL став реляційною базою даних з відкритим кодом, яку вибирають багато людей та організацій.

PostgreSQL постачається з багатьма функціями, які допомагають розробникам створювати програми, адміністраторам захищати цілісність даних і створювати відмовостійкі середовища, а також допомагають вам керувати даними незалежно від того, наскільки великий чи малий набір даних. Окрім того, що PostgreSQL є безкоштовним і з відкритим вихідним кодом, він дуже розширюваний. Наприклад, надається можливість визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування без перекомпіляції бази даних.

PostgreSQL намагається відповідати стандарту SQL, якщо така відповідність не суперечить традиційним функціям або може призвести до неправильних архітектурних рішень. Багато функцій, необхідних стандартом SQL,

підтримуються, хоча іноді вони мають дещо відмінний синтаксис або функції. З часом можна очікувати подальших кроків у напрямку відповідності. Починаючи з версії 14 у вересні 2021 року, PostgreSQL відповідає щонайменше 170 із 179 обов'язкових функцій для відповідності SQL:2016 Core. На момент написання цієї статті жодна реляційна база даних не відповідає повній відповідності цьому стандарту.

Нижче наведено невичерпний список різноманітних функцій, які можна знайти в PostgreSQL.

Таблиця 2.2 – функції та можливості PostgreSQL.

Функція	**Опис**
Типи даних	– примітиви: Integer, Numeric, String, Boolean; – структури: Date/Time, Array, Range / Multirange, UUID; – документи: JSON/JSONB, XML, Key-value (Hstore); – геометрія: Point, Line, Circle, Polygon; – кастомізація: Composite, Custom Types.
Цілісність даних	– UNIQUE, NOT NULL; – первинний ключ; – зовнішні ключі; – обмеження виключення; – явні блокування, рекомендаційні блокування.
Надійність, аварійне відновлення	– попереднє ведення журналу (WAL); – реплікація: асинхронна, синхронна, логічна; – відновлення на момент часу (PITR), активний режим очікування;

Кінець таблиці 2.2.

Паралелізм, продуктивність	– індексування: B-дерево, багатостовпцеве, вирази, часткове; – розширене індексування: GiST, SP-Gist, KNN Gist, GIN, BRIN, індекси покриття, фільтри Блума; – складний планувальник/оптимізатор запитів, сканування лише для індексів, багатоколонкова статистика; – транзакції, вкладені транзакції (через точки збереження); – багатоверсійний контроль паралельності (MVCC); – паралелізація запитів на читання та побудова індексів B-дерева; – розбиття таблиць; – всі рівні ізоляції транзакцій, визначені в стандарті SQL, включаючи Serializable; – компіляція виразів JIT.
Безпека	– аутентифікація: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, сертифікат тощо; – надійна система контролю доступу; – безпека стовпців і рядків; – багатофакторна аутентифікація за допомогою сертифікатів.
Інтернаціоналізація, текстовий пошук	– підтримка міжнародних наборів символів; – зіставлення без урахування регістру та акценту; – повнотекстовий пошук; – міжтабличний пошук.

Після вибору бази даних слід проаналізувати серверну частину програми. На даному етапі розглянемо одразу дві технології: базову та незамінну Node.js, а також надсучасний та зручний Nest.js. Обидві технології мають підтримку мов програмування JavaScript та TypeScript, які варто обирати виключно виходячи з вимог та специфіки проєкту. Для розробки інформаційної системи підтримки проведення змагань з плавання було обрано мову програмування TypeScript, адже вона є суворо типізованою, та може виявити помилки ще на етапі розробки, що є дуже цінним показником при розробці серверу, якій взаємодіє з базою даних. В іншому випадку, можуть мати місця необроблені помилки, які можуть прямо завдати шкоди даним, тому вибір TypeScript є більше ніж виправданий.

Node.js — це міжплатформене середовище виконання JavaScript з відкритим вихідним кодом. Node.js запускає движок JavaScript V8, ядро Google Chrome, поза браузером. Це дозволяє технології бути дуже продуктивним.

Програма Node.js працює в одному процесі, не створюючи новий потік для кожного запиту. Node.js містить у своїй стандартній бібліотеці набір асинхронних примітивів введення-виводу, які запобігають блокування коду JavaScript, і загалом бібліотеки в Node.js написані з використанням неблокуючих парадигм, що робить поведінку блокування скоріше винятком, ніж нормою. Коли Node.js виконує операцію введення-виведення, наприклад, читання з мережі, доступ до бази даних або файлової системи, замість того, щоб блокувати потік і витрачати цикли ЦП на очікування, Node.js відновить операції, коли відповідь повернеться. Це дозволяє Node.js обробляти тисячі одночасних підключень з одним сервером, не вводячи тягар керування паралельністю потоків, що може бути значним джерелом помилок.

Node.js має унікальну перевагу, оскільки мільйони розробників інтерфейсу, які пишуть JavaScript для браузера, тепер можуть писати код на стороні сервера на застосунок до коду на стороні клієнта без необхідності вивчати зовсім іншу мову.

У Node.js нові стандарти ECMAScript можна використовувати без проблем, оскільки не потрібно чекати, поки всі користувачі оновлять свої браузери – розробники самі вирішують, яку версію ECMAScript використовувати.

Nest – це відносно новий фреймворк Node.js, який не тільки імітує, але й виправляє недоліки попередніх фреймворків.

При старті планування розробки програмного забезпечення за допомогою Node.js, NestJS є набагато кращим вибором, ніж ExpressJS, оскільки він побудований на чіткому дизайні з кількома простими компонентами (контролерами, модулями та провайдерами). Це робить поділ програм на мікросервіси досить легким.

Слід розглянути архітектури Nest.js задля більш широкого розуміння побудови системи:

– контролери несуть відповідальність за обробку вхідних запитів і реагування на клієнтську сторону програми;

– модулі використовуються для структурування коду та розділення функціональних можливостей на логічні шматки, які можна повторно використовувати;

– провайдери або сервіси, які абстрагують складність і логіку від користувача. Можна впровадити сервіс в контролери або інші сервіси.

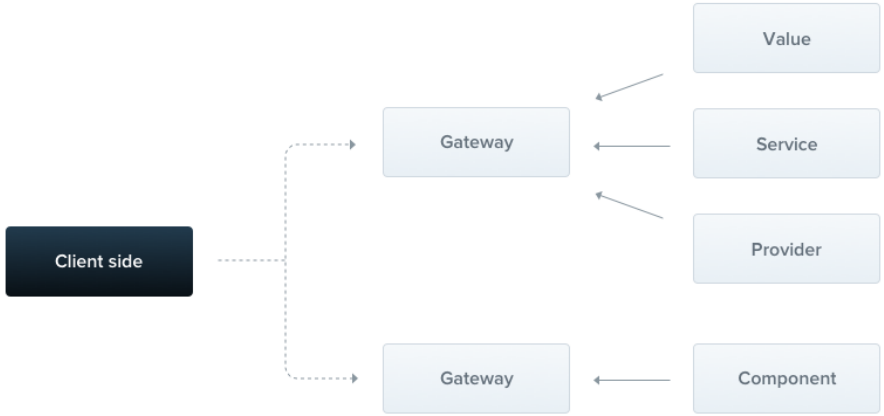

Рисунок 2.2 – Візуалізація архітектури Nest.js

Серед безсумнівних переваг Nest.js виділяють одразу декілька пунктів:

– Nest.js був створений, щоб допомогти розробникам створювати моноліти та мікросервіси;

– він простий у використанні, швидкий в освоєнні та легкий у застосуванні;

– фреймворк використовує TypeScript — строго типізовану мову, яка є наднабором JavaScript;

– потужний інтерфейс командного рядка (CLI) для підвищення продуктивності та простоти розробки;

– підтримка десятків специфічних модулів, які допоможуть легко інтегруватися із звичайними технологіями та концепціями, такими як Type ORM, Mongoose, GraphQL, Logging, Validation, Caching, WebSockets та багато інших;

– прості програми модульного тестування;

– чудова документація;

– створено для великомасштабних корпоративних застосунків.

Після визначення усіх пунктів стосовно серверної частини варто звертати увагу на наступний етап планування розробки та вибору технології. Цього разу розглянемо клієнтську частину та найрозповсюдженішу, найпопулярнішу технологію у даному напрямку – React.

React – це бібліотека JavaScript для створення інтерфейсів користувача. React дозволяє безболісно створювати інтерактивні інтерфейси. Створені прості представлення для кожного стану у програмі React ефективно оновлюватиме й відображатиме потрібні компоненти, коли дані змінюються. Декларативні уявлення роблять код більш передбачуваним і легшим для налагодження. React надає можливість створювати інкапсульовані компоненти, які керують своїм власним станом, які в свою чергу можливо застосувати у плануванні складних інтерфейсів.

Оскільки логіка компонентів написана на JavaScript, а не на шаблонах, розробники можуть легко передавати розширені дані через свою програму і зберігати стан поза межами DOM.

Останньою серед найбільш масштабних технологій використовуємо npm.

Npm – найбільший у світі реєстр програмного забезпечення. Розробники з відкритим кодом з усіх континентів використовують npm для спільного використання та запозичування пакетів, і багато організацій також використовують npm для управління приватною розробкою.

Npm складається з трьох окремих компонентів:

– вебсайт;

– інтерфейс командного рядка (CLI);

– реєстр.

Вебсайт використовується для того, щоб знаходити пакети, налаштовувати профілі та керувати іншими аспектами роботи з npm. Наприклад, розробники можуть налаштувати організації для керування доступом до загальнодоступних або приватних пакетів.

CLI працює з терміналу, і саме так більшість розробників взаємодіє з npm.

Реєстр є великою загальнодоступною базою даних програмного забезпечення JavaScript та метаінформації, що її оточує.

Npm слід використовувати для:

– адаптації пакетів коду для програм;

– завантаження окремих інструментів, які ви зможете користуватися відразу;

– запуску пакетів без завантаження за допомогою npx;

– поділіться кодом з будь-яким користувачем npm будь-де.

– обмежити код для певних розробників.

– Створіть організації для координації обслуговування пакетів, кодування та розробників.

Висновки до розділу 2

Правильне та послідовне виконання проєктного плану є основою будь-якого успіху. Вкрай важливо слідувати методологіям, правилам, принципам розробки задля того, щоб якнайшвидше отримати бажаний результат.

Модель розробки програмного забезпечення — це потужний інструмент, який допомагає організувати роботу як великої команди, так і інді-розробника. Оскільки навколишній світ розвивається, з'являються нові технології, які спонукають до змін, в тому числі і в інформаційних технологіях, то досить важливо тримати та розвивати проєкти сучасними методами та моделями розробки настільки, наскільки це є можливим. Адже сам послідовне виконання плану розробки вкладає надзвичайно вагомий внесок у глобальний успіх.

Було створено досить значиме число різноманітних моделей розробки та методологій для того, аби власники проєкту та окремі працівники мали можливість повноцінно, безперебійно досягати поставленої мети у значно швидших часових інтервалах, зокрема для продуктів, які обмежені в питаннях коштів та часу. Саме тому було розглянуто вісім найуспішніших та найрозповсюдженіших моделей серед світу розробки проєктів. До даного переліку потрапили наступні моделі: Agile, Waterfall, V Model, RAD Model, Incremental Model, Iterative Model, Spiral Model, та Prototyping Model.

Порівнюючи специфіку проєкту та функціональність кожної з названих моделей, виникла необхідність проаналізувати переваги та недоліки, та з'ясувати найбільш зручну модель для розробки, місце якої посіла Agile Model.

Наступним важливим кроком став вибір архітектури. Внаслідок аналізу вимог та мети проєкту було обрано клієнт-серверну архітектуру, яка повністю задовільняє усі необхідні важелі для виконання поставлених цілей та завдань. Суттєвою перевагою даної архітектури є можливість будування вебплатформи, доступ до якої буде здійснюватись, фактично, з усієї земної кулі, де можливо тільки скористатися мережею інтернет.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Загальні методи роботи з TypeORM

Одним з найголовніших, та водночас одним з найважчих завдань у створенні інформаційної системи є розробки та налаштування серверної частини зі зв'язком з базою даних. Це саме той процес, на якому не варто заощаджувати час або кошти, адже він є фундаментальним у будівництві системи та безпеки. Основним інструментом «комунікації» з базою даних у коді є об'єктно-реляційне відображення, за допомогою якого відбувається конвертація даних між технологіями. Більш детальними словами, об'єктно-реляційне відображення(ORM) – це техніка програмування для перетворення даних між системами типів за допомогою об'єктно-орієнтованих мов програмування. Це створює, по суті, «базу даних віртуальних об'єктів», яку можна використовувати за допомогою мови програмування.

Для створення інформаційної системи підтримки проведення змагань з плавання було обрано технологію TypeORM. TypeORM – це ORM, яке може працювати на платформах NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo та Electron і може використовуватися з TypeScript та JavaScript (ES5, ES6, ES7, ES8). Його мета – завжди підтримувати найновіші функції JavaScript та надавати додаткові функції, які допоможуть вам розробляти будь-які програми, які використовують бази даних – від невеликих програм з кількома таблицями до великомасштабних корпоративних програм із кількома базами даних.

TypeORM підтримує шаблони Active Record і Data Mapper, на відміну від усіх інших JavaScript ORM, які зараз існують, що означає, що надається змога писати високоякісні, слабо пов'язані, масштабовані та підтримувані програми найпродуктивнішим способом. На TypeORM сильно впливають інші ORM, такі як Hibernate, Doctrine і Entity Framework.

Основні переваги TypeORM:

- підтримує як DataMapper, так і ActiveRecord (на вибір користувача);
- сутності та стовпці;
- типи стовпців для бази даних;
- менеджер суб'єкта;
- репозиторії та користувацькі сховища;
- реляційна модель чистого об'єкта;
- асоціації (відносини);
- завзяті та ледачі відношення;
- односпрямовані, двонаправлені та зв’язки, що посилаються на себе;
- підтримує кілька моделей успадкування;
- каскади;
- індекси;
- транзакції;
- міграції та автоматичне генерування міграцій;
- пул підключень;
- реплікація;
- використання кількох екземплярів бази даних;
- робота з багатьма типами баз даних;
- запити між базами даних і між схемами;
- елегантний синтаксис, гнучкий і потужний QueryBuilder;
- ліві та внутрішні з’єднання;
- правильне розбиття на сторінки для запитів із використанням об’єднань;
- кешування запитів;
- потокове передавання необроблених результатів;
- заготівля лісу;
- слухачі та передплатники (гачки);

– підтримує шаблон закриття таблиці;
– оголошення схеми в моделях або окремих файлах конфігурації;
– конфігурація підключення у форматах json / xml / yml / env;
– підтримує MySQL / MariaDB / Postgres / CockroachDB / SQLite / Microsoft SQL Server / Oracle / SAP Hana / sql.js;
– підтримує базу даних MongoDB NoSQL;
– працює на платформах NodeJS / Browser / Ionic / Cordova / React Native / NativeScript / Expo / Electron;
– підтримка TypeScript і JavaScript;
– підтримка ESM і CommonJS;
– створений код є продуктивним, гнучким, чистим і придатним для обслуговування;
– дотримується всіх можливих найкращих практик;
– CLI.

Для того, щоб встановити TypeORM, потрібно виконати команду

```
npm install typeorm --save
```

Після встановлення пакету одразу відкривається можливість використати клас DataSource для запису конфігурацій підключення. Даний клас має наступні властивості:

– type – тип підключення;
– host – хост підключення;
– port – порт підключення;
– username – ім’я творця бази даних;
– password – пароль творця бази даних;
– database – ім’я бази даних;
– synchronize – чи ввімкнена синхронізація таблиць;
– logging – чи показувати логи в системі;
– entities – список всіх сутностей;

– migrations – список всіх міграцій.

Базове налаштування закінчене, програма має успішно запуститися і підтримати зв’язок з базою даних. Наступним кроком потрібно сформувати модель та функції редагування записів бази даних.

Entity — це модель, оформлена декоратором *@Entity*. Для таких моделей буде створена таблиця бази даних. Співпраця з сутностями приходить скрізь у TypeORM. Дані моделі підтримують завантаження, вставки, оновлення, видалення та виконання інші операції з ними.

Щоб додати стовпці бази даних, просто потрібно прикрасити властивості сутності, які потрібно перетворити в стовпець, за допомогою декоратора *@Column.*

Кожна сутність повинна мати принаймні один стовпець первинного ключа. Це вимога, яку неможливо оминути. Щоб зробити стовпець первинним ключем, потрібно використовувати декоратор *@PrimaryColumn*.

Для того, щоб стовпець ідентифікатора генерувався автоматично, потрібно змінити декоратор @PrimaryColumn на декоратор *@PrimaryGeneratedColumn*.

Загальні DataSource опції:

– type – тип СКБД. Необхідно вказати, який механізм баз даних використовується. Можливі значення: "mysql", "postgres", "cockroachdb", "sap", "spanner", "mariadb", "sqlite", "cordova", "react-native", "nativescript", "sqljs", "oracle", "mssql", "mongodb", "aurora-mysql", "aurora-postgres", "expo", "better-sqlite3", "capacitor". Цей параметр є обов’язковим;

– extra – додаткові параметри, які будуть передані базовому драйверу. Використовується, якщо є мета передати додаткові параметри базовому драйверу бази даних;

– entities – Entities або схеми Entity, які будуть завантажені та використані для цього джерела даних. Приймає як класи сутностей, класи схеми сутностей, так і шляхи до каталогів для завантаження. Каталоги підтримують шаблони glob. Приклад: сутності: [Post, Category, "entity/*.js", "modules/**/entity/*.js"];

– subscribers – підписники, які будуть завантажені та використані для цього джерела даних. Приймають як класи сутностей, так і каталоги для завантаження. Каталоги підтримують шаблони glob. Приклад підписників: [PostSubscriber, AppSubscriber, "subscriber/*.js", "modules/**/subscriber/*.js"];

– migrations – міграції, які будуть завантажені та використані для цього джерела даних. Приймають як класи міграції, так і каталоги для завантаження. Каталоги підтримують шаблони glob. Приклад: міграції: [FirstMigration,"migration/*.js", "modules/**/migration/*.js"];

– logging – вказує, чи увімкнено ведення журналу чи ні. Якщо встановлено значення true, буде ввімкнено реєстрацію запитів і помилок. Також можливо вказати різні типи журналів, які потрібно ввімкнути, наприклад ["query", "error", "schema"];

– logger – функція, яка буде використовуватися для ведення журналів. Можливі значення: «advanced-console», «simple-console» і «file». За замовчуванням – «advanced-console»;

– maxQueryExecutionTime – якщо час виконання запиту перевищує вказаний максимальний час виконання (у мілісекундах), реєстратор зареєструє цей запит;

– namingStrategy – стратегія іменування, яка буде використовуватися для іменування таблиць і стовпців у базі даних;

– entityPrefix – префікси з заданим рядком усіх таблиць (або колекцій) у цьому джерелі даних;

– entitySkipConstructor – вказує, чи слід TypeORM пропускати конструктори під час десеріалізації сутностей з бази даних. Якщо не викликається конструктор, як приватні властивості, так і властивості за замовчуванням не працюватимуть належним чином;

– dropSchema – скидає схему щоразу, коли джерело даних ініціалізується. Цей параметр корисний під час налагодження та розробки;

– synchronize – вказує, чи слід автоматично створювати схему бази даних при кожному запуску програми. Цей параметр корисний під час налагодження та розробки;

– migrationsRun – вказує, чи слід міграції запускати автоматично при кожному запуску програми. Як альтернатива, є можливим використання CLI та запуску команди migration:run;

– migrationsTableName – ім'я таблиці в базі даних, яка буде містити інформацію про виконані міграції. За замовчуванням ця таблиця називається «migrations»;

– migrationsTransactionMode – контроль транзакцій для міграцій (за замовчуванням: all);

– metadataTableName – ім'я таблиці в базі даних, яка буде містити інформацію про метадані таблиці. За замовчуванням ця таблиця називається "typeorm_metadata";

– cache – вмикає кешування результатів об’єктів;

– cli.entitiesDir – каталог, де сутності повинні бути створені за замовчуванням CLI;

– cli.subscribersDir – каталог, де абоненти повинні бути створені за замовчуванням CLI.

Отже, TypeORM є надпотужним інструментом, який допоможе повністю встановити контроль над взаємодією з базою даних, та уникнути непередбачених помилок. Простота використання TypeORM виключає необхідність спеціалізованого досвіду, а CLI дозволяє швидко реагувати на будь-які потреби, які необхідно виконати згідно з вимогами завдань. Саме тому цей вид ORM захоплює думки розробників, як одна з найзручніших систем, що виконує низку необхідних процесів.

3.2 Загальні методи роботи з Ant Design

Розробка будь-якої системи передбачає створення візуального інтерфейсу, який буде поєднувати потенційного користувача з доступними функціями програми. З розвитком індустрії розробки програмного забезпечення постійно оновлюються доступні бібліотеки, які можуть бути використані в різних програмах, в залежності від переліку вимог. Здебільшого, бібліотеки візуальних компонентів можуть бути кастомізовані та підібрані під стиль проєкту, тому вибір серед UI бібліотек базується в основному на кількості та якості доступних функцій. Однією з найбільш поширених та розвинених бібліотек компонентів для JavaScript фрейморку React є Ant Design.

Ant Design – це UI бібліотека, що має багато простих у використанні компонентів, які корисні для створення елегантних інтерфейсів користувача. Завдяки Ant Design можна розробити досить складний інтерфейс з використанням практично всіх візуальних компонентів, які наразі існують. При створенні інформаційної системи підтримки проведення змагань з плавання було використано велику кількість компонентів, що значно прискорило швидкість розробки, а також збільшило рівень розуміння системи.

Основним та найрозповсюдженим елементом користувацького інтерфейсу є звичайна кнопка (Ant Design element – **Button**). Сама сутність кнопки означає операцію, або декілька операцій, виконання яких зумовлює виконання частини бізнес-логіки, яка попередньо запрограмована. Ant Design пропонує п’ять типів кнопок:

– *primary button*: вказує головну дію, передбачає не більше однієї основної кнопки в одному розділі;

– *default button*: вказує на серію дій без пріоритету;

– *dashed button*: зазвичай використовується для додавання дії;

– *text button*: використовується для найбільш другорядної дії;

– *link button*: використовується для зовнішніх посилань.

Також Button має чотири додаткові властивості, від яких кнопка змінює стилі:

– *danger*: використовується для дій ризику, як-от видалення чи авторизація;

– *ghost*: використовується в ситуаціях зі складним фоном, зазвичай домашніми сторінками;

– *disabled*: коли дії недоступні;

– *loading*: додає спінер завантаження в кнопку, уникаючи також багаторазового надсилання.

Наступним компонентом, що значно впливає на візуальне уявлення програми є значок (Ant Design element – **Icon**). Сам компонент дозволяє виконувати дві функції: завантаження зовнішнього значка та використання значків бібліотеки. Бібліотеки налічує дуже велику кількість особистих значків, та розрізняє їх за наступними видами:

– *Outlined*: значки, які не розфарбовані кольорами;

– *Filled*: значки, які зафарбовані кольорами;

– *Two tone:* значки, частини яких не розфарбовані, а частини зафарбовані кольорами.

Також слід розрізняти значки згідно різних категорій, які представлені спеціально під вимоги бізнес-потреб:

– *Directional*: значки напрямку;

– *Suggested*: значки пропозицій;

– *Editor*: значки редактору;

– *Data*: значки даних;

– *Brand and Logos*: значки відомих брендів;

– *Application*: загальні значки програми.

Загалом компоненти Ant Design можна представити у вигляді однієї таблиці, що розділяє запис на три блоки: компонент, його значення, та його застосування.

Таблиця 3.1 – Опис компонентів Ant Design.

Компонент	**Значення**	**Коли використовується**
Divider	Розділ блоків лінією	– коли є потреба відокремити певні частини тексту, або іншого контекту.
Grid	Система розділу області	– коли необхідність в вирішення багатьох проблем зі збереженням інформації в області проєктування.
Cascader	Каскадне поле вибору	– коли потрібно вибрати з набору пов’язаних даних. Наприклад, провінція/місто/район, рівень компанії, класифікація речей; – під час вибору з великого набору даних із роздільною багатоетапною класифікацією для легкого вибору; – вибір каскадних елементів в одному плаваючому шарі для кращого користування.
AutoComplete	Автозаповнення поля введення	– коли потрібно поле введення замість селектора; – коли потрібні пропозиції або довідковий текст.

Продовження таблиці 3.1.

Select	Розкривне меню для відображення варіантів	– коли необхідно обрати варіант з-поміж запропонованих.
Menu	Меню навігації	– верхня навігація містить усі категорії та функції вебсайту; – бічна навігація забезпечує багаторівневу структуру вебсайту.
Steps	Панель навігації, яка веде користувачів через кроки завдання.	– коли завдання є складним або має певну послідовність у серії підзадач.
Dropdown	Спадне меню вибору	– коли необхідно обрати варіант з-поміж запропонованих
Affix	Закріплення контенту на сторінці	– якщо зустрічається в програмі різноманітні меню або дії.
Breadcrumb	Відображає поточне розташування користувача в ієрархії.	– коли система має більше двох рівнів в ієрархії; – коли потрібно повідомити користувача про те, де він знаходиться; – коли користувачеві може знадобитися повернутися на вищий рівень.
Checkbox	Прапорець вибору	– використовується для вибору кількох значень із кількох параметрів;
DatePicker	Введення дати	– коли необхідно обрати дату.

Продовження таблиці 3.1.

Form	Високопродуктивний компонент форми з керуванням обсягом даних. Включає в собі збір даних, перевірку та стилі.	– коли вам потрібно створити екземпляр або зібрати інформацію; – коли потрібно перевірити поля в певних правилах.
Input	Основним віджетом для отримання введених користувачами даних є текстове поле.	– потрібен введення користувача в поле форми; – необхідно ввести пошуковий запит.
InputNumber	введення числа в певному діапазоні за допомогою миші або клавіатури.	– коли потрібно надати числове значення.
Mentions	Компонент, що дозволяє згадати певну сутність.	– коли потрібно згадати когось або щось.
Radio	Радіо кнопка	– використовується для вибору одного стану з кількох варіантів; – відмінність від Select полягає в тому, що радіо видиме для користувача і може полегшити порівняння вибору, а це означає, що їх не повинно бути занадто багато.
Rate	Оцінка	– операція швидкої оцінки чогось.

Продовження таблиці 3.1.

Switch	Селектор перемикання	– якщо потрібно представити перемикання між двома станами або станом увімкнення-вимкнення; – різниця між Switch та Checkbox полягає в тому, що Switch ініціює зміну стану безпосередньо, коли перемикається, тоді як Checkbox зазвичай використовується для позначення стану, що має працювати разом з операцією надсилання.
TimePicker	Введення часу	– коли необхідно ввести час.
Transfer	Поле вибору перенесення подвійних стовпців.	– по суті, це елемент керування вибором, який можна використовувати для вибору кількох елементів. – Transfer може відображати більше інформації про елементи та займати більше місця.
TreeSelect	Контроль вибору у вигляді дерева.	– TreeSelect подібний до Select, але значення надаються у деревоподібній структурі. Будь-які дані, записи яких визначені ієрархічним чином, підходять для використання цього елемента керування.

Продовження таблиці 3.1.

Slider	Повзунок. Компонент для відображення поточного значення та інтервалів у діапазоні.	– щоб ввести значення в діапазоні.
Upload	Завантажує файли, підтримує вибір та перетягування.	– коли потрібно завантажити один або кілька файлів; – коли потрібно показати процес завантаження; – коли потрібно завантажити файли шляхом перетягування.
Avatar	Фото	– аватари можна використовувати для зображення людей або об’єктів. Компонент підтримує зображення, значки або букви.
Badge	Невелике числове значення або дескриптор стану для елементів інтерфейсу користувача.	– значок зазвичай з’являється поблизу сповіщень або аватарів користувачів, зазвичай показуючи кількість повідомлень.
Calendar	Контейнер для відображення даних у вигляді календаря.	– коли дані представлені у формі дат, таких як розклади, календар цін, місячний календар тощо. Цей компонент також підтримує перемикання рік/місяць.
Card	Простий прямокутний контейнер.	– картку можна використовувати для відображення вмісту, пов’язаного з однією темою.

Кінець таблиці 3.1.

Carousel	Компонент каруселі. Перемикач видів з контейнером.	– коли на одному рівні є група вмісту; – при недостатньому вмісті простору його можна використовувати для економії місця у вигляді обертових дверей; – зазвичай використовується для групи малюнків/карток.
Table	Таблиця. Відображає дані у рядках.	– щоб відобразити набір структурованих даних; – для сортування, пошуку, розбиття на сторінки, фільтрації даних.
Modal	Модальні діалоги.	– надати можливість для користувачів на додаткову взаємодію з програмою, але без переходу на нову сторінку та переривання робочого процесу користувача. – використовується на поточній сторінці, щоб отримати відгуки користувачів або відобразити інформацію.

Бібліотека Ant Design має багато корисного функціоналу, використання якого суттєво економить час в розробці, паралельно надаючи можливості рендеру візуально зручних та гарних компонентів, які своєю сукупністю дарують користувачеві приємний інтерфейс.

Висновки до розділу 3

Кожен проєкт починає своє життя з різноманітних процесів, таких як: створення переліку вимог, побажань клієнтів, команди працівників різних позицій та рівнів, планового опису тощо. Варто зазначити, що більшість вищенаведених процесів охоплюються переважно позиціями project manager, product manager, або ж project coordinator. Скоуп завдань для розробників здебільшого відрізняється присутністю технічних аспектів проєкту, першим з яких проявляється вибір технологій. Успішне застосування обраних технологій грає важливу роль в глобальному успіху проєкту, адже можливості масштабування залежать від можливості інструментів розробки, а інструменти розробки залежать від вимог проєкту.

Плануючи розробку серверу, необхідно добре усвідомити як саме будується архітектура та взаємодія з базою даних. Надійна TypeORM надійно виконує весь спектр вимог, які мають в обов'язковому форматі виконуватись. Простота та універсальність TypeORM допомагає швидко взаємодіяти з базою даних. Питання створення, читання, редагування, або видалення записів бази даних вирішене завдяки запропонованої функціональності обраної ORM.

Не менш важливим за взаємодію з базою даних є процес взаємодії з кінцевим користувачем, коли необхідно візуально надати можливість користуватися усім доступним спектром функцій, які запрограмовані на сторонах серверу та клієнту. UI Бібліотека Ant Design чудово запропонувала себе у якості вирішення складних питань з простим користувацьким інтерфейсом, що наповнений зручним переліком компонентів.

У результаті аналізу та тестування системи сформувався висновок стосовно працездатності вищеназваних технологій, який розповідає, що UI бібліотека Ant Design та ORM TypeORM чудово та безперебійно виконують поставлені задачі, а саме зв'язують користувача з системою, та систему з базою даних, отже попередньо поставлена ціль – виконана.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОКУМЕНТАЦІЇ

Для реалізації інформаційної системи була обрана клієнт-серверна архітектура з технологіями React на клієнті та Nest на сервері. У даних технологій велика кількість переваг, завдяки яким багато розробників вибирають саме ці фреймворки. Простота - перша технічна їхня перевага. Вони мають чіткі синтаксичні правила і зрозумілу семантику. Раціональність і стислість дуже корисні для обробки коду машинами з обмеженим обсягом ресурсів. Обрання середовища програмування VS Code обумовлене досвідом програмування у цьому середовищі, його стабільною роботою та наявністю вбудованих допоміжних засобів, модулів та доступності додаткових розширень.

4.1 Опис середовища розробки

Visual Studio Code – це переозначений та оптимізований редактор коду для створення та налагодження сучасних веб- та хмарних застосунків. VS Code поєднує в собі простоту редактора вихідного коду з потужними інструментами розробника, такими як завершення та налагодження коду IntelliSense.

Перш за все, це редактор, який допомагаю впродовж всього шляху розробки. Дивовижний цикл редагування-складання-налагодження означає менше часу на те, щоб налагоджувати середовище, і більше часу на виконання ваших ідей.

В основі Visual Studio Code є блискавичний редактор вихідного коду, ідеальний для повсякденного використання. Завдяки підтримці сотень мов VS Code допомагає миттєво працювати з підсвічуванням синтаксису, узгодженням дужок, автоматичним відступом, виділенням поля, фрагментами тощо. Інтуїтивно зрозумілі комбінації клавіш, легка настройка та зіставлення комбінацій клавіш, створені спільнотою, дозволяють легко переміщатися по коду. Для серйозного кодування часто можна отримати вигоду від інструментів з більшим розумінням коду, ніж просто блоки тексту. Visual Studio Code включає вбудовану підтримку для завершення коду IntelliSense, розширене розуміння семантичного коду та

навігацію, а також рефакторинг коду. І коли кодування стає важким, важкі компоненти піддаються налагодженню. Налагодження часто є єдиною функцією, яку розробники найбільше пропускають при більш економному кодуванні. Visual Studio Code включає інтерактивний налагоджувач, тож можливо переходити через вихідний код, перевіряти змінні, переглядати стеки викликів та виконувати команди на консолі.

VS Code також інтегрується з інструментами збірки та сценаріїв, щоб виконувати звичайні завдання, що прискорює повсякденні робочі процеси. VS Code підтримує Git, тому доволі зручно працювати з керуванням кодом, не виходячи з редактора, включно з переглядом змін, що очікують на розгляд.

Безсумнівно, VS Code посів доволі гідне місце серед спільноти розробників за свої простоту, швидкість, зручність, та якість.

4.2 Опис програмної реалізації та огляд можливостей системи

Інформаційну систему підтримки проведення змагань з плавання було розроблено у якості вебзастосунку. Першим кроком користувач потрапляє на сторінку авторизації.

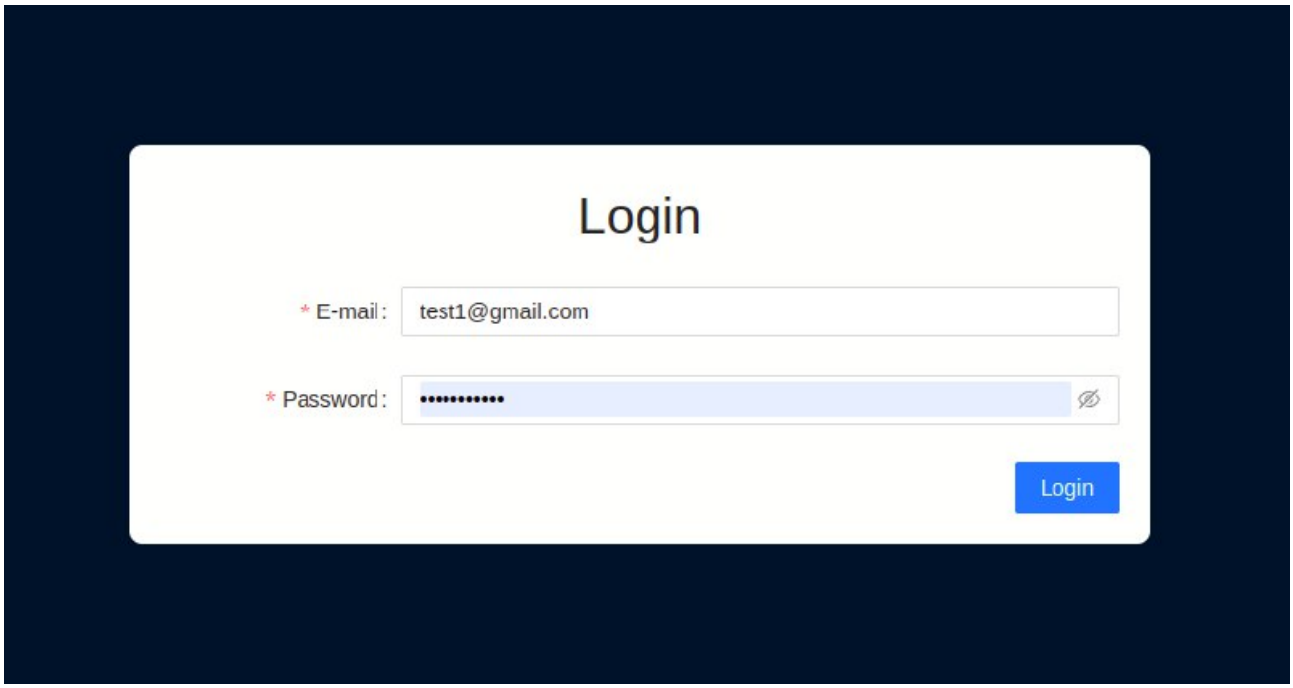

Рисунок 4.1 – Форма авторизації

Користувач має можливість лише авторизуватися в системі під ролями Coach, Admin, Mentor. Оскільки логікою розробки передбачено створення бодай одного акаунту адміністратора платформи, перші кроки управління в системі виконує саме адміністратор. Перед тим як успішно авторизуватися, перевіримо наявність валідації полів для вводу, а також відповідь від серверу у разі помилки.

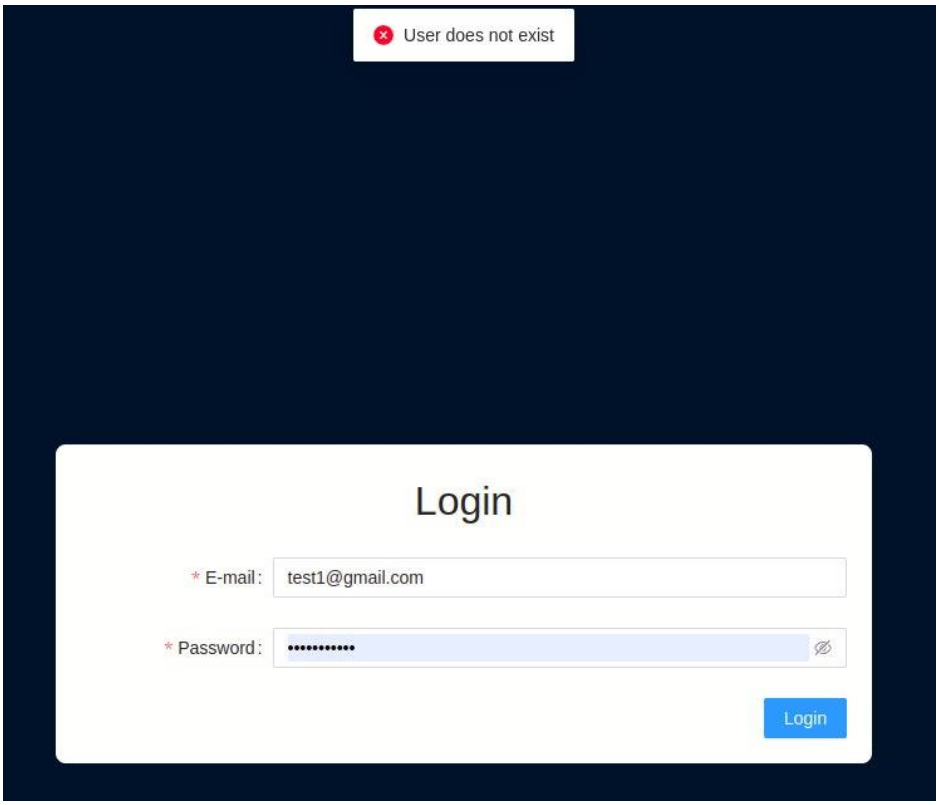

Рисунок 4.2 – Відповідь від серверу у разі помилки

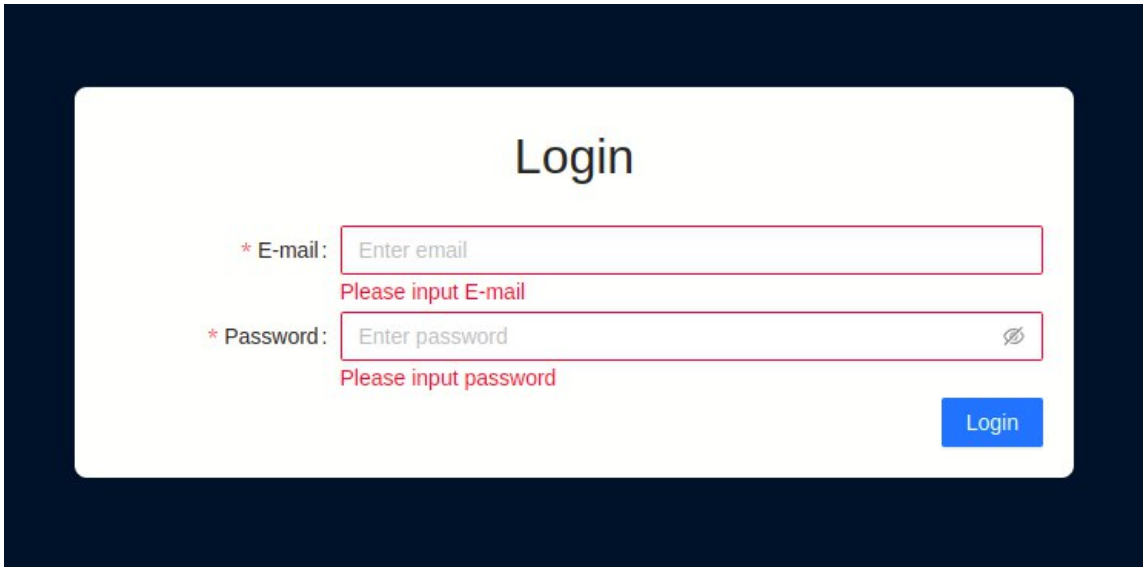

Рисунок 4.3 – Валідація полів для вводу

Після успішної авторизації адміністратор потрапляє до адмін-панелі, де має можливість користуватися доступною за вимогами бізнес-логіки навігацією.

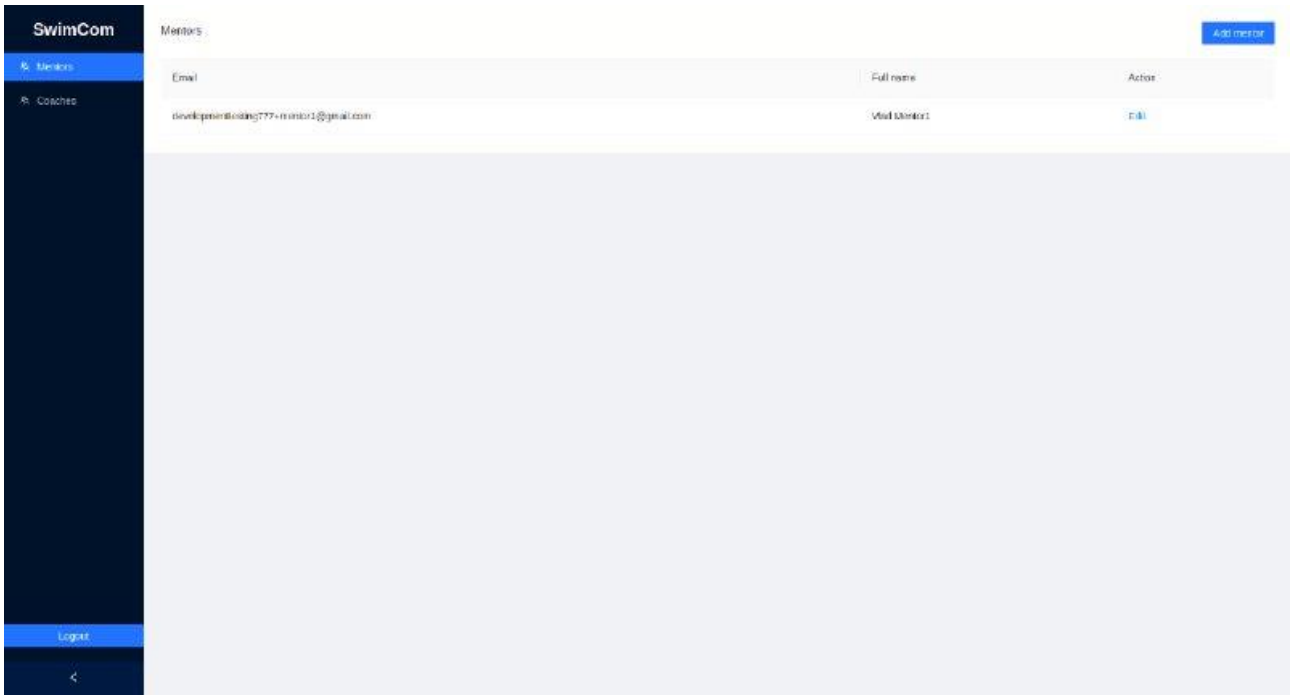

Рисунок 4.4 – Сторінка списку менторів

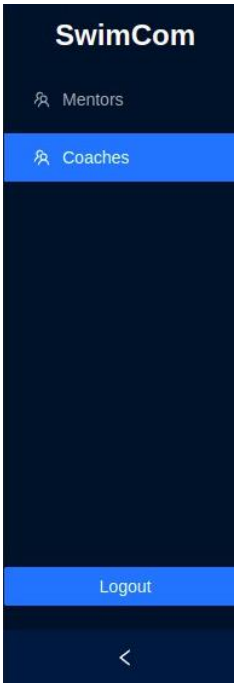

Рисунок 4.5 – Навігаційне меню адміністратора

Адміністратор має можливість створювати, переглядати, редагувати, та видаляти користувачів. Саме ці дії закладені до ролі адміністратора.

Рисунок 4.6 – Список менторів

Створення користувачів з ролями Mentor та Coach означає створення організаторів та тренерів відповідно. Роль тренера обмежена лише можливістю створенням заяви від свого імені для змагань. Тим часом організатор має право збирати заяви, створювати єдиний протокол змагань, та керувати процесами проведення змагань. Саме організатор відповідає за стартові протоколи та кінцеві результати. Оскільки програма є внутрішньою та непублічною, то реєстрація користувачів можлива лише за допомогою адміністратора, головна мета якого – слідкувати за системою, та за необхідності вносити корективи до акаунтів користувачів. При створені акаунту, адміністратор заповнює поля пошти, пароля, та ім’я.

Рисунок 4.7 – Форма створення ментору

Після створення, акаунт з’являється в переліку всіх менторів.

Рисунок 4.8 – Оновлений список менторів

Абсолютно ідентична логіка працює так само і для тренерів:

Рисунок 4.9 – Оновлений список тренерів

Адміністратор також має можливість редагувати та видаляти користувачів, попередньо натиснувши на кнопку Edit.

Рисунок 4.10 – Форма редагування користувачів

Наразі дії адміністратора обмежуються таким чином, адже за іншу бізнес-логіку відповідають ролі coach та mentor. Виконаємо вихід з системи, натиснувши на кнопку Logout.

Розглянемо наступну роль – ментора. Роль ментора є найбільш функціональною у порівнянні з іншими ролями. Авторизуючись, ментор потрапляє на першу сторінку навігації – Competitions, що відповідає за створення самої сутності змагань.

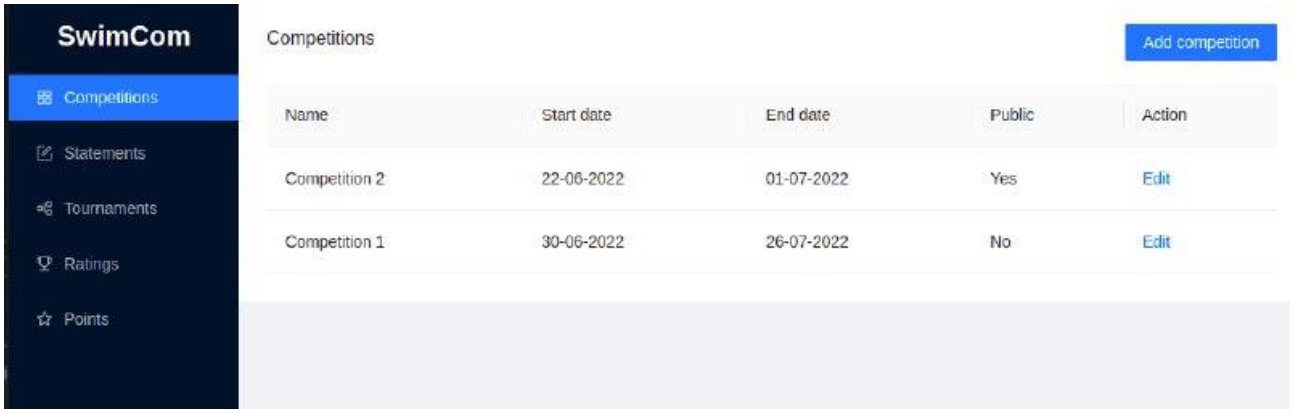

Рисунок 4.11 – Сторінка списку змагань

Створюючи змагання, ментор обирає прапор публічності. Якщо створене змагання є публічним, то тренери мають право надавати заяви на дане змагання, в іншому випадку – тренери не матимуть змоги створити заяву. Метою даної логіки є структуризація даних згідно змагань та зменшення кількості заяв, що не можуть бути застосовані, тобто іншими словами валідація сміття. Поле дати відповідає за номінальну дату проведення змагань. Це може бути як один день, так і декілька днів.

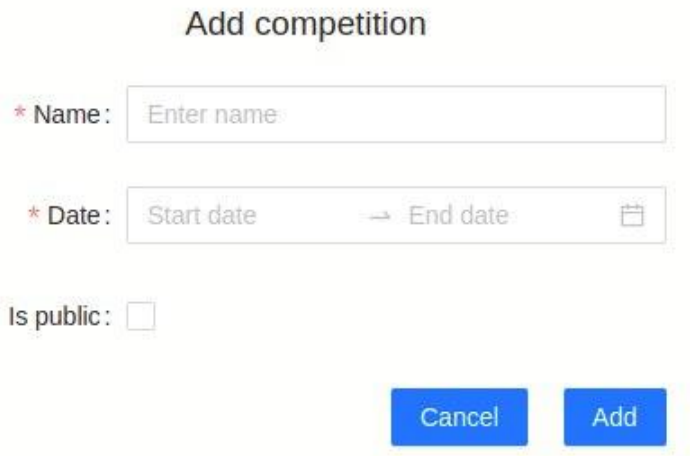

Рисунок 4.12 – Форма створення змагання

Після створення змагань, у тренерів та інших менторів з’являється можливість створити заяву саме під конкретні змагання.

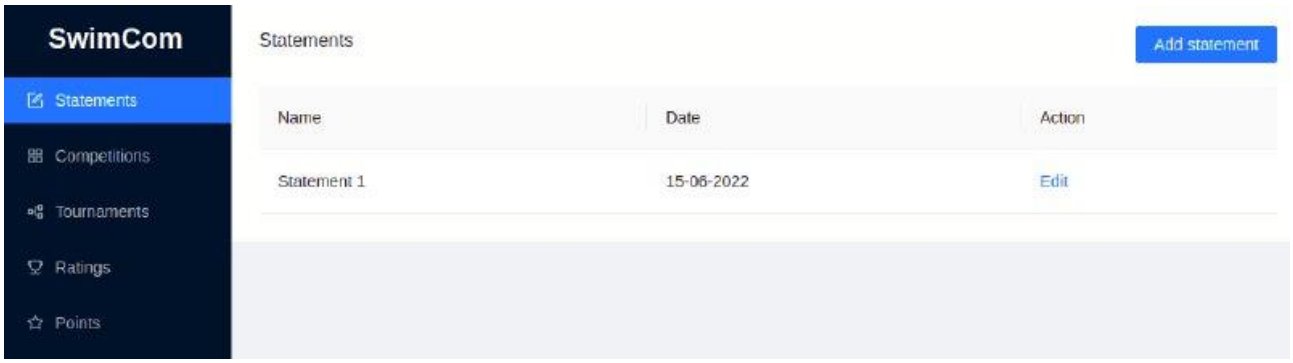

Рисунок 4.13 – Сторінка списку заяв

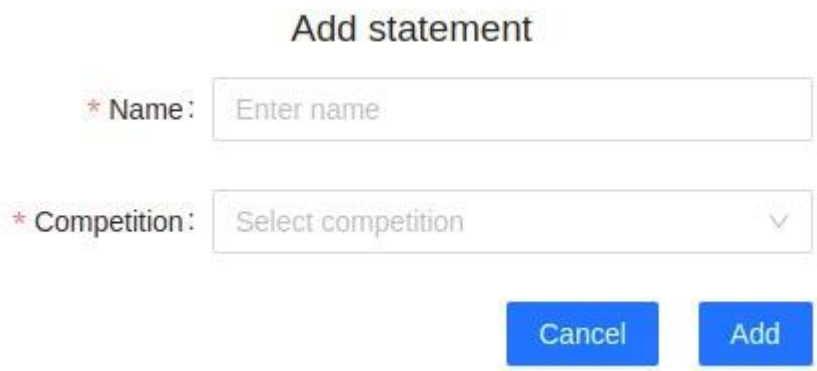

Рисунок 4.14 – Сторінка списку заяв

При створені заяви, користувач обирає до якого змагання дана заява має відношення.

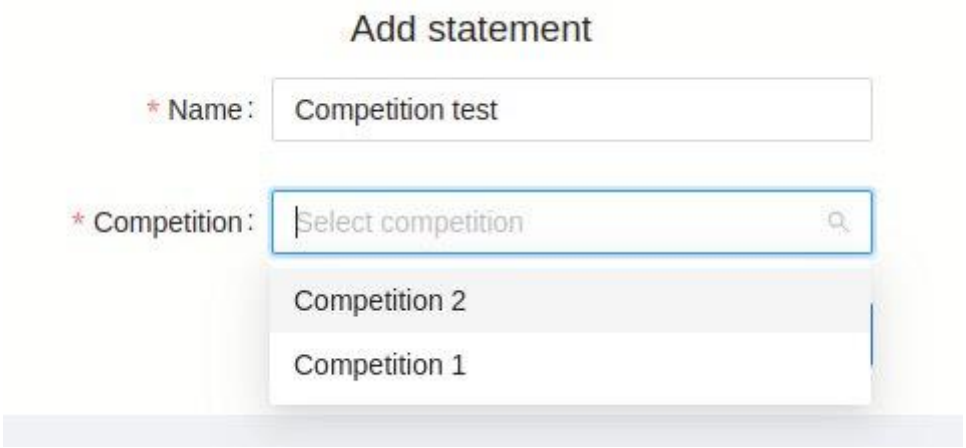

Рисунок 4.15 – Сторінка списку заяв

Після заповнення полів, натискаємо Add, після чого заява створюється. Перевіряємо оновлення списку заяв:

Рисунок 4.16 – Оновлений список заяв

У результаті успішного створення заяви, тренер має можливість додавати учасників змагань.

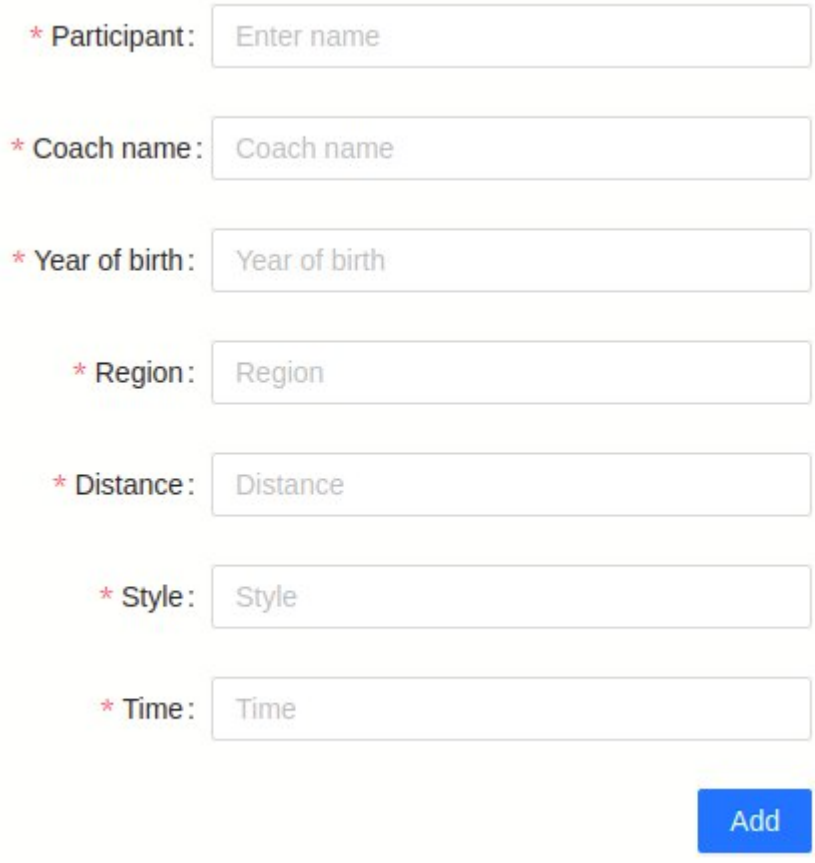

Рисунок 4.17 – Форма додавання учасника змагань

Тренер має можливість додати ім’я учасника, ім’я тренера, рік народження, регіон, а також інформацію про заплив.

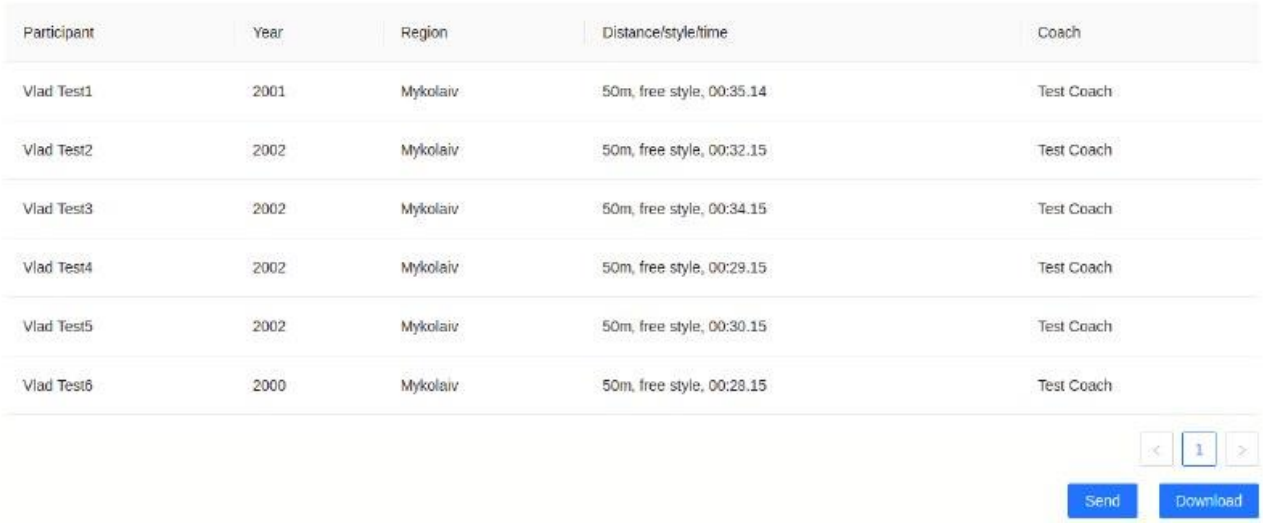

Рисунок 4.18 – Таблиця учасників змагань

Натискаючи на кнопку Download, тренер завантажує документ формату .docx собі на пристрій.

Participant	Year	Region	Distance/style/time	Coach
Vlad Test1	2001	Mykolaiv	50m,free style,00:35.14	Test Coach
Vlad Test2	2002	Mykolaiv	50m,free style,00:32.15	Test Coach
Vlad Test3	2002	Mykolaiv	50m,free style,00:34.15	Test Coach
Vlad Test4	2002	Mykolaiv	50m,free style,00:29.15	Test Coach
Vlad Test5	2002	Mykolaiv	50m,free style,00:30.15	Test Coach
Vlad Test6	2000	Mykolaiv	50m,free style,00:28.15	Test Coach

Рисунок 4.19 – Вигляд таблиці учасників у форматі .docx

Тим часом кнопка Send відправляє заяву до ментора, тобто автора створених змагань, та потрапляє до меню Tournaments в розділ стартовий протокол.

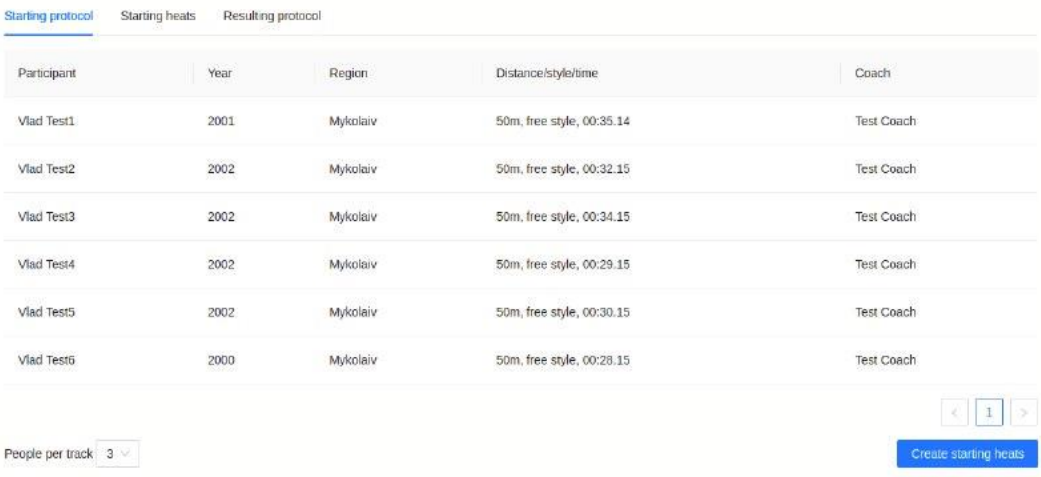

Рисунок 4.20 – Стартовий протокол

На стартовому протоколі організатор може обирати кількість зайнятих доріжек, та натискаючи на кнопку Create starting heats створювати стартові запливи.

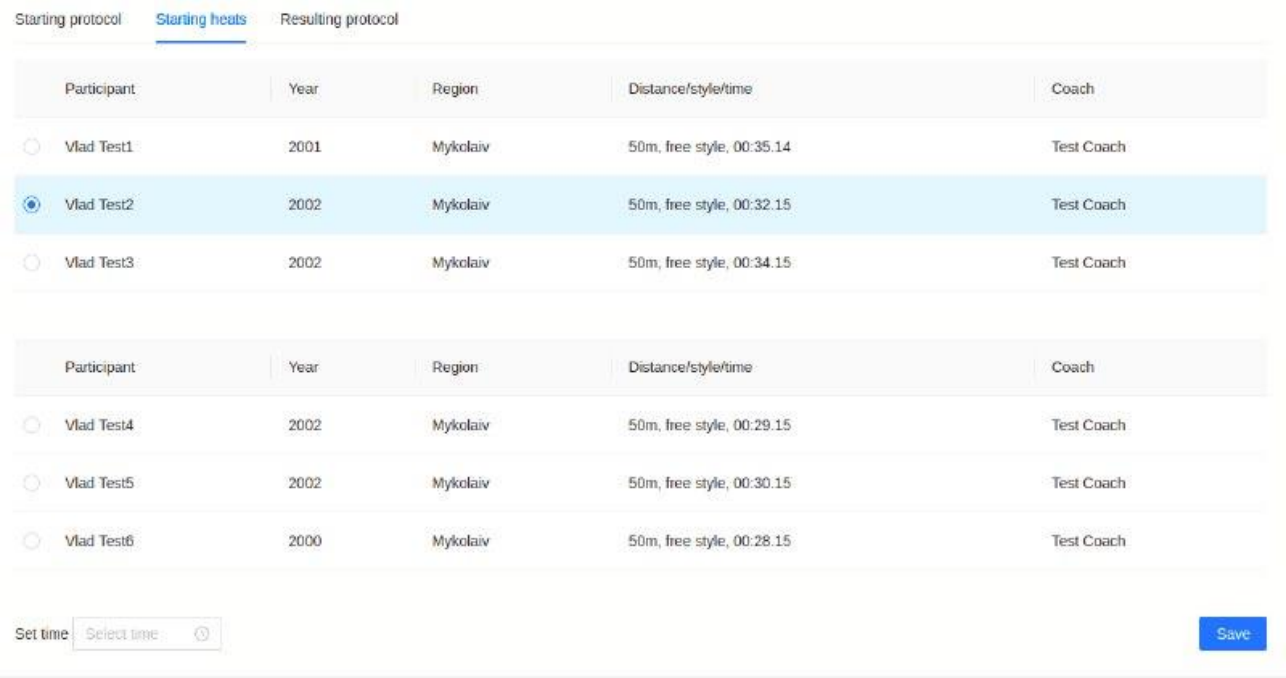

Рисунок 4.21 – Стартові запливи

На сторінці стартових запливів ментор виставляє результати для кожного спортсмена окремо, та натискаючи на кнопку Save створює результуючий протокол з результатами та розподілом місць на змаганнях.

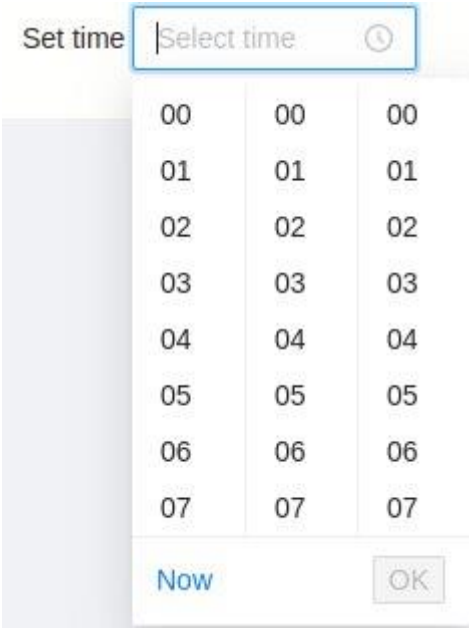

Рисунок 4.22 – Вибір часу запливу

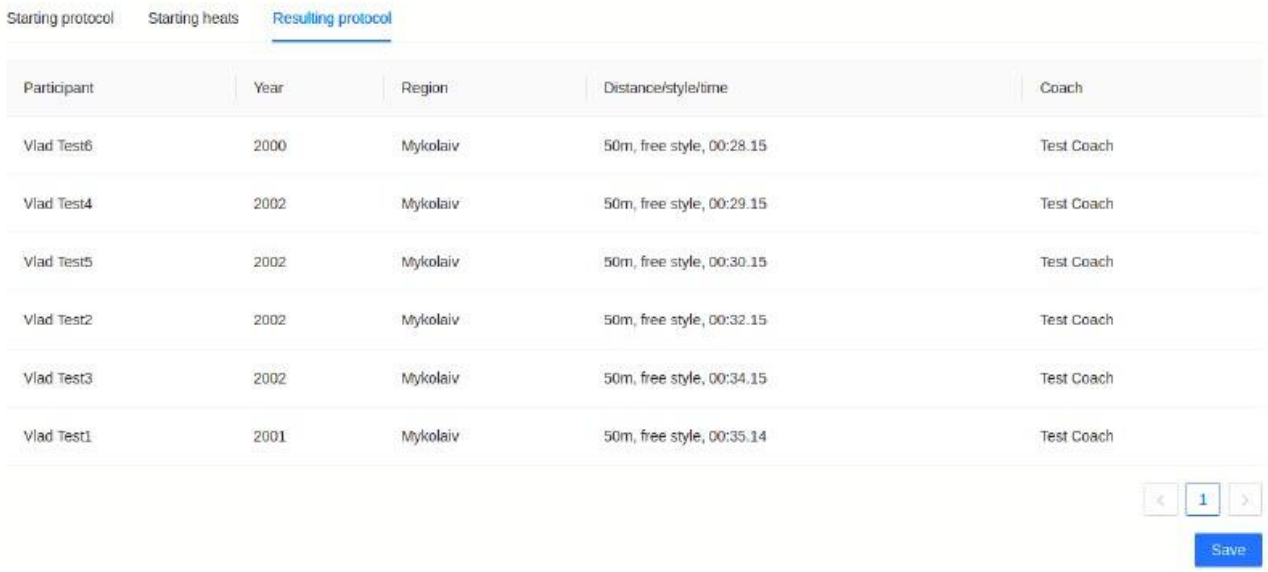

Рисунок 4.23 – Результуючий протокол

Функція збереження протоколу зберігає дані до таблиці рейтинги.

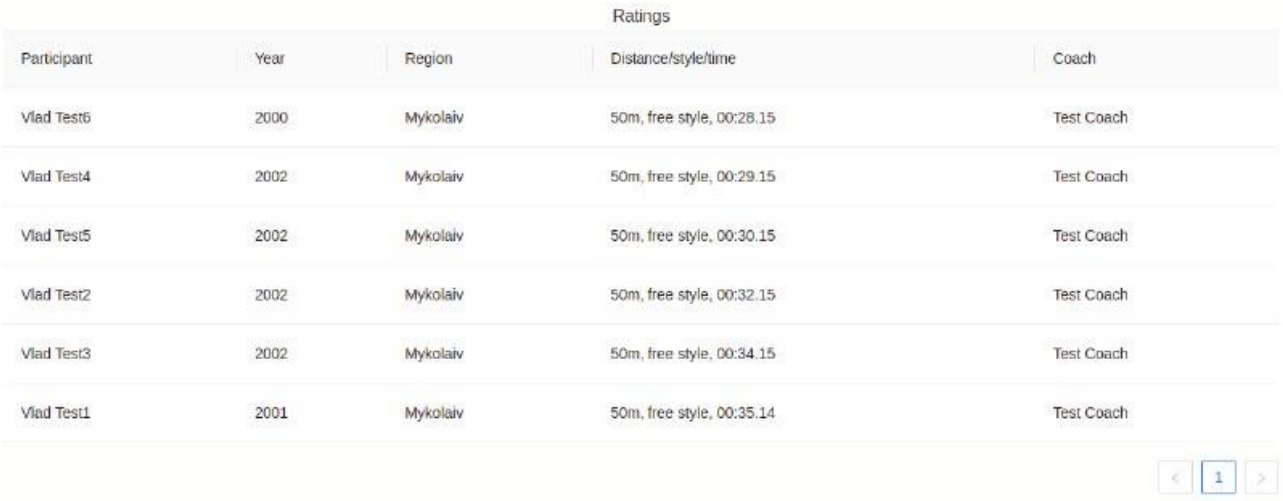

Рисунок 4.24 – Сторінка рейтингів

Таким чином, програма повністю функціонує та виконує поставлені цілі. Користувачі мають змогу провести організацію змагань за допомогою інформаційної системи підтримки змагань з плавання безкоштовно, та з використанням всього доступного функціоналу, що відповідає попереднім вимогам

Висновки до розділу 4

Кожна програма має свій цикл розробки, починаючи від проєктного планування, та закінчуючи тестуванням та полішингом. Одним з основних кроків створення продукту є крок програмної реалізації. Під час написання коду проєкт розкривається зовсім інакше, а перші успіхи синхронізації візуальної частини з логікою надаються додаткової мотивації. Програмна реалізація інформаційної системи підтримки проведення змагань з плавання була досить складної, але напрочуд цікавою. Під час виконання поставленої мети було засвоєно різноманітні технології як на серверній стороні розробки, та і на клієнтській. Використання додаткових бібліотек суттєво допомогли реалізувати об’ємні та складні частини, зекономивши доволі достатньо часу на реалізацію інших функцій.

Остання версія системи дозволяє повністю контролювати процеси організації змагань, а також допомагаю відслідковувати прогрес плавців, та окремих тренерів. Переглянути минулі змагання досить легко та зручно, оскільки база даних зберігає дані допоки вони не будуть видалені адміністратором.

Проаналізувавши та порівнявши вимоги на початку розробки проєкту та працездатності кінцевої версії програми сформувався висновок, що всі цілі виконані, а програмна реалізація цілком може використовуватися в цілях підтримки проведення змагань.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«ІНФОРМАЦІЙНА СИСТЕМА ПІДТРИМКА ПРОВЕДЕННЯ ЗМАГАНЬ З ПЛАВАННЯ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810314

Виконав студент 4-го курсу, групи 402
В. В. Котляренко
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Консультант
канд. техн. наук, доцент
(наук. ступінь, вчене звання)
А. О. Алексєєва
(підпис, ініціали та прізвище)
«___» _____ 2022 р.

Миколаїв – 2022

5 АНАЛІЗ ВИМОГ ДО ОРГАНІЗАЦІЇ ЗАХОДІВ ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ ПІД ЧАС РОБОТИ ЗА КОМП'ЮТЕРОМ

5.1 Основні шкідливі фактори

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп'ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп'ютера, законодавством України чітко врегульовано норми та вимоги до використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп'ютером.

Відповідальність за виконання усіх цих норм покладається на посадових осіб, фізичних осіб, які займаються підприємницькою діяльністю та здійснюють застосування електронно-обчислювальних машин в адміністративних та промислових приміщеннях.

Шкідливий виробничий фактор - фактор середовища або трудового процесу, вплив якого на працівника за певних умов (інтенсивність, тривалість дії тощо) може спричинити професійне або виробничо обумовлене захворювання, тимчасове або стійке зниження працездатності, підвищення частоти соматичних та інфекційних захворювань, призвести до порушення здоров'я як працівника, так і його нащадків [10].

Шкідливі умови праці – стан умов праці, за якого рівень впливу одного або більше факторів виробничого середовища та/або трудового процесу перевищує допустимий [10].

Шкідливими виробничими факторами є:

1) фізичні фактори:

– мікроклімат (температура, вологість, швидкість руху повітря, інфрачервоне випромінювання);

– барометричний тиск;

– неіонізуючі електромагнітні поля та випромінювання: електростатичні поля, постійні магнітні поля, електричні та магнітні поля промислової частоти (50 Гц), електромагнітні випромінювання радіочастотного діапазону, електромагнітні випромінювання оптичного діапазону, зокрема лазерне та ультрафіолетове;

– іонізуючі випромінювання;

– виробничий шум, ультразвук, інфразвук;

– вібрація (локальна, загальна);

– освітлення: природне (відсутність або недостатність), штучне (недостатня освітленість, прямий і відбитий сліпучий відблиск тощо);

– іонізація повітря;

2) хімічні фактори:

– речовини хімічного походження, деякі речовини біологічної природи, які отримані хімічним синтезом та/або для контролю яких використовуються методи хімічного аналізу, аерозолі фіброгенної дії (пил);

3) біологічні фактори:

– мікроорганізми - продуценти, живі клітини та спори мікроорганізмів, що містяться в бактеріальних препаратах, патогенні мікроорганізми;

4) фактори трудового процесу:

– важкість (тяжкість) праці – характеристика трудового процесу, що відображає рівень загальних енергозатрат, переважне навантаження на опорно-руховий апарат, серцево-судинну, дихальну та інші системи.

Важкість праці характеризується рівнем загальних енергозатрат організму або фізичним динамічним навантаженням, масою вантажу, що піднімається і переміщується, загальною кількістю стереотипних робочих рухів, величиною статичного навантаження, робочою позою, переміщенням у просторі.

Категорії робіт за важкістю: легка, середньої важкості, важка, дуже важка.

Напруженість праці – характеристика трудового процесу, що відображає навантаження переважно на центральну нервову систему, органи чуттів, емоційну сферу працівника [10].

До показників, що характеризують напруженість праці, належать: інтелектуальні, сенсорні, емоційні навантаження, ступінь монотонності навантажень, режим роботи [10].

Крім згаданих показників праці, комплексна оцінка її умов включає характеристики небезпечних та шкідливих чинників виробничого середовища, згідно з наведеною раніше класифікацією, та ряд інших чинників трудового процесу, серед яких слід зазначити робочу позу, змінність та напруженість праці, чинники, які характеризують напруженість праці [11].

Комплексну оцінку умов праці за показниками шкідливості і небезпеки чинників виробничого середовища, тяжкості і напруженості трудового процесу здійснюють згідно з Державними санітарними нормами та правилами «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу», затвердженою наказом Міністерства охорони здоров’я України 08.04.2014 р. № 248 (далі – Гігієнічна класифікація праці) [11].

5.2 Нормативна база

З метою створення належних, безпечних і здорових умов праці при роботі з екранними пристроями роботодавець має керуватися наступними нормативно-правовими актами:

– наказом Мінсоцполітики «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» від 14.02.2018 р. № 207, який поширюється на всіх суб’єктів господарювання незалежно від форм власності, організаційно-правової форми і видів діяльності та встановлює мінімальні вимоги безпеки та захисту здоров’я під час здійснення роботи, пов’язаної з використанням екранних пристроїв незалежно від їхнього типу та моделі [12];

– державними санітарними правилами і нормами роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затвердженими

постановою Головного державного санітарного лікаря України 10.12.1998 р. № 7, що застосовуються на підприємствах, організаціях, установах незалежно від форми власності та поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами усіх типів вітчизняного та закордонного виробництва на основі електронно-променевих трубок, якими укомплектовані електронно-обчислювальні машини колективного використання та персональні ЕО [12].

5.3 Загальні вимоги до виробничих приміщень

Приміщення, в яких планується установка та подальша робота з комп'ютером, повинні відповідати проєктній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Конкретні показники зазначених санітарних норм див. в Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98, затверджених Постановою Головного державного санітарного лікаря України №7 від 10 грудня 1998 року. Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ). Так, наприклад, роботодавцю заборонено установлювати комп'ютери в приміщеннях, розташованих у підвалах будинків.

Таблиця 5.1 – Норми клімату для приміщень з екранними пристроями

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с

Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп'ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.

5.4 Вимоги до роботи із комп'ютерними пристроями

Вимоги безпеки до робочих місць працівників з екранними пристроями:

– робочі місця працівників з екранними пристроями мають бути спроєктовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів;

– для забезпечення безпеки та захисту здоров'я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів довкілля - шуму, вібрації, забруднювачів, температури тощо, який не спричиняє соматичних або психічних розладів, а також змін стану здоров'я, працездатності, поведінки, що виходять за межі пристосувальних реакцій) з погляду безпеки та охорони здоров'я працівників;

– організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх елементів робочого місця та їх розташування

ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт;

– освітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98;

– мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 01 грудня 1999 року № 42 (далі - ДСН 3.3.6.042-99);

– робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування;

– робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення;

– сидіння має регулюватися по висоті, спинка сидіння - як по висоті, так і по нахилу;

– слід передбачати підніжку для тих, кому це необхідно для зручності.

Мінімальні вимоги безпеки під час роботи з екранними пристроями:

– щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;

– після закінчення роботи екранні пристрої слід відключати від електричної мережі;

– у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;

– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності;

– Під час виконання робіт операторського типу, пов’язаних з нервово-емоційним напруженням, у приміщеннях під час роботи з екранними пристроями, на пультах і постах керування технологічними процесами та в інших приміщеннях мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 .

5.5 Шуми та вібрації на виробництві

Серед небезпечних факторів під час роботи на виробництві також розрізняють поняття шум та вібрація. Шуми вкрай негативно впливають на здоров’я людини та можуть викликати такі наслідки, як зниження працездатності, спричинення захворювань органів слуху, ендокринної, нервової, серцево-судинної систем, погіршення пам’яті. Вібрація в свою чергу сприяє поширенню сильної втоми, струсу мозку, порушення серцевої діяльності та нервової системи тощо.

Згідно постанови Міністерства охорони здоров’я України про «Санітарні норми виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037.99» [4], за характером спектра шуми слід поділяти на:

– широкосмугові, з безперервним спектром шириною більш ніж одна октава;

– вузькосмужні або тональні , в спектрі яких є виражені дискретні тони. Тональний характер шуму встановлюється вимірюванням випромінювання у третиноктавних смугах частот по перевищенню рівня шуму в одній смузі над сусідніми не менш ніж на 10 дБ.

За часовими характеристика шуми слід поділяти на:

– постійні, рівень шуму яких за повний робочий день при роботі технологічного обладнання змінюється не більш ніж на 5 дБА при вимірюваннях на часовій характеристиці «повільно» шумоміра по шкалі «А»;

– непостійні, рівень шуму яких за повний робочий день при роботі технологічного обладнання змінюється більш ніж на 5 дБА при вимірюваннях за часовою характеристикою «повільно» шумоміра по шкалі «А»4.

Непостійні шуми поділяються на:

– мінливі, рівень яких безперервно змінюється у часі;

– переривчасті, рівень шуму яких змінюється ступінчасто на 5 дБА і більше при вимірюваннях на часовій характеристиці «повільно» шумоміра по шкалі «А», при цьому довжина інтервалів, під час яких рівень залишається сталим, становить 1 с і більше;

– імпульсні, які складаються із одного або декількох звукових сигналів, кожен з яких довжиною менше 1 с, при цьому, рівні шуму у дБ(А1) і дБ(А), виміряні на часових характеристиках «імпульс» та «повільно» шумоміра, відрізняються не менш ніж на 7 дБ.

Захист від шуму:

– архітектурно-планувальні методи: раціональне розміщення споруд, зон і режимів руху транспортних засобів, улаштування захисниг смгу із дерев та кущів;

– організаційно-технічні методи: удосконалення конструкції машин, застосування малошумних машин, дотримання режимів праці і відпочинку;

– застосування акустичних засобів: звукоізоляція машин (кожухи, екрани, перегородки), звукопоглинання, глушники шуму;

– використання засобів індивідуального захисту: протишумові навушники, вкладиші, шоломи, костюми.

Захист від вібрації:

– зниження вібрації у джерелі збудження;

– відсторонення від режиму резонансу;

– вібродемпфірування;

– пружні основи й опори, динамічне гасіння вібрації (установка агрегатів на фундаменти, застосування пружин під сидіннями водіїв);

– віброізоляція.

5.6 Режим праці та відпочинку на підприємствах з комп'ютерними пристроями

Під час планування, аналізу та організації підприємства необхідно розрахувати перерви для відпочинку згідно з офіційним регламентом для збереження здоров'я працівників.

Режими праці і відпочинку мають передбачати додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності. За основну роботу з персональним комп'ютером слід вважати таку, що займає не менше 50 % часу впродовж робочої зміни.

Відповідно до п. 5.3 ДСанПіН 3.3.2.007-98 [6] протягом дня мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством про працю і правилами внутрішнього трудового розпорядку.

Пунктом 5.8 ДСанПіН 3.3.2.007-98 [6] встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні залежно від характеру праці:

– для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп'ютером;

– для операторів персональних комп’ютерів слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп’ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за персональним комп’ютером.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з персональним комп’ютером не повинна перевищувати 4 години. При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин (п. 5.9 та п. 5.10 ДСанПіН 3.3.2.007-98 [6]).

З метою зменшення негативного впливу монотонності є доцільним застосовувати чергування операцій обробки тексту і числових даних (зміна змісту роботи), чергування вводу даних та редагування текстів. Для зниження нервово-емоційного напруження, стомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільні деякі перерви використовувати для виконання комплексу вправ, приклади яких також наведено в ДСанПіН 3.3.2.007-98 [6].

В окремих випадках — при хронічних скаргах працюючих на зорове стомлення, незважаючи на дотримання санітарно-гігієнічних вимог до режимів праці і відпочинку, а також застосування засобів локального захисту очей — допускається індивідуальний підхід до обмеження часу робіт з персональним комп’ютером, зміни характеру праці, чергування з іншими видами діяльності, не пов’язаними з персональним комп’ютером.

Висновки до розділу 5

У результаті виконання спеціальної частини з охорони праці було досліджено вимоги до організації заходів техніки безпеки, охорони праці під час роботи за комп’ютером та правила організації роботи з електронними пристроями.

Під час роботи за персональним комп’ютером людина швидко втомлюється та від цього страждає опорно-руховий, зоровий апарат, нервова система та в окремих випадках психічне здоров’я. Саме тому правила, які прописані у нормативних документах, є актуальними та надзвичайно важливими так, як майже кожне підприємство потребує використання електронних пристроїв.

На жаль, сьогодні існує багато шкідливих умов, які погіршують процес трудової діяльності людини, такі як: шум, вібрація, ультразвук тощо.

Завадити дії шкідливих умов можуть дисципліноване виконання вимог та нормативів, що стосуються охорони праці на підприємстві при роботі з комп’ютерами. Дотримання наведених правил дозволить мінімізувати ризики та можливі негативні наслідки для здоров’я працівника.

ВИСНОВКИ

В наш час існує достатньо інформаційних систем, які можуть використовуватися для найрізноманітніших потреб. Є програми, які мають зручний функціонал, достатньо простий інтерфейс, що вільно або за кошти набувають певного поширення серед потенційних користувачів. Але, незважаючи на кількість функцій, кожна програма має свої можливості та обмеження, свої переваги та недоліки.

Розробка простої інформаційної системи для задоволення особистих потреб пересічного користувача є досить актуальним питанням, оскільки ручна праця поступово автоматизується, або ж навіть стає повністю автоматичною, що суттєво відзначається на кінцевому продукті. Будь-які дії сьогодні будуються в тому числі з обов'язковим прорахунком сил, коштів та часу. Це основні аспекти, які приймаються до уваги, і саме в бік інформаційних систем схиляється суспільство сьогодні.

Метою кваліфікаційного проєкту була розробка інформаційної системи підтримки проведення змагань з плавання. У процесі виконання БКР було вивчено та реалізовано операції організації змагань різних рівнів з використанням сучасних фреймворків та технологій.

Виконані наступні завдання:

- аналіз вимог та існуючих аналогів;
- розробка програмного забезпечення інформаційної системи;
- проведення тестування системи;
- аналіз результатів розробки.

Розроблена система надає можливість створювати змагання, заяви, протоколи, рейтинги, дозволяє редагувати всі наявні в програмі дані. Виходячи з поставлених задач, програма є доступною всюди, де є доступ до інтернет мережі, має досить простий інтерфейс та функціонал, які не потребують додаткового досвіду в даній сфері, а однією з найбільших переваг системи є її безкоштовність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт Міжнародної Федерації з плавання.
URL: https://www.fina.org (дата звернення: 24.05.2022)
2. Ліга плавання України.
URL: https://ukrswim.org (дата звернення: 24.05.2022)
3. Інформаційний ресурс стосовно змагань з плавання в Україні.
URL: https://swimrank.net/ua/News/ (дата звернення: 24.05.2022)
4. Федерація плавання України.
URL: https://www.usf.org.ua/ua/Novyny-Masters.html (дата звернення: 24.05.2022)
5. Моделі розробки програмного забезпечення.
URL: https://ncube.com/blog/top-software-development-models (дата звернення: 24.05.2022)
6. Інформація стосовно бази даних PostgreSQL.
URL: https://www.postgresql.org/about/ (дата звернення: 24.05.2022)
7. Інформація стосовно реляційіних баз даних.
URL: https://aws.amazon.com/relational-database/?nc1=h_ls (дата звернення: 24.05.2022)
8. Документація фреймворку Nest.js.
URL: https://enlear.academy/why-you-should-use-nestjs-as-your-backend-framework-bd1ff1acce5d (дата звернення: 24.05.2022)
9. Інформація стосовно TypeORM.
URL: https://typeorm.io/data-source-api (дата звернення: 24.05.2022)
10. Про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу»: Закон України від 6 трав. 2014 р. за № 472/25249.

URL: https://zakon.rada.gov.ua/laws/show/z0472-14 (дата звернення: 20.05.2022).

11. Голінько В. І. Основи охорони праці. Підручник. Дніпро: НГУ 2014. 272 с.

12. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями [Електронний ресурс] – Режим доступу http://zakon3.rada.gov.ua/laws/show/z0508-18 - Загол. з екрану.

13. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин [Електронний ресурс] – Режим доступу http://zakon3.rada.gov.ua/laws/show/z0382-99 - Загол. з екрану.

14. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98 [Електронний ресурс] – Режим доступу: http://mozdocs.kiev.ua/view.php?id=2445 – Загол. з екрану.

15. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин [Електронний ресурс] – Режим доступу http://zakon3.rada.gov.ua/laws/show/z0382-98 - Загол. з екрану.

16. Санітарно – гігієнічним нормам №2152-80 [Електронний ресурс] – Режим доступу: https://dnaop.com/html/2296/doc -ГН_2152-80 - Загол. з екрану.

17. СН 3044-84. Санітарні норми вібрації робочих місць [Електронний ресурс] – Режим доступу: https://dnaop.com/html/43248/doc -ДНАОП_3044-84 – Загол. з екрану.

18. ГОСТ 12.1.012-90. Вібраційна безпека. Загальні вимоги [Електронний ресурс] – Режим доступу: https://dnaop.com/html/1602/doc -ГОСТ_12.1.012-90 – Загол. з екрану.