

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Кафедра комп'ютеризованих систем управління**

ДОПУСТИТИ ДО ЗАХИСТУ  
Завідувач кафедри

“ \_\_\_\_\_ ” \_\_\_\_\_ 202\_ р.

**КВАЛІФІКАЦІЙНА РОБОТА  
(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ  
“МАГІСТР”**

**Тема:** \_\_\_\_\_ Програма автоматизації керування спортивно-оздоровчим  
комплексом \_\_\_\_\_

**Виконавець:** \_\_\_\_\_ Терехов Владислав Олегович \_\_\_\_\_

**Керівник:** \_\_\_\_\_ Масловський Борис Георгійович \_\_\_\_\_

**Консультанти з окремих розділів пояснювальної записки:**

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

**Нормоконтролер:** \_\_\_\_\_ Масловський Борис Георгійович \_\_\_\_\_

**Київ 2022**

# НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»  
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»

Форма навчання денна

ЗАТВЕРДЖУЮ  
Завідувач кафедри

«\_\_\_\_\_» \_\_\_\_\_ 202\_\_ р.

## ЗАВДАННЯ

на виконання кваліфікаційної роботи (проєкту)

Терехова Владислава Олеговича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи (проєкту) Програма автоматизації керування спортивно-оздоровчим комплексом

затверджена наказом ректора від «14» листопада 2022р. № 368/08

2. Термін виконання роботи (проєкту): з 22.08.2022 по 20.11.2022

3. Вихідні дані до роботи (проєкту): Методи та правила побудови, складання та оформлення програми автоматизації керування спортивно-оздоровчим комплексом

4. Зміст пояснювальної записки: Розділ 1 Аналіз існуючих програм керування підприємством, Розділ 2 Програма автоматизації спортивно-оздоровчого комплексу, Розділ 3 Програмна реалізація розроблених алгоритмів тренувань

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

Загальна характеристика програми, структурна схема алгоритму роботи програми, концептуальна модель таблиці з інформацією про користувачів, вигляд таблиці з інформацією про програму тренувань, діаграма алгоритмів сайту, схема алгоритму надання доступу до створення програми тренувань



## РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Програма автоматизації керування спортивно-оздоровчим комплексом»: с., 12 рис., 4 табл., 14 літературних джерела. Об'єкт дослідження: програма організації роботи з відвідувачами спортивно-оздоровчого комплексу. Мета роботи: проєктування та розробка програми автоматизації керування спортивно-оздоровчим комплексом. Методи дослідження: проведення порівняльного аналізу, обробка літературних джерел. Для організації роботи з відвідувачами спортивно-оздоровчого комплексу, було прийнято рішення створити WEB - програму, яка буде посередником у взаємодії між відвідувачем та тренером, таким чином це буде сприяти підвищенню комфорту відвідувачів, а отже і популярності та іміджу спортивного залу. Розробка сайту програм тренувань дасть можливість скоротити час відвідувачів та тренерів зникне потреба в заповнюванні різних видів тренувань від руки, достатньо створити програму онлайн. Для розробки додатку було використано мову HTML, управління базами даних застосовують за допомогою системи PostgreSQL. Клієнтська та програмно-апаратна частина веб-додатку написано за допомогою бібліотеки React та Node.js відповідно. Візуальне стилізування сайту створене засобами таблиць стилів CSS, основна логіка елементів сайту реалізується за допомогою надбудови мови JavaScript — React.js. Слід зазначити, що інтегрування автоматизації спортивно-оздоровчого комплексу повинно здійснюватися, починаючи з найважливішої частини його роботи, автоматизація якої дасть можливість швидко отримати гарний результат. Під час розробки веб - додатку потрібно забезпечити можливість роботи за новою, та за старою технологіями, щоб не втручатися в функціонування спортивно-оздоровчого комплексу.

АВТОМАТИЗАЦІЯ ІНФОРМАЦІЙНИЙ СИСТЕМА, ДОДАТОК СПОРТИВНОГО ЗАЛУ, ФРЕЙМВОРК, HTML, CSS, POSTGRESQL, JAVASCRIPT, REACT.JS

## Зміст

ВСТУП.....	6
Розділ 1. Аналіз існуючих програм керування підприємством.....	14
1.1. Загальна характеристика керування підприємством за допомогою програми 14	
1.2. Аналіз існуючих програм .....	16
1.3. Висновки до розділу .....	21
Розділ 2. Програма автоматизації спортивно-оздоровчого комплексу .....	23
2.1. Основні положення.....	23
2.2. Серверна частина програми. ....	25
2.3. Клієнтська частина системи.....	28
2.4. Висновки до розділу .....	33
Розділ 3. Програмна реалізація розроблених алгоритмів тренувань.....	35
3.1. Вибір технологій для програмної реалізації .....	35
3.2. Обґрунтування вибору мови .....	40
3.3. Тестування програми.....	46
3.4. Висновки до розділу .....	55
ВИСНОВКИ .....	57
СПИСОК ЛІТЕРАТУРИ.....	59
ДОДАТОК.....	60

## ВСТУП

В процесі популяризації здорового образу життя збільшилась кількість спортивних закладів. Тому актуальним є питання автоматизації роботи цих закладів. Комплексна автоматизація бізнес-процесів допоможе впоратися з цими завданнями і забезпечити сталий розвиток.

Завдання автоматизації того чи іншого закладу сфери послуг визначаються його бізнес-процесами.

Перш за все, це важливість персонального підходу до клієнта тому що, в спортивний зал клієнт йде за певною процедурою, і навіть до певного тренера. Відвідувач спортивного клубу оплачує послуги на суму декілька тисяч гривень і розраховує отримати за ці гроші індивідуальне обслуговування, розраховане саме на його потреби. Йому важливо час надання послуги, місце, фахівець, який надає послугу, а не просто «доступ» до тренажера.

По-друге, тут важлива можливість забронювати послугу. По-третє, спортивні зали постійно розширюють спектр пропонованих послуг: групові, силові, танцювальні заняття, водні програми, послуги тренажерного залу, персональні тренування та інше. Це вимагає від програми автоматизації більшої гнучкості і можливості настройки в процесі експлуатації. Нарешті, спортивні зали працюють з клієнтами протягом, як правило, тривалого часу. Причому, чим більше період дії клубної картки, і вище відсоток продовження, тим послуга вигідніше для центру. Тому сьогодні, намагаючись закріпити клієнтів, спортклуби пропонують великий вибір послуг - сімейні, ранкові, дитячі карти, клубні річні карти, корпоративні карти, абонементи. І це теж система повинна враховувати, як на програмному, так і на апаратному рівні.

Якщо ж говорити про окупність систем автоматизації в будь-яких видах послуг, в тому числі спортивних, то ефект дуже великий. Наприклад, такий факт: після впровадження автоматизації в гірськолижному комплексі надходження грошових коштів збільшується на 30-35%, що пояснюється, скажімо так, усуненням

людського фактора. Звичайно, при автоматизації роботи спортивного клубу, в зв'язку з тим, що затребуваність послуги залежить від різних умов (багато що залежить від того ж тренера, наприклад), важче виокремити саме «системну» складову прибутку. Проте, і виключення зловживань з боку персоналу, і підвищення лояльності клієнта безпосередньо і відчутно впливають на прибутковість.

Створення “програми автоматизації керування спортивно-оздоровчим комплексом” дозволить частково автоматизувати процес роботи, що збільшить прибутковість організації та лояльність клієнтів.

Спортивно-оздоровчий комплекс здійснює фізкультурно-оздоровчу діяльність, а саме: проведення занять у групах з різних напрямків (фітнес, степ аеробіка, дитяча спортивна аеробіка, танці тощо), а також самостійні заняття у спортзалі під наглядом тренера, індивідуальні заняття з тренером.

Як нове підприємство комплекс росте та розвивається. Ведеться постійний пошук перспективних напрямів діяльності, нових, ефективніших способів організації управління.

Найбільш формалізованою сферою діяльності є робота з відвідувачами.

Робота з постачальниками має менш постійний характер. Наприклад, разові закупівлі обладнання, разові послуги будівельних організацій. Постійнішими є закупівлі спеціального спортивного харчування, постачання чистої води.

Взаємодія з іншими організаціями – це разові заходи, коли клуб запрошується для участі у змаганнях (наприклад федерацією аеробіки), взаємодіє з іншими клубами в інших містах, обмінюється аудіо-відео-інформацією, майстер-класи.

Розвиток у нашій країні ринкових відносин, включення до світових інтеграційних процесів змушує вже сьогодні наближатися до вимог світових стандартів. Зростають вимоги до об'єктивної оцінки фінансового стану, становища підприємств, координації стратегій, що спрямоване на зниження фінансових ризиків та отримання конкурентних переваг.

Це зумовлює необхідність застосування автоматизованих інформаційних технологій.

Впровадження інформаційних технологій у процес планування і управління діяльністю підприємств передбачає як обробку великих і взаємозалежних масивів даних, але можна використовувати також їхнього аналізу та обґрунтувань варіантів управлінських рішень. При цьому важливу роль відіграють облік різноманітних відомостей про сектори економіки, регіон, фірму та інші суб'єкти господарювання, а також облік фінансових, трудових і матеріальних ресурсів.

Обсяги інформації, високі вимоги до точності та достовірності, необхідність ефективного аналізу фінансового стану клієнтури та фірми – ось основні причини, що визначають автоматизацію бізнесу.

Автоматизація завдань залежить в першу чергу від зміни форм взаємодії машини та користувача. На початкових етапах застосування ЕОМ переважно вирішувалися завдання, що формують зведені дані про результати діяльності організації за звітний період. ЕОМ використовувалися в режимі пакетної обробки, що викликало затримку у прийнятті рішень та виключало роботу співробітників з клієнтами в реальному масштабі часу.

Положення змінилося в даний час, коли значна

частина працюючих у компаніях комп'ютерів є одним із типів розподілених обчислювальних систем. Розподілені системи будуються на базі АРМ спеціаліста, з'єднаних каналами зв'язку в обчислювальній мережі багатопроцесорних комп'ютерів та багатомашинних обчислювальних комплексів.

Основне призначення робочих місць управлінських та інших працівників – забезпечити умови для комфортної, високопродуктивної та якісної роботи. Для цієї мети робоче місце фахівця має бути зручно сплановане, оснащене всім необхідним, що гарантує його безпеку роботи та забезпечує його безперебійну інформаційну підтримку та раціональну організацію праці на цьому місці.



У найзагальнішому вигляді робочі місця управлінських працівників можна поділити на індивідуальні та групові, що нараховують одну або кілька одиниць обладнання; поодинокі, серійного типу чи масового типу; ручні, механізовані, автоматизовані.

Індивідуальні робочі місця характерні керівників різних рангів, групові – особам, які готують інформацію з її подальшого використання та прийняття управлінських рішень керівниками.

На ручному немеханізованому робочому місці працівник сфери організаційного управління користується столом та спеціальними меблями, телефоном, лінійкою, таблицями та іншими підсобними засобами. Механізоване робоче місце характеризується включенням у процес простих і програмованих калькуляторів, що виконується на ньому, спеціалізованих засобів реалізації інформаційних технологій, що дозволяють скоротити тривалість виконання деяких часто повторюваних розрахунків і операцій.

Створення автоматизованих робочих місць передбачає, що основні операції з накопичення, зберігання та переробки інформації покладаються на обчислювальну техніку, а спеціаліст виконує певну частину ручних операцій та операцій, що вимагають творчого підходу при підготовці управлінських рішень. Обчислювальна техніка при цьому працює в тісній взаємодії з користувачем, який контролює її дії, змінює значення окремих параметрів у ході розв'язання задачі, а також вводить вихідні дані для вирішення задач та функцій управління.

Термін «автоматизоване робоче місце» нині знайшов стала вельми поширеною, як і літературі, і практично.

Серед фахівців з інформаційних технологій і систем автоматизованої обробки економічної інформації найбільш усталеним є визначення АРМ як деякої сукупності апаратних, програмних, методичних і мовних засобів, що забезпечують автоматизацію функцій користувача в деякій предметній області і дозволяють оперативно задовольняти його інформаційні та обчислювальні потреби забезпечення переважно на магнітних дисках.

При побудові АРМ можливо кілька архітектурно-технологічних рішень:

з урахуванням великих універсальних ЕОМ;

з урахуванням малих ЕОМ;

з урахуванням персональних комп'ютерів.

Однак користувач при будь-якому рішенні повинен мати на своєму робочому місці пристрої, які допомагають виконувати його посадові функції.

АРМ, побудовані на базі великих універсальних ЕОМ, забезпечують спеціалістам організаційно-економічного управління можливість працювати з великими базами даних за технічної та програмної підтримки, що здійснюється силами професіоналів у галузі обчислювальної техніки. Однак ряд факторів ускладнює широке використання великих універсальних ЕОМ як базу для створення таких АРМ. Насамперед, ця вимога мати в організації спеціальний підрозділ з технічного та програмного забезпечення обчислювальних засобів, недостатня гнучкість програмних засобів, жорсткі вимоги технічних засобів до операційної системи; висока вартість машинних ресурсів, слабка орієнтація обчислювальної системи на користувача-непрограміста та ін.

АРМ, побудовані з урахуванням малих ЕОМ, дещо знижують вартісні витрати, але зберігають більшість зазначених недоліків.

АРМ, створені з урахуванням персональних комп'ютерів, найпростіший і найпоширеніший варіант автоматизованого робочого місця для працівників сфери організаційного управління. І тут АРМ сприймається як система, що у інтерактивному режимі взаємодії представляє конкретному працівнику (користувачу) всі види забезпечення монополюю весь час сеансу роботи. Користувач сам виконує всі функціональні обов'язки з перетворення інформації.

Автоматизація робочого місця спеціаліста організаційного управління приносить значно більшу користь, коли користувач даного АРМ працює в контакті з іншими фахівцями, що використовують АРМ, а також підключається до локальних та регіональних мереж обробки даних. У зв'язку з цим при функціонуванні ПЕОМ в автономному режимі слід передбачати відповідні процедури обміну інформацією між фахівцями, наприклад, шляхом простого обміну дискетами, використовуючи канали обміну інформацією та засоби передачі даних.

Найбільш ефективним режимом роботи АРМ є його функціонування в рамках локальної обчислювальної мережі як робоча станція. Особливо доцільним є такий варіант, коли потрібно розподіляти інформаційно-обчислювальні ресурси між кількома користувачами.

Наступною більш складною формою є АРМ з використанням ПЕОМ як інтелектуальний термінал, а також з віддаленим доступом до ресурсів центральної ЕОМ або зовнішньої мережі. В даному випадку кілька ПЕОМ підключаються каналами зв'язку до головної ЕОМ, при цьому кожна ПЕОМ може працювати і як самостійний термінальний пристрій.

Проектування та впровадження АРМ ґрунтуються на низці загальних та приватних принципів проектування систем обробки даних:

принцип максимальної орієнтації на кінцевого користувача – досягається створенням спеціальних засобів адаптації АРМ до рівня підготовки користувача та можливістю його навчання, у тому числі самонавчання безпосередньо на цьому АРМ;

принцип проблемної орієнтації – під проблемною орієнтацією зазвичай розуміється спеціалізація АРМ на вирішенні певного класу завдань, об'єднаних загальною технологією обробки даних, єдністю режимів роботи та експлуатації, або у вузькому сенсі – орієнтація на автоматизацію певної групи функцій, які постійно виконуються працівниками;

принципу відповідності інформаційних потреб користувачів використовуваним технічним засобам.

Функціонування будь-якого АРМ потребує різних видів забезпечення:

інформаційне забезпечення АРМ передбачає організацію його інформаційної бази, регламентує інформаційні зв'язки та визначає склад та зміст усієї системи інформаційного відображення;

математичне забезпечення АРМ є сукупністю алгоритмів, що забезпечують введення, контроль, зберігання та коригування інформації, формування результатної інформації та оформлення її у вигляді таблиць, графіків, діаграм, а також забезпечення достовірності та захисту інформації;

програмне забезпечення (ПЗ) будь-якого АРМ в основному визначає його інтелектуальні можливості, професійну спрямованість, широту та повноту здійснення функцій, можливості застосування різних технічних пристроїв;

лінгвістичне забезпечення АРМ включає мови спілкування з користувачем, мови запитів, інформаційно-пошукові мови, мови-посередники у мережах;

технологічне забезпечення АРМ є деякою чітко встановленою сукупністю проектних рішень, що визначають послідовність операцій, процедур, етапів у відповідній сфері діяльності користувача;

організаційне забезпечення включає комплекс документів, які регламентують діяльність фахівців під час використання ПЕОМ чи терміналу з їхньої робочому місці;

методичне забезпечення АРМ складається з методичних вказівок, рекомендацій та положень щодо впровадження, експлуатації та оцінки ефективності їх функціонування;

ергономічне забезпечення АРМ є комплексом заходів, виконання яких має створювати максимально комфортні умови для використання АРМ фахівцями, якнайшвидшого освоєння технології та якісної роботи на АРМ;

правове забезпечення включає систему нормативно-правових документів, які мають чітко визначати правничий та обов'язки фахівців за умов функціонування АРМ.

## **Розділ 1. Аналіз існуючих програм керування підприємством**

### **1.1. Загальна характеристика керування підприємством за допомогою програми**

Люди отримують інформацію за допомогою своїх органів почуттів: звуки через слух, зображення і текст через зір, форми, температуру і прихильність через дотик, і запахи через нюх. Щоб інтерпретувати сигнали, отримані від органів почуттів, люди розробили і вивчили складні системи мов, що складаються з "алфавітів" символів і стимулів, і пов'язаних з ними правил використання. Це дозволило їм розпізнавати об'єкти, які вони бачать, розуміти повідомлення, які вони читають або чують, і розуміти знаки, отримані через тактильні і нюхові почуття.

Носіями інформаційно-передавальних знаків, що сприймаються органами почуттів, є енергетичні явища — звукові хвилі, світлові хвилі, хімічні та електрохімічні стимули. На інженерній мові люди є рецепторами аналогових сигналів та за дещо розпливчастою угодою, повідомлення, що передаються через ці носії, називаються інформацією аналогової форми або просто аналоговою інформацією. До появи цифрового комп'ютера Когнітивна інформація зберігалася і оброблялася тільки в аналоговій формі, в основному за допомогою технологій друку, фотографії та телефонії.

Хоча люди вправні в обробці інформації, що зберігається в їх пам'яті, аналогова інформація, що зберігається поза розумом, обробляється нелегко. Сучасні інформаційні технології значно полегшують маніпулювання зовні зберігається інформацією в результаті її представлення у вигляді цифрових сигналів, тобто у вигляді наявності або відсутності енергії (електрики, світла або магнетизму). Інформація, представлена в цифровому вигляді в двох станах або двійковій формі, часто називається цифровою інформацією.

Сучасні програми характеризуються великими об'ємами аналогової і цифрової інформації. Що стосується зберігання та передачі інформації, то перехід від аналогової до цифрової інформації настільки поширений, що призвів до історичної трансформації способу, яким люди створюють, отримують доступ та використовують інформацію.

Автоматизація означає колекцію безлічі одиниць обладнання, задіяних у процесі, зберігання та обробці, інформації під впливом людини.

Апаратне, програмне забезпечення, зв'язки та інформація комп'ютерної системи, користувачі програми та корпус системи — все це частина. Персональні комп'ютери, смартфони, бази даних та мережі — це лише деякі приклади інформаційних систем.

Підприємства та корпорації використовують інформаційні системи для взаємодії зі своїми постачальниками та клієнтською базою, виконують їх операції, управляють їх організацією та проводять маркетингові кампанії.

Вони можуть використовуватися для найрізноманітніших цілей, від управління ланцюгами поставок до взаємодії з цифровими ринками. Люди також покладаються на програми для взаємодії з однолітками та друзями через соціальні мережі, здійснюючи повсякденну діяльність, таку як банківська справа та покупки або просто шукаючи знання та інформацію.

Актуальність роботи полягає в доцільності використання даної платформи у наш час, у зв'язку з ситуацією у країні.

Мета: Підвищення ефективності онлайн взаємодії тренерів з клієнтами (скорочення часу на передачу інформації в взаємодії). Об'єкт дослідження: процес взаємодії тренерів та спортсменів Предмет дослідження: застосування сучасних технологій створення вебсервісу для розробки веб-застосунку для взаємодії тренерів та спортсменів. Методи дослідження – Під час розв'язку поставлених задач було використано методи порівняння, за допомогою яких було проаналізовано існуючі рішення та аналогів. Наукова новизна – наукова новизна роботи полягає у розробці вебзастосуноку спрямованого на підвищення ефективності взаємодії тренерів з клієнтами через перехід їх

взаємодію в онлайн. Практична цінність – розроблено вебзастосунок з використання сучасних технологічних рішень, що вирішує актуальні проблеми та актуальні недоліки конкурентів.



Рисунок 1.1 — Функціональна схема програми

## 1.2. Аналіз існуючих програм

Коли ви можете підвищити ефективність свого робочого процесу при одночасному зниженні витрат, ви досягаєте оптимізації робочого процесу.

Мета оптимізації полягає в тому, щоб забезпечити протікання робочих процесів з максимальною ефективністю. Ви повинні застосовувати автоматизацію, але тільки тоді, коли ваші процеси виконуються належним чином. Перш ніж зробити це, їх потрібно оптимізувати.

Робочий процес включає в себе повторювані завдання, які повинні тривати для досягнення мети організації розроблені унікально для досягнення пов'язаних з ними цілей і завдань.



Один з найяскравіших практичних прикладів використання інформаційних технологій на підприємстві - автоматизація управління. Завдяки програмному забезпеченню і технічних засобів управлінська діяльність стала трохи ефективніше. Справа в тому, що сучасним керівникам доводиться працювати з великими обсягами інформації, швидко приймати рішення і доводити їх до відома виконавців, постійно здійснювати контроль за фінансовими, економічними та технічними процесами.

Просто установка персональних комп'ютерів не може вирішити дане питання. Це тільки перший крок до комплексної автоматизації підприємства. Головна мета автоматизації — створення потужного і універсального інструменту, який дозволить швидко аналізувати діяльність компанії і приймати ефективні рішення. Важливо, що саме керівник ставить завдання і визначає стратегію.

Автоматизація підприємства — перспективний напрямок, що дозволяє:

- 1) замінити ручну працю;
- 2) знизити витрати підприємства;
- 3) підвищити ефективність діяльності організації;
- 4) спростити велику кількість процесів.

Правильна комбінація комп'ютерного обладнання та спеціального програмного забезпечення покращує і спрощує безліч аспектів роботи підприємства та збільшує прибутковість.

Суть автоматизації полягає не тільки в установці сучасного обладнання, а й у вирішенні безлічі комплексних завдань з використанням інноваційних підходів. Удосконаливши і структурувавши підприємство, складно уявити його роботу без певних фахівців. Для проведення автоматизації нерідко потрібні відповідні спеціально розроблені під конкретне підприємство програми.

Супермаркети завдяки автоматизації можуть скоротити час і підвищити рівень обслуговування покупців. Для керівників полегшується завдання контролю роботи підлеглих, з'являється можливість своєчасно виявляти нові

можливості або ризику, вживати необхідних заходів. Статистичні дані полегшують прогнозування діяльності. Завдяки автоматизації можна забезпечити ефективну взаємодію різних підрозділів одного підприємства.

Автоматизація — це актуальна тенденція у всіх сферах бізнесу, на виробничих підприємствах. Завдяки автоматизації з'явилася можливість ефективно проводити більшість аспектів діяльності будь-якого підприємства.

Спортивно-оздоровчі комплекси та інші організації, які здійснюють фізкультурно-оздоровчу діяльність, останнім часом стають все більш популярними. У зв'язку з цим зріс попит на інформаційні системи, що дозволяють вирішувати не тільки типові, але і специфічні завдання.

На ринку представлений ряд програмних продуктів, покликаних виконувати облік завдань, характерних для даного виду діяльності. Для огляду вибрано декілька інформаційних систем, які проаналізовані по ряду критеріїв: функціональність системи, вартість використання, а також виявлені переваги і недоліки кожної з систем.

Для аналізу було обрано такі інформаційні системи: «UNIVERSE Фітнес», «1С: ФІТНЕС-клуб», «Абонемент», «КРАФТ ERP Фітнес-клуб», «Фітнес-клуб». Всі п'ять програмних продуктів позиціонуються як програмне забезпечення для автоматизації діяльності спортивних клубів, фітнес центрів, тренажерних залів тощо.

«UNIVERSE Фітнес» — програма розроблена компанією «UNIVERSE — Soft» для управління процесами, що відбуваються в фітнесцентрі. Забезпечує ведення клієнтської бази, зберігає історію спілкування з клієнтом, виводить нагадування, реєструє виконання завдань. Включає в себе систему обліку контролю робочого часу, планування групових та персональних занять, автоматичний розрахунок заробітної плати, облік товарів і введення складського обліку.

Система відрізняється досить широким функціоналом. Однак основний акцент зроблено на роботу з клієнтами, в той час як в складському та касовому обліку реалізований лише базовий функціонал. Також система має дуже

високу вартість, в повній конфігурації вона коштуватиме покупцю більше 20000 грн.

«ІС: ФІТНЕС-КЛУБ». Галузеве рішення, розроблене «Helix-Group» на платформі «ІС: Підприємство», призначене для автоматизації спортивних клубів та інших організацій подібного спрямування, сферою діяльності яких є надання клієнтам як можливості заняття спортом, так і послуг різного роду: солярій, масаж тощо.

Дане рішення має широкий функціонал, розробники виділяють кілька основних функцій:

1) Робота з клієнтами. Дозволяє формувати клієнтську базу, і при цьому зберігати різноманітну інформацію про клієнта: починаючи з персональних даних і рахунків клієнта та закінчуючи цілями відвідування, уподобаннями в напоях і приналежності до певного сегменту;

2) Облік фінансів. Даний розділ враховує всі маніпуляції грошовими коштами, які проводяться при роботі з клієнтами;

3) Управління персоналом. Основні можливості: зберігання даних про співробітників, формування робочого графіка, облік фактично відпрацьованого ними часу, розрахунок заробітної плати, проведення розрахунків з співробітниками;

4) Облік запасів на складі. Включає автоматизацію взаємодії з постачальниками, контроль закупок і інвентаризацію.

Слід зазначити, що даний програмний продукт не передбачає ведення бухгалтерського та податкового обліку. Для реалізації цих можливостей необхідно провести інтеграцію з продуктом «ІС: БУХГАЛТЕРІЯ 8», що вимагає додаткових ресурсів. Також до недоліків можна віднести короткий термін безкоштовного гарантійного обслуговування в порівнянні з іншими програмними продуктами.

«Абонемент» — інформаційна система розроблена компанією «UCS» для автоматизації роботи спортивного клубу. Вартість даного програмного забезпечення становить 7000 гривень. При цьому у вартість не включена підтримка клієнта, для цього необхідно додаткове укладення договору на абонентське обслуговування.

Інформаційна система «Абонемент» дозволяє автоматизувати в основному облік клієнтів і маніпуляцій з абонементом. Це локальна програма, що не підходить для мережі спортклубів. До того ж, ще одним недоліком є відсутність складського обліку, а також можливості ведення фінансової, бухгалтерської звітності без інтеграції з іншим програмним забезпеченням.

«КРАФТ Фітнес-клуб». Програмний комплекс для фітнес-клубів, розроблений «Біном - Софт», що дозволяє автоматизувати діяльність не тільки фітнес-клубів, але і SPA-салонів та інших підприємств, надаючих послуги для клієнтів на клубній основі та має наступний функціонал:

- реєстрація клієнтів у системі;
- підтримка клієнтської бази;
- автоматизація заповнення договорів та оплата додаткових послуг;
- оплата послуг;
- складання звітів.

«КРАФТ Фітнес-клуб» має досить широкі функціональні можливості по роботі з клієнтами. Однак в системі не реалізована можливість ведення бухгалтерського і податкового обліку. Але інтеграція з іншими додатками, розробленими організацією «Біном Софт» дозволить після покупки додаткового модуля здійснювати комплексний облік в організації. «Фітнес клуб» — система розроблена організацією ТОВ «Простий софт» дозволяє створити повноцінний облік відвідуваності клубу, сформувавши при цьому базу даних клієнтів.

«Фітнес клуб» дозволяє здійснювати облік клієнтів, абонементів та маніпуляцій з ними, але набір функцій значно менше набору функцій

попередніх систем. Даний програмний продукт слід використовувати в організаціях з невеликою клієнтською базою.

В результаті огляду ряду програмних продуктів, що існують в даний момент на ринку, були переваги та недоліки кожного з них.. Незважаючи на те, що деякі з них мають широкий функціонал, можливості складського і бухгалтерського обліку, а також доступну цінову політику, характеристики даних програмних продуктів не збігаються з заявленими вимогами.

По-перше, спортивний зал, для якого розробляється продукт, не потребує функціонал, яким володіють більшість існуючих рішень. Так що розроблена інформаційна система не повинна мати можливості ведення бухгалтерського та податкового обліку, так як фінансовою звітністю займається головний бухгалтер. Також програма передбачає відсутність складського обліку, оскільки складу в спортклубі не має. Спортклуб не потребує реалізацію додаткових послуг, таких як можливість відвідування солярію або масажу, через їх відсутність.

По-друге, необхідно враховувати вимоги замовника, основними з яких є простота реалізації і мінімальна вартість продукту. Таким чином, специфіка розроблюваної інформаційної системи не дозволяє скористатися будь-яким існуючим рішенням і вимагає створення нового продукту, що враховує всі особливості його діяльності.

### **1.3. Висновки до розділу**

Цілю кваліфікаційної роботи є створення макету та побудова додатку організації роботи спортивно-оздоровчого комплексу.

Існуюче на даний момент обстеження стан справ на об'єкті автоматизації, можна назвати такі недоліки ведення документації у цій предметної області:

- 1) значні часові витрати на оформлення заяв;
- 2) можливе дублювання даних клієнта під час оформлення;
- 3) ведення обліку клієнтів вручну на паперовому носії;

- 4) зберігання великого обсягу документів на паперовому носії;
- 5) пошук документації є досить трудомістким і потребує великих тимчасових витрат;
- 6) адміністратору, що веде облік клієнтів необхідно швидко та головне безпомилково доповнювати та виправляти дані;
- 7) невелика кількість клієнтів, що обслуговуються.

Грунтуючись з цього було прийняте рішення створити програмний додаток для автоматизації роботи спортивного залу. Поставлена мета створити WEB-додаток для організації роботи з відвідувачами спортивного-оздоровчого комплексу, що повинен виконувати наступні завдання:

- 1) механізм ідентифікації користувача у системі;
- 2) створення та зберігання програм тренувань у цифровому форматі;
- 3) додавання, видалення, перегляд програм тренувань;
- 4) надання доступу до програми тренувань;
- 5) перевірка правильності введення інформації.

Проведений аналіз існуючих рішень організації роботи з клієнтами спортивних споруд дає можливість зазначити, що розглянуті рішення надають можливість побдови програми.

Для цього необхідно:

- 1) Розробити алгоритми програми
- 2) Провести розробку серверної частини програми
- 3) Провести розробку клієнтської частини програми
- 4) Обрати мову програмної реалізації
- 5) Провести тестування розробленої програми

## **Розділ 2. Програма автоматизації спортивно-оздоровчого комплексу**

### **2.1. Основні положення**

Для запровадження системи організації роботи з відвідувачами спортивного залу треба розробити WEB-сайт, який може служити потужним інструментом для створення іміджу організації. Вся інформація, яка розміщується на сайті, формує думку споживача про організацію як про компанію. Є можливість змінити цю інформацію і використовувати її для просування бренду в Інтернеті.

Сайт відкриває широкі можливості для залучення цільової аудиторії за рахунок впровадження різних інструментів інтернет-маркетингу: SEO просування, контекстна реклама, розсилка по електронній пошті — це лише мала частина того, що можливо використовувати для підвищення інтересу потенційних клієнтів до компанії.

Веб-сторінки можуть і будуть виглядати дуже по-різному, але всі вони, як правило, мають загальні стандартні компоненти, якщо тільки сторінка не відображає повноекранне відео або гру, не є частиною художнього проекту або просто погано структурована:

1) Назва. Зазвичай у верхній частині є велика панель з назвою організації, логотипом і, можливо, слоганом. Дана інформація зазвичай залишається незмінною від однієї веб-сторінки до іншої.

2) Панель навігації — це посилання на основні розділи сайту, зазвичай представлені кнопками меню, посиланнями або вкладками. Як і назва, цей вміст, як правило, залишається незмінним від однієї веб-сторінки до іншої. Неправильна навігація на вашому сайті призведе до плутанини та розчарування користувачів. Більшість веб-дизайнерів вважають панель навігації частиною заголовка, а не окремим компонентом, але це не є обов'язковою вимогою. Деякі також стверджують, що наявність двох окремих функцій кращим рішенням, тому що програми читання з екрану можуть краще зчитувати дві функції, якщо вони розділені.

3) Основний зміст — це велика область в центрі, яка містить більшу частину унікального вмісту веб-сторінки: відео, яке хоче переглянути користувач, історія яку він читає, карта, заголовки новин тощо. Це та частина сайту, яка, зазвичай, буде змінюватися від сторінки до сторінки.

4) Бічна панель — це деяка периферійна інформація, посилання, цитати, оголошення тощо. Зазвичай це пов'язано з тим, що міститься в основному контенті (наприклад, на сторінці новинної статті бічна панель може містити біографію автора або посилання на відповідні статті), але бувають також випадки, коли можна знайти деякі повторювані елементи, наприклад, додаткову систему навігації.

5) Нижній колонтитул — це панель внизу сторінки, яка зазвичай містить дрібний шрифт, повідомлення про авторські права або контактну інформацію. Це місце для розміщення загальної інформації (наприклад, назви), але зазвичай ця інформація не є критичною, а являється вторинною по відношенню до самого веб-сайту. Нижній колонтитул також іноді використовується в цілях SEO, надаючи посилання для швидкого доступу до популярного контенту.

У міру розвитку сучасних веб-технологій переважна більшість вебсайтів використовують інтерфейсні бібліотеки, фреймворки і плагіни, які полегшують управління HTML, CSS і JavaScript, що відображаються в браузері. З високого рівня вони можуть бути згруповані в три категорії:

1) бібліотеки: фреймворки JavaScript, існує безліч популярних бібліотек JavaScript, які надають широкий спектр функціональних можливостей і підходів. Деякі з найбільш бажаних-jQuery, Angular, React, тощо.

2) адаптивні бібліотеки: фреймворки зі зростанням популярності мобільних пристроїв кілька інтерфейсних бібліотек компонентів надають адаптивні ґрид-системи і готові компоненти користувацького інтерфейсу, що допомагають зробити веб-сайти зручними для мобільних пристроїв. Найпопулярнішими з них є Bootstrap і Foundation.



3) плагіни: тисячі віджетів користувальницького інтерфейсу, таких як слайдери, діаграми, сітки даних, датчики, анімація, елементи управління введенням тощо. Ці плагіни можуть варіюватися від завантаження на кожній сторінці вашого сайту до всього лише декількох сторінок.

Хоча сучасний веб-сайт, як правило, реалізує лише одну основну бібліотеку JavaScript та один адаптивний фреймворк, вони, швидше за все, матимуть багато плагінів на основі JavaScript. Для великих або сильно інтерактивних сайтів це може бути більше десятка або двох.

## **2.2. Серверна частина програми.**

Метою дипломної роботи є створення програми організації роботи з відвідувачами спортивно-оздоровчого комплексу з можливістю ідентифікації в системі, додавання, редагування та видалення даних, а також надання доступу необхідним користувачам. Проектування програми здійснюється з двох частин. Спочатку проектується система управління базою даних(серверна частина), а потім зовнішній інтерфейс, в нашому випадку ним виступає веб-сайт(клієнтська частина).

Перед створенням самої бази даних необхідно розробити структурний алгоритм роботи програми (рис 2.2).



Рисунок 2.2 — Схема алгоритма програми

Спочатку здійснюється з'єднання з базою даних. В базу заноситься окремо заноситься таблиця з інформацією про користувачів(рис. 2.3) та безпосередньо таблиця з програмами тренувань (рис. 2.5).

Згідно вимог замовника таблиця з інформацією про користувачів повинна мати таку структуру: прізвище та ім'я користувача, фотографію(за бажанням), електронна адреса та унікальний пароль.

Users	
PK	id
	first_name
	last_name
	email
	photo
	password

Рисунок 2.3 — Концептуальна модель таблиці з інформацією про користувачів

Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Null	Default
123 id	1	serial						[v]	nextval('users_id_seq'::regclass)
ABC first_name	2	varchar					default	[v]	
ABC last_name	3	varchar					default	[v]	
ABC photo	4	varchar					default	[v]	
ABC email	5	varchar					default	[v]	
ABC password	6	varchar					default	[v]	

Рисунок 2.4 — Вигляд таблиці з інформацією про користувачів

Для користувачів які мають статус «Тренер» необхідно надати змогу додавати програми тренувань та зберігати їх в базі даних. Згідно вимог замовника таблиця з інформацією про програму тренувань повинна таку структуру: назва програми, опис програми, фотографію в якості аватару, дату складання програми та безпосередньо список тренувальних вправ.

program	
PK	id
	name
	description
	photo
	date
	content

Рисунок 2.5— Концептуальна модель таблиці з інформацією про програму тренувань

Column Name	#	Data type	Length	Precision	Scale	Identity	Collation	Not Null	Default
id	1	serial						[v]	nextval('programs_id_seq':re...
name	2	varchar					default	[v]	
description	3	varchar					default	[v]	
photo	4	varchar					default	[v]	
date	5	varchar					default	[v]	
content	6	json						[v]	

Рисунок 2.6 — Вигляд таблиці з інформацією про програму тренувань

### 2.3. Клієнтська частина системи

Для організації роботи з відвідувачами спортивно-оздоровчого комплексу потрібно створити WEB - додаток, який буде посередником у взаємодії між відвідувачем та тренером, таким чином це буде сприяти підвищенню комфорту відвідувачів, а отже і популярності та іміджу спортивно-оздоровчого комплексу. Розробка сайту програм тренувань дасть можливість скоротити час відвідувачів та

тренерів, а також зменшити кількість паперу, оскільки зникне потреба в заповнюванні різні види програм тренувань від руки, достатньо створити програму тренувань на сайті. Задля відображення функціоналу сайту потрібно розділити можливості на алгоритми (рис 2.7):

- 1) алгоритм надання доступу до створення програм тренувань;
- 2) алгоритм додавання програми тренувань;
- 3) алгоритм перегляду програми тренувань;
- 4) алгоритм редагування програми тренувань.



Рисунок 2.7 — Діаграма алгоритмів сайту

Наступним кроком детально розберемо кожен із алгоритмів. На рисунку 2.8 зображено алгоритм надання доступу до створення програм тренувань.



Рисунок 2.8 — Схема алгоритму надання доступу до створення програми тренувань

Алгоритм надання доступу до створення програм тренувань необхідний для обмеження кількості осіб, що мають доступ до створення програм. Доступ надається

лише в тому випадку, коли користувач ідентифікується як тренер «Тренер». Характеристику алгоритма надання доступу до створення програм тренувань продемонстровано в таблиці 2.1.

Характеристики	Значення
Ім'я процесу	Процес надання доступу до створення програм тренувань
Основні учасники	Тренер, клієнт
Вхідна подія	Ідентифікація в системі
Вихідна подія	Надання доступу надання доступу до створення програми тренувань
Клієнт процесу	Тренер

Таблиця 2.1 Опис алгоритму надання доступу до створення програм тренувань

На рисунку 2.9 зображено алгоритм додавання програми тренувань. Алгоритм додавання програми тренувань є одним із найважливіших, тому що в ньому тренер додає інформацію до бази даних.



Рисунок 2.9 — Схема алгоритму додавання програми тренувань

Щоб додати програму тренувань тренер має заповнити форму в котрій повинен заповнити наступні поля: назва програми, опис програми, фотографію в

якості аватару, дату складання програми та безпосередньо список тренувальних вправ. Після збереження запису клієнт отримує доступ до даної програми. Характеристику алгоритма додавання програм тренувань зазначено в таблиці 2.2.

Характеристики	Значення
Ім'я процесу	Додавання програми тренувань
Основні учасники	Тренер, клієнт
Вхідна подія	Введення даних
Вихідна подія	Збереження програми
Клієнт процесу	Тренер

Таблиця 2.2 Опис алгоритма створення програми тренувань

На рисунку 2.10 відображена схема алгоритму перегляду програм тренувань.

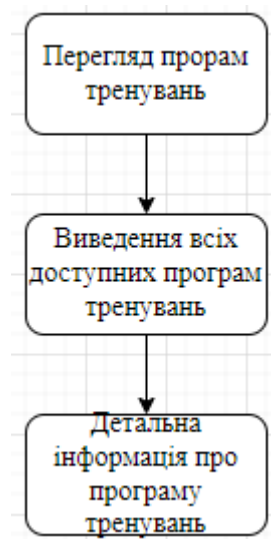


Рисунок 2.10 — Алгоритм перегляду програм тренувань

Перегляд програм тренувань потрібний для надання користувачу можливості переглянути додані програми тренувань. Інформація виводиться у вигляді списку. Опис алгоритма перегляду записів зображено в таблиці 2.3.

Характеристики	Значення
Ім'я процесу	Перегляд програм тренувань
Основні учасники	Тренер, клієнт
Вхідна подія	Перехід на відповідну сторінку
Вихідна подія	Виведення програм тренувань
Клієнт процесу	Тренер

Таблиця 2.3 Опис алгоритма перегляду програм тренування

На рисунку 2.11 зображено алгоритм редагування програм тренувань

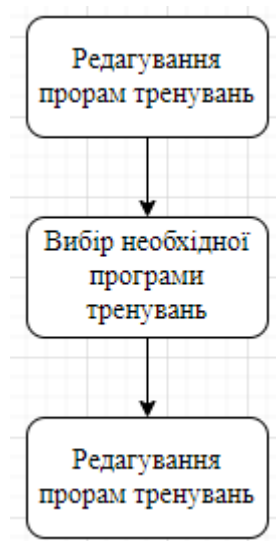


Рисунок 2.11 — Алгоритм редагування програми тренувань

Алгоритм редагування програми тренувань створено для зменшення витрат часу тренера, тому що відповідає необхідність створювати нову програму, достатньо внести зміни у вже існуючу. Опис алгоритма редагування програми тренувань зображено в таблиці 2.4.



Таблиця 2.4 Опис алгоритма редагування програми тренувань

Характеристики	Значення
Ім'я процесу	Редагування програми тренувань
Основні учасники	Тренер, клієнт
Вхідна подія	Вибір необхідної програми
Вихідна подія	Внесення змін у програму
Клієнт процесу	Тренер

#### 2.4. Висновки до розділу

У результаті було створено програму організацію роботи з відвідувачами спортивно-оздоровчого комплексу з можливістю ідентифікації в системі, додавання, редагування та видалення даних, а також надання доступу необхідним користувачам. Проектувалася система управління базою даних(серверна частина), а потім зовнішній інтерфейс, в нашому випадку ним виступає веб-сайт(клієнтська частина).

Було розроблено структурну схему алгоритму роботи програми.

Спочатку здійснюється з'єднання з базою даних. В базу заноситься окремо заноситься таблиця з інфомрацією про користувачів та безпосередньо таблиця з програмами тренувань. Створено WEB - додаток, який буде посередником у взаємодії між відвідувачем та тренером, таким чином це буде сприяти підвищенню комфорту відвідувачів.

Було розроблено алгоритмі надання доступу до створення програм тренувань;

- 1) алгоритм надання доступу до створювання програм тренувань;
- 2) алгоритм додавання програми тренувань;
- 3) алгоритм перегляду програми тренувань;
- 4) алгоритм редагування програми тренувань.

Алгоритм надання доступу до створення програм тренувань необхідний для обмеження кількості осіб, що мають доступ до створення програм.

Алгоритм додавання програми тренувань є одним із найважливіших, тому що в ньому тренер додає інформацію до бази даних.

Алгоритм редагування програми тренувань створено для зменшення витрат часу тренера, тому що відпадає необхідність створювати нову програму, достатньо внести зміни у вже існуючу.

## Розділ 3. Програмна реалізація розроблених алгоритмів тренувань

### 3.1. Вибір технологій для програмної реалізації

HTML-код забезпечує правильне форматування тексту і зображень для вашого інтернет-браузера. Без HTML браузер не знав би, як відображати текст у вигляді елементів або завантажувати зображення чи інші елементи. HTML також забезпечує базову структуру сторінки, на яку накладаються каскадні таблиці стилів для зміни її зовнішнього вигляду.

PostgreSQL, часто просто "Postgres", являє собою об'єктно-реляційну систему управління базами даних (ORDBMS) з акцентом на розширюваність і відповідність стандартам. Як сервер баз даних, його основна функція полягає в безпечному зберіганні даних і підтримки кращих практик, а потім в їх видаленні за запитом інших програмних додатків, будь-то на тому ж комп'ютері або на іншому комп'ютері в мережі (включаючи Інтернет). Він може обробляти робочі навантаження, починаючи від невеликих додатків на одній машині і закінчуючи великими додатками, що виходять в Інтернет, з великою кількістю одночасних користувачів. Останні версії також забезпечують реплікацію самої бази даних для забезпечення безпеки і масштабованості.

PostgreSQL реалізує більшість стандартів SQL:2011 та є сумісним з ACID і транзакційним (включаючи більшість операторів DDL), уникаючи проблем з блокуванням за допомогою управління паралелізмом з декількома версіями (MVCC), забезпечує імунітет до брудних зчитувань і повну серіалізуємість; обробляє складні SQL-запити з використанням багатьох методів індексування, які недоступні в інших базах даних; має оновлювані та матеріалізовані уявлення, тригери, зовнішні ключі; підтримує функції і процедури та іншу розширюваність, а також має велику кількість розширень, написаних третіми сторонами.

На доданок до можливості роботи з основними пропрієтарними базами даних і базами даних з відкритим вихідним кодом, PostgreSQL підтримує міграцію з них за допомогою великої стандартної підтримки SQL і доступних інструментів міграції. І якби були використані пропрієтарні розширення, завдяки своїй розширюваності може емулювати багатьох з них за допомогою деяких вбудованих і сторонніх розширень сумісності з відкритим вихідним кодом, таких як для Oracle. CSS — це аббревіатура каскадних таблиць стилів.

CSS — це мова таблиць стилів, яка використовується для опису подання {документа, написаного на мові розмітки, зазвичай HTML. CSS визначає, як макет і зміст веб-сторінки повинні відображатися на екрані, папері або інших носіях. CSS заощаджує багато роботи, оскільки він керує макетом декількох веб-сторінок одночасно.

Документ зазвичай являє собою текстовий файл, структурований з використанням мови розмітки, наприклад HTML. Представлення документа означає перетворення його в придатну для використання форму, представлену візуально на екрані комп'ютера через веб-браузер, такий як Chrome, Firefox, Opera, Microsoft Edge та інші. Веб-браузери застосовують правила CSS до документа, щоб вплинути на те, як вони відображаються.

Ще одна перевага полягає в тому, що ми можемо користуватися гнучкістю каскадних таблиць, де представлений наш контент. Ми можемо застосовувати певні правила, коли сторінка переглядається на екрані, але інший набір правил буде використовуватися при друку сторінки. Ми також можемо налаштувати відображення для різних розмірів екрану.

Наприклад, ми можемо формувати наш зміст одним способом для користувача, який переглядає його на великому екрані робочого столу, і іншим способом, якщо він переглядає його на меншому екрані планшета.

Крім того, люди можуть створити свій власний особистий CSS стиль, щоб перевизначити результат відображення. Це часто робиться з міркувань доступності. Наприклад, хтось з порушенням зору може створити набір правил для збільшення

розміру тексту на сторінці, або хтось з дальтонізмом може перевизначити кольори для певних елементів, щоб вони могли краще їх ідентифікувати.

JavaScript — це мова програмування, спочатку призначена для взаємодії з елементами веб-сторінок. У веб-браузерах JavaScript складається з трьох основних частин:

- ECMAScript забезпечує основну функціональність.
- Об'єктна модель документа (DOM) надає інтерфейси для взаємодії з елементами на веб-сторінках
- Об'єктна модель браузера (BOM) надає API браузера для взаємодії з веб-браузером.
- JavaScript дозволяє додати інтерактивність до веб-сторінки.

JavaScript часто використовується разом з HTML і CSS для розширення функціональних можливостей веб-сторінки, таких як перевірка форм, створення інтерактивних карт і відображення анімованих діаграм.

Коли веб-сторінка завантажується, тобто після завантаження HTML і CSS, середа розробки JavaScript у веб-браузері виконує код JavaScript. Потім код JavaScript змінює HTML та CSS для динамічного оновлення інтерфейсу користувача.

Мова, заснована на прототипах, як JavaScript, не робить такої різниці: вона просто має об'єкти. Ми можемо вказати властивості для будь-якого об'єкта; ми можемо зробити це при їх створенні або під час виконання.

Таким чином, JavaScript ідеально підходить для спілкування з серверами, так як сервери можуть змінювати структури даних, і код все одно буде працювати (в більшості випадків).

Будь-який об'єкт може стати прототипом іншого об'єкта. У цьому випадку це означає, що другий об'єкт буде розділяти властивості першого об'єкта (це не означає, що значення повинні бути однаковими).

React — це JavaScript-фреймворк, який використовується для побудови графічного інтерфейсу веб-додатків. Мобільні додатки або односторінкові веб-

додатки створюються за допомогою цієї бібліотеки JavaScript для веб - платформи. Програмний код React складається з компонентів, які можуть бути класифіковані як функціональні компоненти та компоненти на основі класів. Він реалізований з використанням об'єктної моделі документа (DOM), віртуального DOM, JSX (JavaScript XML) і методів життєвого циклу, таких як "render", "shouldComponentUpdate", 'componentDidMount' і "componentWillUnmount".

React дозволяє повторно використовувати компоненти, які були розроблені в інших додатках, що використовують ту ж функцію. Можливість повторного використання компонента є явною перевагою для розробників.

Компонент React створити простіше, оскільки він використовує JSX), опціональне розширення синтаксису JavaScript, яке дозволяє комбінувати HTML з JavaScript. JSX - це відмінна суміш JavaScript і HTML . Він робить весь процес написання структури сайту зрозумілішим. Крім того, розширення також значно спрощує рендеринг декількох функцій. Хоча JSX може бути не найпопулярнішим розширенням синтаксису, воно довело свою ефективність при розробці спеціальних компонентів або додатків великого обсягу.

React ефективно оновлює процес DOM (об'єктна модель документа). Інструмент дозволяє створювати віртуальні DOM і розміщувати їх в пам'яті. В результаті кожен раз, коли відбувається зміна в реальному DOM, віртуальний змінюється негайно. Ця система буде перешкоджати тому, щоб фактичний DOM не форсувати постійні оновлення. Як результат, не страждатиме швидкість вашої програми.

React дозволяє створювати призначені для користувача інтерфейси, до яких можна отримати доступ з різних пошукових систем. Ця особливість є величезною перевагою, тому що не всі JavaScript-фреймворки оптимізовані під SEO. Оскільки React може прискорити ваш додаток, він також може поліпшити результати SEO. Швидкість завантаження грає важливу роль в SEO-оптимізації.

NodeJS — це кроссплатформлене середовище виконання Javascript з відкритим вихідним кодом, розроблене на JavaScript в V8 Chrome. Це полегшене середовище, що використовується для розробки веб-додатків на стороні сервера. Він

в основному використовується для створення великомасштабних додатків, в основному для потокових веб-сайтів, односторінкових та інших веб-додатків.

Node.js використовує керовану подіями неблокуючу модель введення/виведення, яка робить її придатною для додатків, що працюють з великими обсягами даних в реальному часі.

Node.js — це платформа з відкритим вихідним кодом. Це означає, що правовласник надав різні права на вивчення, редагування і поширення програмного забезпечення будь-кому будь-якою метою.

Оскільки він використовує механізм подій, Node.js має високу масштабованість і допомагає сервера в неблокуючій відповіді. Node.js використовує процес зациклення подій, це означає, що він може слідувати однопоточній моделі. Це допомагає одному користувачеві обробляти більше одного запиту

Модулі Node.js представляють кілька функцій, згрупованих в один або кілька файлів JS. Всі ці модулі мають унікальний контекст і не заважають роботі інших модулів. Ці модулі дозволяють повторно використовувати код і підвищують зручність використання.

Три модуля, які пропонує Node.js:

- основні модулі.
- локальні модулі;
- сторонні модулі.

Node.js — це полегшений фреймворк, коли модулі групують абсолютні мінімальні функціональні можливості. Як правило, ці модулі завантажуються відразу після запуску процесу Node. Єдине, що потрібно зробити, це імпортувати всі ці основні модулі, щоб використовувати їх в коді.

Локальні модулі створюються локально користувачем або розробником програмного забезпечення. Всі такі модулі можуть мати кілька функцій, згрупованих в різні файли і папки. І все це можна поширювати в суспільстві Node.js за допомогою Node Package Manager.

Ви можете легко використовувати ці модулі, завантаживши їх через Node Package Manager. Більш того, як правило, ці модулі розробляються іншими розробниками, і будь-хто може використовувати їх безкоштовно.

### 3.2. Обґрунтування вибору мови

«HTML» (від англ. «HyperText Markup Language» - «мова розмітки гіпертексту») - стандартна мова розмітки документів у мережі Інтернет. Більшість веб-сторінок створюються за допомогою мови «HTML». Мова «HTML» інтерпретується браузером і відображається у вигляді документа, в зручній для користувача формі. Великим недоліком даної мови є те, що він не підтримує реалізацію таких функцій, як коментарі, реєстрація на сайті, відправка листа адміністратору і робота з базою даних.

Для виконання управління базами даних застосовують систему PostgreSQL. Клієнтська та програмно-апаратна частина веб-сайту написано за допомогою бібліотеки React та Node.js відповідно. Візуальне стилізування сайту створене засобами таблиць стилів CSS, основна логіка елементів сайту реалізується за допомогою надбудови мови JavaScript — React.js.

JavaScript – це мова програмування, що дозволяє зробити Web -сторінку інтерактивною, тобто такою що реагує на дії користувача.

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web -браузер. Іншими словами, код програми вбудовується в HTML - документ і виконується на боці клієнта. Для виконання програми не потрібно навіть перезавантажувати Web -сторінку, всі програми виконуються в відповідь на будь-яку подію. Наприклад,



перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуванім, заборонити відправлення даних.

JavaScript - об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить у браузерях як мова сценаріїв для надання інтерактивності веб-сторінкам.

Основні архітектурні риси:

- 1) динамічна типізація,
- 2) автоматичне керування пам'яттю,
- 3) прототипне програмування,
- 4) функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легкою для використання непрограмістами. JavaScript має низку властивостей об'єктно-орієнтованої мови, але реалізоване в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними об'єктно-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання - що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript у порівнянні з мовою Сі має корінні відмінності:

- 1) об'єкти, з можливістю інтроспекції;
- 2) функції як об'єкти першого класу;
- 3) автоматичне приведення типів;
- 4) автоматичне прибирання сміття;
- 5) анонімні функції.

У мові відсутні такі корисні речі, як:

- модульна система - JavaScript не надає можливості управляти залежностями та ізоляцією областей видимості;

- стандартна бібліотека - зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками вводу/виводу, базових типів для бінарних даних;

- стандартні інтерфейси до веб-серверів та баз даних;

- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

### Семантика і синтаксис

Синтаксис мови JavaScript дуже нагадує синтаксис Сі і Java, семантично ж мова набагато ближче до Self, Smalltalk або навіть Ліспу.

В JavaScript :

- 1) всі ідентифікатори Реєстрозалежні,
- 2) в назвах змінних можна використовувати літери, підкреслення, символ долара, арабські цифри,
- 3) назви змінних не можуть починатися з цифри,
- 4) для оформлення однорядкових коментарів використовуються / /, багаторядкові і внутрішньорядкові коментарі починаються з / \* і закінчуються \* /.

## Структура мови

Структурно JavaScript можна представити у вигляді об'єднання трьох частин, що чітко різняться одна від одної :

- ядро (ECMAScript ),
- об'єктна модель браузера (Browser Object Model або BOM),
- об'єктна модель документа (Document Object Model або DOM).

Об'єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежного від мови інтерфейсу документа.

## Ядро

ECMAScript не є браузерною мовою і насправді в ній не визначаються методи введення і виведення інформації. Це скоріше основа для побудови скриптових мов. Специфікація ECMAScript описує типи даних, інструкції, ключові і зарезервовані слова, оператори, об'єкти, регулярні вирази, не обмежуючи авторів похідних мов від розширення їх новими складовими.

## Об'єктна модель браузера

Об'єктна модель браузера - браузероспецифічна частина мови, яка являється прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - керування вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом BOM. Об'єктна модель браузера на даний момент не стандартизована, проте специфікація знаходиться в розробці WHATWG та W3C. Крім управління вікнами, в рамках об'єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей:

- керування фреймами;
- підтримка затримки у виконанні коду і зациклювання з затримкою;
- системні діалоги;
- управління адресою відкритої сторінки;
- управління інформацією про браузер;

управління інформацією про параметри монітора;  
обмежене керування історією перегляду сторінок;  
підтримка роботи з HTTP cookie.

#### Об'єктна модель документа

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM документом можна поставити у відповідність дерево об'єктів, які мають ряд властивостей, які дозволяють робити з ним різні маніпуляції:

- 1) отримання вузлів;
- 2) зміна вузлів;
- 3) зміна зв'язків між вузлами;
- 4) видалення вузлів.

#### Область застосування

Використання на веб-сторінках;

- Розташування всередині сторінки.
- Розташування всередині тега

Специфікація HTML описує набір атрибутів, використовуваних для завдання обробників подій.

Використання коду JavaScript в контексті розмітки сторінки в рамках ненав'язливого JavaScript розцінюється як погана практика.

#### Винесення в окремий файл

Є і третя можливість підключення JavaScript - написати скрипт в окремому файлі, а потім підключити його за допомогою конструкції

Тег `script`, що широко використовується для підключення до сторінки JavaScript, має декілька атрибутів.

Обов'язковий атрибут `type` для вказівки MIME-типу вмісту.

Проте, згідно специфікації HTML 4.01 в якості значення `type` повинно бути зазначено застаріле `"text/javascript"`. Тому що JavaScript є мовою програмування за

замовчуванням у всіх браузерах, починаючи з Netscape 2, Дуглас Крокфорд дотримується думки про недоцільність використання атрибута `type`, рекомендуючи вказувати його в XHTML.

Необов'язковий атрибут `src`, що приймає в якості значення адресу до файлу зі скриптом.

Необов'язковий атрибут `charset`, використовуваний разом з `src` для вказівки кодування, що використовується для зовнішнього файлу.

Необов'язковий атрибут `defer`, що використовується для того, щоб показати, що скрипт не генерує ніякого вмісту (що означає, зокрема, те, що в цьому скрипті відсутній виклик `document.write ()`).

При цьому атрибут `language="JavaScript"`, незважаючи на його активне використання (у 2008 році цей атрибут був найбільш часто використовуваний у тегах `<script>`), відноситься до нерекомендованих (`deprecated`), відсутній в DTD, тому вважається некоректним.

## Букмарклети

JavaScript використовується для створення невеликих програм, що розміщуються в закладках браузера. При цьому використовуються URL-адреси зі специфікатором `javascript`.

## Користувацькі скрипти в браузері

Користувацькі скрипти в браузері - це програми, написані на JavaScript, що виконуються в браузері користувача при завантаженні сторінки. Вони дозволяють автоматично заповнювати форми, переформовувати сторінки, приховувати небажаний вміст та вбудовувати бажаний для відображення вміст, змінювати поведінку клієнтської частини веб-додатків, додавати елементи керування на сторінку і т. д.

Для управління призначеними для користувача скриптами в Mozilla Firefox використовується розширення Greasemonkey; Opera надає засоби підтримки

користувальницьких скриптів і можливості для виконання ряду скриптів Greasemonkey. Деякі скрипти Greasemonkey можуть виконуватися в Google Chrome при використанні Greasemental.

### 3.3. Тестування програми

Програма для роботи з відвідувачами спортивно-оздоровчого комплексу реалізована за допомогою HTML — для розмітки гіпертексту, CSS — каскадних таблиць стилів, надбудови над JavaScript — React та фреймворку Node.js. Адаптивність сайту виконана за допомогою каскадних таблиць, тобто налаштування таблиць змінює відображення контенту залежно від діагоналі екрану. Завдяки цьому сайт адекватно відображається на пристроях з різним розширенням екрану: комп'ютерах, ноутбуках, планшетах, телефонах тощо.

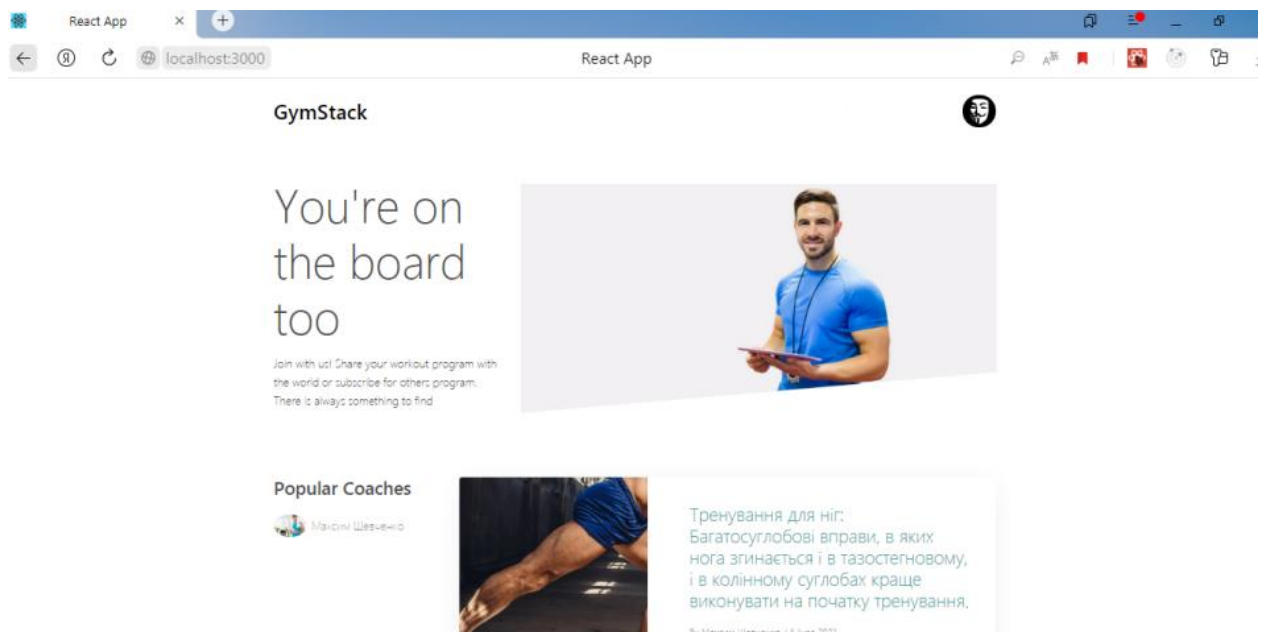


Рисунок 3.1 — Головна сторінка при розширенні 3840 x 2160

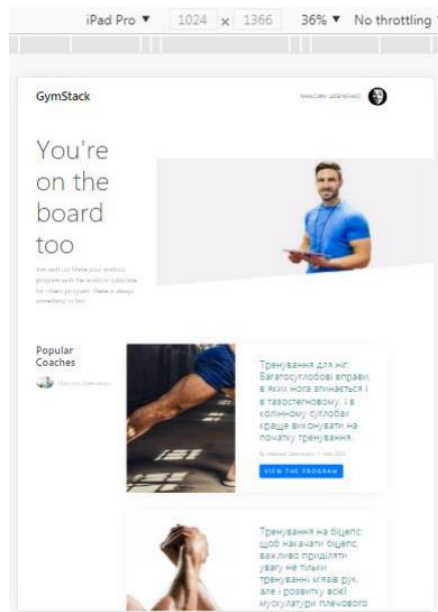


Рисунок 3.2 — Головна сторінка при розширенні 1024 x 1366

На головній сторінці сайту розташовано: блок з кнопками реєстрації та авторизації в системі або профіль користувача при автоматичному вході в систему, кнопки налаштування та виходу з профілю, блок з останніми доданими програмами тренувань з кнопками їх перегляду, кнопка для перегляду всіх доступних програм тренувань.

Щоб зареєструватися в системі необхідно заповнити форму реєстрації.

# Registration

First name	Last name
<input type="text" value="Vladislav"/>	<input type="text" value="Tieriekhov"/>
Photo link	
<input type="text" value="http://linkToPhoto.com"/>	
Email	
<input type="text" value="scienceintegro@gmail.com"/>	
Password	
<input type="password" value="*****"/>	
<input type="button" value="SIGN UP"/>	

Рисунок 3.3 — Форма реєстрації користувача

Всі поля форми на яких стоять обмеження повинні бути заповненими згідно підказкам. Деякі з них запобігають введенню некоректного формату даних. Наприклад, пароль повинен складатися мінімум з 5 символів. Потрібно звернути увагу на те, що електронна адреса повинна відповідати всім критеріям валідного емейлу При правильному заповненні полів користувач ввійде в систему, або на екран з'явиться помилка з підказками по неправильному заповненню даних



# Registration

First name	Last name
<input type="text" value="Vladyslav"/>	<input type="text" value="Tieriekhov"/>
Photo link	
<input type="text" value="http://linkToPhoto.com"/>	
Email	
<input type="text" value="scien"/>	
<p>"email" length must be at least 6 characters long</p>	
Password	
<input type="password" value="***"/>	
<p>"password" length must be at least 5 characters long</p>	
<input type="button" value="SIGN UP"/>	

Рисунок 3.4 — Неправильно введені дані  
Щоб увійти в систему достатньо заповнити форму авторизації.

# Login form

Your e-mail

Your password

LOG IN

Рисунок 3.5 — Форма авторизації користувача

Якщо користувач матиме статус «Тренер» то при натисканні кнопки «Профіль» він потрапляє до налаштувань свого профілю, де може редагувати свої дані. У тренера є доступ до блоку з переглядом ,додаванням та редагуванням програм тренувань. Якщо користувач матиме статус «Клієнт» до блоку управління програмами тренувань він не матиме доступу.

Tieriexhov Vladislav



Activity

[VIEW YOUR PROGRAMS](#)

[CREATE YOUR PROGRAM](#)

[BACK TO HOMEPAGE](#)

First name

Максим

Last name

Шевченко

Photo link

<https://anatomy-fitness.ru/wp-content/uploads/2015/09/сам-себе-тренер.jpg>

Email

scienceintegro@gmail.com

[SAVE CHANGES](#)

Рисунок 3.6 — Профіль тренера

Іван Іванов



First name

Іван

Last name

Іванов

Photo link

<https://pixelbox.ru/wp-content/uploads/2021/05/ava-vk-animal-91.jpg>

Email

scienceintegro@gmail.com

[SAVE CHANGES](#)

Рисунок 3.7 — Профіль клієнта

Для створення програми тренувань тренер повинен натиснути кнопку «Створити програму» та заповнити відповідну форму.

# Create program

Program name

Program photo

Description

Exercise	Sets x Reps
<input type="text" value="Pushups"/>	<input type="text" value="3x15"/>

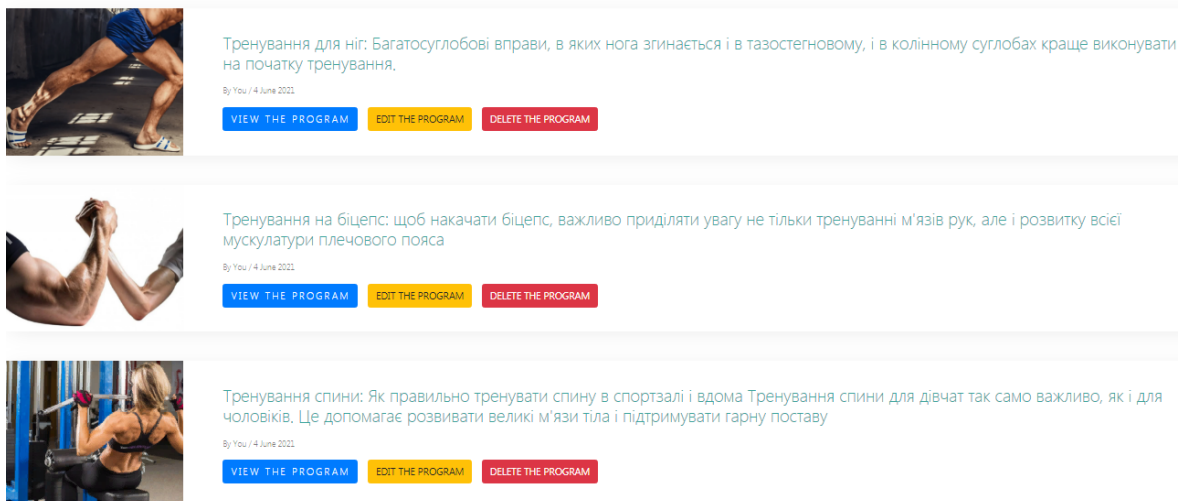
**ADD ITEM**

**CREATE**

Рисунок 3.8 — Форма для створення програми тренувань

Для перегляду вже створених користувачем програм необхідно натиснути кнопку «Переглянути ваші програми». Програми тренувань відкриються у вигляді списку з назвою програми та блоком кнопок: переглянути програму, видалити програму та редагувати програму.

## Your Programs



The image shows three program cards. Each card has a thumbnail image on the left, a title and description on the right, a date 'By You / 4 June 2021', and three buttons: 'VIEW THE PROGRAM' (blue), 'EDIT THE PROGRAM' (yellow), and 'DELETE THE PROGRAM' (red).

- Card 1:** Thumbnail of a person performing a squat. Title: "Тренування для ніг: Багатосуглобові вправи, в яких нога згинається і в тазостегновому, і в колінному суглобах краще виконувати на початку тренування." Buttons: VIEW THE PROGRAM, EDIT THE PROGRAM, DELETE THE PROGRAM.
- Card 2:** Thumbnail of a person performing a bicep curl. Title: "Тренування на біцепс: щоб накачати біцепс, важливо приділяти увагу не тільки тренуванні м'язів рук, але і розвитку всієї мускулатури плечового пояса." Buttons: VIEW THE PROGRAM, EDIT THE PROGRAM, DELETE THE PROGRAM.
- Card 3:** Thumbnail of a person performing a back exercise on a machine. Title: "Тренування спини: Як правильно тренувати спину в спортзалі і вдома Тренування спини для дівчат так само важливо, як і для чоловіків. Це допомагає розвивати великі м'язи тіла і підтримувати гарну поставу." Buttons: VIEW THE PROGRAM, EDIT THE PROGRAM, DELETE THE PROGRAM.

Рисунок 3.9 — Сторінка перегляду програм тренувань

При натисненні кнопки «Переглянути програму» користувач потрапить на сторінку з більш детальною інформацією про програму тренувань.

## Тренування для ніг

By Tieriekhov Vladyslav

Description : Багатосуглобові вправи, в яких нога згинається і в тазостегновому, і в колінному суглобах краще виконувати на початку тренування.



### Content

Присідання зі штангою на спині - 3x15

Підйоми на носок з гантелею - 2x20 на ногу

Підйоми корпусу на GHD - 3x25

BACK TO HOME PAGE

Рисунок 3.10 — Детальна інформація про програму тренувань

При натисненні кнопки «Редагувати програму» користувач отримає доступ до форми редагування. Тренер може редагувати зображення програми, назву та опис програми, створювати нові вправи та видаляти існуючі.

## Edit program

Program name

Тренування для ніг

Program photo

Description

Багатосуглобові вправи, в яких нога згинається і в тазостегновому, і в колінному суглобах краще виконувати на початку тренування.

Exercise	Sets x Reps
<span style="border: 1px solid #ccc; padding: 2px 5px;">🔍</span> Pushups	<span style="border: 1px solid #ccc; padding: 2px 5px;">□</span> 3x15
<span style="background-color: #007bff; color: white; padding: 5px 15px; border-radius: 5px;">ADD ITEM</span>	
Присідання зі штангою на спині - 3x15	<span style="background-color: #dc3545; color: white; padding: 5px 10px; border-radius: 5px;">REMOVE</span>
Підйоми на носок з гантелею - 2x20 на ногу	<span style="background-color: #dc3545; color: white; padding: 5px 10px; border-radius: 5px;">REMOVE</span>
Підйоми корпусу на GHD - 3x25	<span style="background-color: #dc3545; color: white; padding: 5px 10px; border-radius: 5px;">REMOVE</span>
<span style="background-color: #28a745; color: white; padding: 10px 30px; border-radius: 5px; font-weight: bold;">CHANGE</span>	

Рисунок 3.11 — Сторінка редагування програми тренувань

Якщо користувач авторизувався зі статусом «Клієнт», він має доступ лише до перегляду доступних програм тренувань.

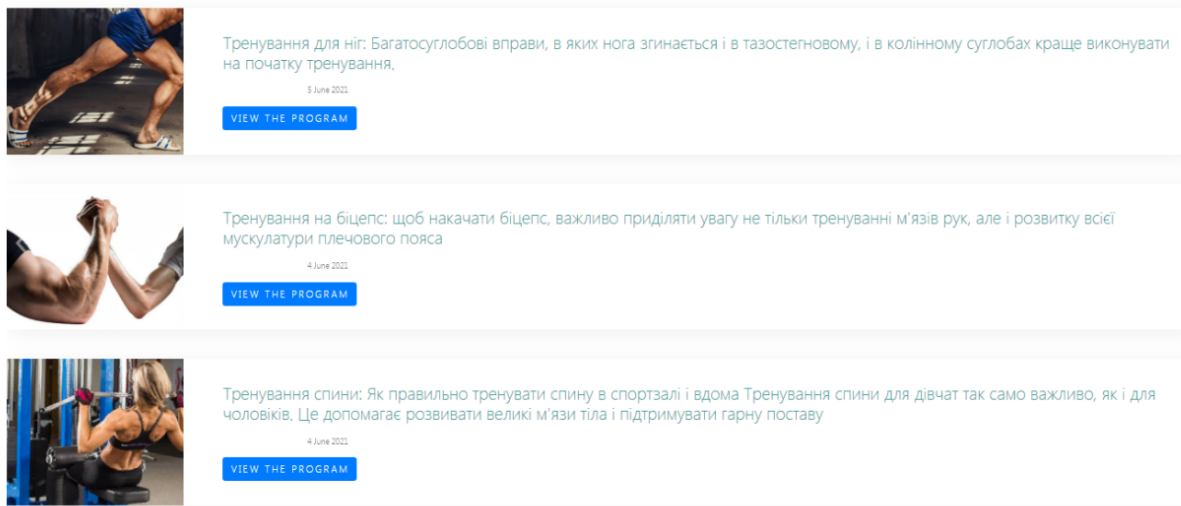


Рисунок 3.12 — Сторінка перегляду програм тренувань

### 3.4. Висновки до розділу

Мова «HTML» інтерпретується браузером і відображається у вигляді документа, в зручній для користувача формі. Великим недоліком даної мови є те, що він не підтримує реалізацію таких функцій, як коментарі, реєстрація на сайті, відправка листа адміністратору і робота з базою даних. Отже, мова «HTML» підходить для створення даного програмного продукту лише частково.

Адаптивність сайту виконана за допомогою каскадних таблиць, тобто налаштування таблиць змінює відображення контенту залежно від діагоналі екрану. Завдяки цьому сайт адекватно відображається на пристроях з різним розширенням екрану: комп'ютерах, ноутбуках, планшетах, телефонах.

Всі поля форми реєстрації на яких стоять обмеження повинні бути заповненими згідно підказкам. Деякі з них запобігають введенню некоректного формату даних. Електронна адреса повинна відповідати всім критеріям валідного емейлу. При правильному заповненні полів користувач увійде в систему, або на екран з'явиться помилка з підказками по неправильному заповненню даних.

Якщо користувач має статус «Тренер» то при натисненні кнопки «Профіль» він потрапляє на сторінку з налаштуваннями свого профілю, де може редагувати свої дані. Тренеру надається доступ до блоку з додаванням, переглядом та редагуванням програм тренувань. Тренер може редагувати зображення програми, назву та опис програми, створювати нові вправи та видаляти існуючі.



## ВИСНОВКИ

Підводячи підсумок випускої роботи, слід зазначити, що інтегрування автоматизації спортивно-оздоровчого комплексу повинно здійснюватися, починаючи з найважливішої частини його роботи, автоматизація якої дасть можливість швидко отримати гарний результат. Під час розробки веб - додатку потрібно забезпечити можливість роботи за новою, та за старою технологіями, щоб не втручатися в функціонування спортивно-оздоровчого комплексу.

Аналіз існуючих рішень організації роботи з клієнтами спортивно-оздоровчого комплексу дозволяє констатувати, що розглянуті програмні рішення не підходять для вирішення поставленого завдання, оскільки їх функціонал не відповідає потребам замовника. На основі цього було розроблено програму автоматизації спортивно-оздоровчого комплексу для роботи з відвідувачами спортивного залу, яка відповідає необхідним вимогам:

- 1) механізм ідентифікації користувача у системі;
- 2) створення та зберігання програм тренувань у цифровому форматі;
- 3) додавання, видалення, перегляд програм тренувань;
- 4) надання доступу до програм тренувань;
- 5) перевірка правильності введення інформації.

В процесі виконання випускної роботи були обрані технології для програмної реалізації сайту, обрано мову розмітки гіпертексту — HTML, систему управління базами даних PostgreSQL,. Клієнтська та програмноапаратна частина веб-сайту написано за допомогою бібліотеки React та Node.js відповідно. Візуальне оформлення сайту здійснюється за допомогою каскадних таблиць стилів CSS, Основна логіка веб-сайту реалізується засобами мови програмування JavaScript. У ході впровадження інформаційної системи була проведена консультація для користувачів програмного додатку.

Впровадження в дію “Автоматизації спортивно-оздоровчого комплексу” дозволить частково автоматизувати процес роботи, що збільшить прибутковість організації та лояльність клієнтів. Це дозволить знизити навантаження на

тренерській склад та дозволить клієнтам користуватися послугами спортивного залу в зручному форматі. З використанням інформаційної займатися спортом можливо самостійно, оскільки клієнт має доступ до необхідних програм тренувань в зручному форматі, навіть за відсутності тренера.

У результаті в кваліфікаційній роботі розроблено програму для організації роботи з відвідувачами спортивного-оздоровчого комплексу, що виконує наступні завдання:

- 1) автоматизує ідентифікацію користувача у системі;
- 2) створює та зберігає програми тренувань у цифровому форматі;
- 3) додає, видаляє, та дає доступ до перегляду програм тренувань;
- 4) надає доступу до програми тренувань;
- 5) перевіряє правильність введення інформації.

## СПИСОК ЛІТЕРАТУРИ

- 1) Система менеджменту якості, Положення про дипломні роботи(проекти) випускників НАУ, СМЯ НАУ П 03.01(10) – 02 – 2017
- 2) Держаній стандарт України, Документація. Звіти у сфері науки і техніки, Структура і правила оформлення ДСТУ 3008-95.
- 3) Н.Купер и К.Джи – "Как создать сайт путеводитель по HTML, CSS, WordPress" Москва : Питер, 2019. - 58 с.
- 4) Д.Дакетт, – "HTML и CSS Разработка и дизайн веб-сайтов ", 2020 –92с.
- 5) М.Тиленс . – "React в действии" – СПб.: Питер, 2019. – 336с.
- 6) К.Дуглас – Как устроен JavaScript – Питер, 2019. – 69 с.
- 7) М.Фаулер –"Рефакторинг кода на JavaScript", 2019. – 110 с.
- 8) Резиг, Бибо, Марас – "Секреты JavaScript ниндзя. " 2019 – 104с.
- 9) Д. Кудрец – "Основы CSS"" 2019 – 104с.
- 10) М. Пацианский – "React.js курс для начинающих." – 2018. – 146с.
- 11) Е.Моргунов – "Postgre. SQL. Основы языка SQL." – БХВ-петербург, 2021. – 44 с.
- 12) Видавництво: "Прометей" С. А. Жданов – "Информационные системы." – 2015. – 76с. 11. Видавництво: "Современные информационные системы " "Современные информационные системы №2"– 2020 – 62с.
- 13) П. Уилтон,Д. Колби – "SQL для начинающих. "- 2006.– 184с.
- 14) Н.Купер и К.Джи – "Как создать сайт путеводитель по HTML, CSS, WordPress" Москва : Питер, 2019. - 58 с.
- 15) Д.Дакетт, – "HTML и CSS Разработка и дизайн веб-сайтов ", 2020 –92с.

## ДОДАТОК

Код головної сторінки:

```
import './HomePage.css';
import { HomeNav } from './additional/HomeNav';
import { HomeInfo } from './additional/HomeInfo';
import { HomePopularCoaches } from './additional/HomePopularCoaches';
import { HomePrograms } from './additional/HomePrograms';
import { User } from '.././interfaces/user.interface';
export const HomePage: React.FunctionComponent;
<{
  interweavings: Array<any>;
  currentUser: User;
  setCurrentUser: Function;
}> = ({ interweavings, currentUser, setCurrentUser }) => {
  return (
    <React.Fragment>
      <div className="homeBody">
        <HomeNav currentUser={currentUser}
          setCurrentUser={setCurrentUser} />
        <HomeInfo />
        <div className="site-section">
          <div className="container">
            <div className="row">
              <HomePopularCoaches />
              <HomePrograms interweavings={interweavings} />
            </div>
          </div>
        </div>
      </div>
    </div>
  )
}
```

```
</React.Fragment>  
  
);  
};
```

Код сторінки реєстрації користувача:

```
import Joi from "joi-browser";  
  
import http from "../services/httpService";  
  
import "./Reg.css";  
  
export const Reg = () => {  
  const [regErrors, setRegErrors] = useState({});  
  const [regFirstName, setRegFirstName] = useState("");  
  const [regLastName, setRegLastName] = useState("");  
  const [regEmail, setRegEmail] = useState("");  
  const [regPhoto, setRegPhoto] = useState("");  
  const [regPassword, setRegPassword] = useState("");  
  const schema = {  
    first_name: Joi.string().required().min(2).max(20),  
    last_name: Joi.string().required().min(2).max(20),  
    email: Joi.string().required().min(6).max(40),  
    password: Joi.string().required().min(5),  
  };  
  const validate = () => {  
    setRegErrors({});  
    let user = {  
      first_name: regFirstName,  
      last_name: regLastName,  
      email: regEmail,  
      password: regPassword,  
    };  
    const result = Joi.validate(user, schema, { abortEarly: false
```

```

});
if (!result.error) {
  return null;
}
const errors = {};
for (let item of result.error.details) {
  errors[item.path[0]] = item.message;
}
return errors;
};
const onClickHandler = async () => {
  const errors = validate();
  if (errors) {
    setRegErrors(errors);
    return;
  }
  let user = {
    first_name: regFirstName,
    last_name: regLastName,
    photo:
      regPhoto ||
      "https://i.ibb.co/Bz7mbcS/36324eb648f2cb507e243e8030e4dc0a.png",
    email: regEmail,
    password: regPassword,
  };
  await http.post(process.env.REACT_APP_API_URL + "reg", user);
  window.location = "/login";
};
return (

```

```

<React.Fragment>
<div className="reg-form">
<form>
<h1 className="reg-header">Registration</h1>
<div className="form-row">
<div className="col">
<label className="reg-label reg-top-label">First
name</label>
<input
onChange={(event) =>
setRegFirstName(event.target.value)}
type="text"
className="form-control reg-input"
placeholder="Andrew"
/>
{regErrors.first_name && (
<div className="alert alert-danger rentAlert">
{regErrors.first_name}
</div>
)}
</div>
<div className="col">
<label className="reg-label reg-top-label">Last
name</label>
<input
onChange={(event) =>
setRegLastName(event.target.value)}
type="text"
className="form-control reg-input"
placeholder="Buhayov"

```

```

/>
{regErrors.last_name && (
<div className="alert alert-danger rentAlert">
{regErrors.last_name}
</div>
)}
</div>
</div>
<div className="form-group reg-bottom-container">
<label className="reg-label"
htmlFor="formGroupRegPhoto">
Photo link
</label>
<input
onChange={(event) =>
setRegPhoto(event.target.value)}
type="text"
className="form-control reg-input"
id="formGroupRegPhoto"
placeholder="http://linkToPhoto.com"
/>
<label className="reg-label"
htmlFor="formGroupRegEmail">
Email
</label>
<input
onChange={(event) =>
setRegEmail(event.target.value)}
type="text"
className="form-control reg-input"

```



```

id="formGroupRegEmail"
placeholder="something@gmail.com"
/>
{regErrors.email && (
<div className="alert alert-danger rentAlert">
{regErrors.email}
</div>
)}
<label className="reg-label"
htmlFor="formGroupRegPassword">
Password
</label>
<input
onChange={(event) => setRegPassword(event.target.value)}
type="password"
className="form-control reg-input"
id="formGroupRegPassword"
/>
{regErrors.password && (
<div className="alert alert-danger rentAlert">
{regErrors.password}
</div>
)}
</div>
<button
onClick={onClickHandler}
type="button"
className="reg-button btn btn-primary"
>
Sign Up

```

```

</button>
</form>
</div>
</React.Fragment>
);
};

```

Код сторінки створення програми тренувань:

```

import Joi from "joi-browser";
import http from "../services/httpService";
import date from "date-and-time";
import "../CreateProgram.css";
let counter = 0;
export const CreateProgram = ({ currentUser }) => {
  const [createProgramName, setCreateProgramName] = useState("");
  const [createProgramDescription, setCreateProgramDescription] =
    useState("");
  const [createProgramPhoto, setCreateProgramPhoto] =
    useState("");
  const [createProgramErrors, setCreateProgramErrors] =
    useState({});
  const [createProgramContent, setCreateProgramContent] =
    useState([]);
  const schema = {
    name: Joi.string().required().min(2).max(20),
    description: Joi.string().required().min(2).max(255),
  };
  const validate = () => {
    setCreateProgramErrors({});
    let program = {
      name: createProgramName,

```

```

description: createProgramDescription,
};
const result = Joi.validate(program, schema, { abortEarly: false
});
if (!result.error && createProgramContent.length > 0) {
  return null;
}
const errors = {};
if (createProgramContent.length === 0) {
  errors["content"] = "Should be at least one item";
}
if (result.errors) {
  for (let item of result.error.details) {
    errors[item.path[0]] = item.message;
  }
}
return errors;
};
const onClickCreateHandler = async () => {
  const errors = validate();
  if (errors) {
    setCreateProgramErrors(errors);
    return;
  }
  let contentObj = [];
  for (let i = 0; i < createProgramContent.length; i++) {
    contentObj.push(createProgramContent[i].value);
  }
  let now = date.format(new Date(), "D MMMM YYYY");
  let program = {

```

```

name: createProgramName,
description: createProgramDescription,
content: contentObj,
photo:
  createProgramPhoto ||
  "https://i.ibb.co/QJZ3Qd8/94610367-images12490520061.jpg",
date: now,
};

const { data } = await http.post(process.env.REACT_APP_API_URL +
"programs", program);
let userEntity = {
  user_id: currentUser.id,
  program_id: data.id,
  role_id: 1,
};
await http.post(process.env.REACT_APP_API_URL + "interweavings",
userEntity);
window.location = "/";
};

const onClickAddHandler = () => {
  const createExercise =
document.getElementById("createProgramExercise");
  const createSteps =
document.getElementById("createProgramSteps");
  let tempCreateContent = [...createProgramContent];
  if (createExercise.value && createSteps.value) {
    let contentItem = `${createExercise.value} -
${createSteps.value}`;
    tempCreateContent.push({
      id: counter,

```

```

value: contentItem,
});
setCreateProgramContet(tempCreateContent);
createExercise.value = "";
createSteps.value = "";
counter++;
}
};
const onClickRemoveHandler = (event) => {
let tempCreateContent = [...createProgramContent];
tempCreateContent = tempCreateContent.filter(
(obj) => obj.id !== Number(event.currentTarget.id)
);
setCreateProgramContet(tempCreateContent);
};
return (
<React.Fragment>
<div className="createProg-form">
<form>
<h1 className="createProg-header">Create program</h1>
<div className="form-group createProg-bottomcontainer">
<label className="createProg-label"
htmlFor="createProgramName">
Program name
</label>
<input
onChange={(event) =>
setCreateProgramName(event.target.value)}
type="text"
className="form-control createProg-input"

```

```

id="createProgramName"
placeholder="Shoulders"
/>
{createProgramErrors.name && (
<div className="alert alert-danger rentAlert">
{createProgramErrors.name}
</div>
)}
</div>
<label className="createProg-label"
htmlFor="createProgramPhoto">
Program photo
</label>
<input
onChange={(event) =>
setCreateProgramPhoto(event.target.value)}
type="text"
className="form-control createProg-input"
id="createProgramPhoto"
placeholder="http://linkToPhoto.com"
/>
<div className="form-group">
<label
className="createProg-label"
htmlFor="createProgramDescription"
>
Description
</label>
<textarea
onChange={(event) =>

```

```

setCreateProgramDescription(event.target.value)
}
id="createProgramDescription"
className="form-control"
resize="none"
aria-label="With textarea"
placeholder="Shoulders day"
></textarea>
{ createProgramErrors.description && (
<div className="alert alert-danger rentAlert">
{ createProgramErrors.description }
</div>
)}
</div>
<div className="d-flex justify-content-sm-between
createProgram-headers">
<h5>Exercise</h5>
<h5>Sets x Reps</h5>
</div>
<div className="form-row">
<div className="col input-group mb-3">
<div className="input-group-prepend">
<span
role="img"
aria-label="bicycleEmoji"
className="input-group-text"
>
&#128170;
</span>
</div>

```

```

<input
id="createProgramExercise"
type="text"
className="form-control"
placeholder="Pushups"
aria-label="Exercise"
aria-describedby="basic-addon1 "
/>
</div>{ " "}
<div className="col input-group mb-3">
<div className="input-group-prepend">
<span
role="img"
aria-label="bicycleEmoji"
className="input-group-text"
>
&#128258;
</span>
</div>
<input
id="createProgramSteps"
type="text"
className="form-control"
placeholder="3x15"
aria-label="SetsxReps"
aria-describedby="basic-addon1 "
/>
</div>
</div>
{createProgramErrors.content && (

```



```
<div className="alert alert-danger rentAlert">
```

Should be at least one item

```
</div>
```

```
)}
```

```
<button
```

```
type="button"
```

```
onClick={onClickAddHandler}
```

```
className="btn btn-primary mb-5"
```

```
>
```

Add item

```
</button>
```

```
<div className="contentItemContainer">
```

```
<div className="container">
```

```
{createProgramContent &&
```

```
createProgramContent.map((item) => (
```

```
<div key={item.id} className="row
```

```
createItemContainer">
```

```
<div className="col-9">
```

```
<span key={item.value} className="contentItem">
```

```
{item.value}
```

```
</span>
```

```
</div>
```

```
<div className="col-3">
```

```
<button
```

```
onClick={onClickRemoveHandler}
```

```
id={item.id}
```

```
type="button"
```

```
className="btn btn-danger
```

```
createProgramRemoveButton"
```

```
>
```

Remove

```
</button>  
</div>  
</div>  
)})  
</div>  
</div>  
<button  
onClick={onClickCreateHandler}  
type="button"  
className="createProg-button btn btn-success"  
>
```

Create

```
</button>  
</form>  
</div>  
</React.Fragment>  
);  
};
```

Код сторінки перегляду програм тренувань:

```
import "./AllPrograms.css";  
  
export const AllPrograms: React.FunctionComponent<{  
  interweavings: Array<Interweaving>;  
  const history = useHistory();  
  const onClickViewHandler = (id: number) => {  
    history.push(/currentProgram/?id=${id});  
  };  
  return (  
    <React.Fragment>
```

```

<h1 className="allProgramsHeader" id="allProgramsHeader">
All Programs
</h1>
{interweavings &&
interweavings.map((item: any) => (
<div
key={item.id}
className="d-block d-md-flex podcast-entry bg-white
mb-5"
data-aos="fade-up"
style={{
boxShadow: "0 5px 40px -10px rgba(0, 0, 0, 0.1)",
borderRadius: "4px",
overflow: "hidden",
}}
>
<div
className="image"
style={{
width: "300px",
height: "auto",
backgroundSize: "cover",
backgroundPosition: "center center",
backgroundImage: url('${item.program.photo}'),
}}
></div>
<div
className="text ml-4"
style={{
width: "calc(100% - 300px)",

```

```

padding: "40px",
}}
>
<h3 className="font-weight-light">
<p
style={{
color: "#3ca59d",
}}
className="homeProgramItemHeader"
>
{item.program.name}: {item.program.description}
</p>
</h3>
<div className="mb-3">
<span className="text-black-opacity-05">
<small
style={{
color: "rgba(0, 0, 0, 0.5)",
fontWeight: 400,
}}
>
By {item.user.first_name}
{item.user.last_name}{" "}
<span className="sep"></span>
{item.program.date}{" "}
</small>
</span>
</div>
<div>
<button

```

```

onClick={() => onClickViewHandler(item.id)}
type="button"
className="btn btn-primary"
>
View the program
</button>
</div>
</div>
</div>
)}}
</React.Fragment>
);
};

```

Код сторінки редагування програм тренувань:

```

import "./EditProgram.css";
let counter = 100;
export const EditProgram = () => {
  const [editProgramName, setEditProgramName] = useState("");
  const [editProgramDescription, setEditProgramDescription] =
  useState("");
  const [editProgramPhoto, setEditProgramPhoto] = useState("");
  const [editProgramErrors, setEditProgramErrors] = useState({});
  const [editProgramContent, setEditProgramContent] = useState([]);
  useEffect(() => {
    const { id } = queryString.parse(window.location.search);
    async function fetchCurrentProgram() {
      const { data } = await
  http.get(`${process.env.REACT_APP_API_URL}programs/${id}`);
      setEditProgramName(data.name);
      setEditProgramDescription(data.description);

```

```

setEditProgramPhoto(data.photo);
const tempContent = [];
for (let item in data.content) {
tempContent.push({
id: Number(item),
value: data.content[item],
});
}
setEditProgramContent(tempContent);
}
fetchCurrentProgram();
}, []);
const schema = {
name: Joi.string().required().min(2).max(20),
description: Joi.string().required().min(2).max(255),
};
const validate = () => {
setEditProgramErrors({});
let program = {
name: editProgramName,
description: editProgramDescription,
};
const result = Joi.validate(program, schema, { abortEarly: false
});
if (!result.error && editProgramContent.length > 0) {
return null;
}
const errors = {};
if (!editProgramContent.length) {
errors["content"] = "Should be at least one item";

```

```

}
for (let item of result.error.details) {
errors[item.path[0]] = item.message;
}
return errors;
};

const onClickChangeHandler = async () => {
const { id } = queryString.parse(window.location.search);
const errors = validate();
if (errors) {
setEditProgramErrors(errors);
return;
}
let contentObj = [];
for (let i = 0; i < editProgramContent.length; i++) {
contentObj[${i}] = editProgramContent[i].value;
}
let now = date.format(new Date(), "D MMMM YYYY");
let program = {
name: editProgramName,
description: editProgramDescription,
content: contentObj,
photo:
editProgramPhoto ||
"https://i.ibb.co/QJZ3Qd8/94610367-images-12490520061.jpg",
date: now,
};
await http.put(http://localhost:5000/programs/${id}, program);
window.location = "/me";
};

```

```

const onClickAddHandler = () => {
  const editExercise =
document.getElementById("editProgramExercise");
  const editSteps = document.getElementById("editProgramSteps");
  let tempEditContent = [...editProgramContent];
  if (editExercise.value && editSteps.value) {
    let contentItem = `${editExercise.value} - ${editSteps.value}`;
    tempEditContent.push({
      id: counter,
      value: contentItem,
    });
    setEditProgramContent(tempEditContent);
    editExercise.value = "";
    editSteps.value = "";
    counter++;
  }
};

const onClickRemoveHandler = (event) => {
  let tempEditContent = [...editProgramContent];
  tempEditContent = tempEditContent.filter(
    (obj) => obj.id !== Number(event.currentTarget.id)
  );
  setEditProgramContent(tempEditContent);
};

return (
  <React.Fragment>
  <div className="createProg-form">
  <form>
  <h1 className="createProg-header">Edit program</h1>
  <div className="form-group createProg-bottom-container">

```



```

<label className="createProg-label"
htmlFor="createProgramName">
  Program name
</label>
<input
value={editProgramName}
onChange={(event) =>
setEditProgramName(event.target.value)}
type="text"
className="form-control editProg-input"
id="editProgramName"
placeholder="Shoulders"
/>
{editProgramErrors.name && (
<div className="alert alert-danger rentAlert">
{editProgramErrors.name}
</div>
)}
</div>
<div className="form-group createProg-bottom-container">
{" "}
<label className="editProg-label"
htmlFor="editProgramPhoto">
  Program photo
</label>
<input
value={editProgramPhoto}
onChange={(event) =>
setEditProgramPhoto(event.target.value)}
type="text"

```

```

className="form-control createProg-input"
id="editProgramPhoto"
placeholder="http://linkToPhoto.com"
/>
</div>
<div className="form-group">
<label className="editProg-label"
htmlFor="editProgramDescription">
Description
</label>
<textarea
onChange={(event) =>
setEditProgramDescription(event.target.value)
}
value={editProgramDescription}
id="editProgramDescription"
className="form-control"
resize="none"
aria-label="With textarea"
placeholder="Shoulders day"
></textarea>
{editProgramErrors.description && (
<div className="alert alert-danger rentAlert">
{editProgramErrors.description}
</div>
)}
</div>
<div className="d-flex justify-content-sm-between
editProgram-headers">
<h5>Exercise</h5>

```

```

<h5>Sets x Reps</h5>
</div>
<div className="form-row">
<div className="col input-group mb-3">
<div className="input-group-prepend">
<span
role="img"
aria-label="bicycleEmoji"
className="input-group-text"
>
&#128170;
</span>
</div>
<input
id="editProgramExercise"
type="text"
className="form-control"
placeholder="Pushups"
aria-label="Exercise"
aria-describedby="basic-addon1 "
/>
</div>{ " "}
<div className="col input-group mb-3">
<div className="input-group-prepend">
<span
role="img"
aria-label="bicycleEmoji"
className="input-group-text"
>
&#128258;

```

```

</span>
</div>
<input
id="editProgramSteps"
type="text"
className="form-control"
placeholder="3x15"
aria-label="SetsxReps"
aria-describedby="basic-addon1"
/>
</div>
</div>
{ editProgramErrors.content && (
<div className="alert alert-danger rentAlert">
Should be at least one item
</div>
)}
<button
type="button"
onClick={ onClickAddHandler }
className="btn btn-primary mb-5"
>
Add item
</button>
<div className="contentItemContainer">
<div className="container">
{ editProgramContent &&
editProgramContent.map((item) => (
<div key={ item.id } className="row
editItemContainer">

```

```

<div className="col-9">
  <span key={item.value} className="contentItem">
    {item.value}
  </span>
</div>
<div className="col-3">
  <button
    onClick={onClickRemoveHandler}
    id={item.id}
    type="button"
    className="btn btn-danger
    editProgramRemoveButton"
  >
    Remove
  </button>
</div>
</div>
))}
</div>
</div>
<button
  onClick={onClickChangeHandler}
  type="button"
  className="editProg-button btn btn-success"
  >
    Change
  </button>
</form>
</div>
</React.Fragment>

```

);  
}