МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет економіки, підприємництва та інформаційних технологій
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2026 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З МОДУЛЕМ КЛАСТЕРИЗАЦІЇ СТРІЧКИ НОВИН

Виконав – студент 4 курсу, групи 41-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Бондаренко Андрій Юрійович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2026

Зміст

ВСТУП 4

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ СОЦІАЛЬНОЇ МЕРЕЖІ 6

1.1. Аналіз предметної області та обґрунтування створення програмного продукту 6

1.2. Порівняльний аналіз аналогів соціальної мережі 7

1.3. Постановка задачі 10

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З МОДУЛЕМ КЛАСТЕРИЗАЦІЇ СТРІЧКИ НОВИН 12

2.1. Визначення функціональних вимог соціальної мережі 12

2.3. Проєктування бази даних 15

2.3. Обґрунтування вибору стеку технологій для реалізації поставлених завдань 17

2.4. Обґрунтування вибору методу кластеризації стрічки новин для користувачів 22

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З МОДУЛЕМ КЛАСТЕРИЗАЦІЇ СТРІЧКИ НОВИН 25

3.1. Реалізація обробки та аналізу даних 25

3.2. Програмна реалізація підсистеми управління базами даних 28

3.3. Програмна реалізація графічного інтерфейсу користувача 30

3.4. Реалізація програмного модуля кластеризації стрічки новин 34

3.5. Тестування роботи веб-додатку для соціальної мережі з модулем кластеризації стрічки новин 36

ВИСНОВКИ 44

СПИСОК ЛІТЕРАТУРИ 46

ДОДАТКИ 50

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

JavaScript – високорівнева мова програмування, що застосовується для розробки інтерактивних компонентів веб-сторінок, проектування мобільних додатків, а також для реалізації серверної частини програмних систем.

REACTJS – бібліотека мови програмування JavaScript з відкритим вихідним кодом, орієнтована на проектування користувацьких інтерфейсів, яка була імплементована у процесі розробки веб-додатку.

HTML – стандартизована мова гіпертекстової розмітки, що забезпечує структурну побудову, організацію та форматування документів у глобальній мережі.

URL – стандартизована система уніфікованих локаторів, призначена для ідентифікації, адресації та визначення місцезнаходження інформаційних ресурсів.

HTTP – мережевий протокол прикладного рівня, базова функція якого полягає у забезпеченні клієнт-серверної взаємодії шляхом передачі даних та обміну гіпертекстовими документами.

MERN – комплексний технологічний стек для веб-розробки, архітектура якого базується на системній інтеграції програмних рішень, а саме бази даних MongoDB, серверного фреймворку ExpressJS, бібліотеки REACTJS та середовища виконання NodeJS.

ВСТУП

Сучасний етап розвитку інформаційно-комунікаційних технологій характеризується глобальним поширенням соціальних мереж та їх глибокою інтеграцією в повсякденну діяльність суспільства. Зростання тривалості перебування користувачів у віртуальному просторі часто корелює зі зниженням їхньої фізичної активності. Водночас комунікаційний потенціал соціальних платформ може бути ефективно використаний для популяризації занять спортом, координації фізичної активності та обміну тематичним досвідом.

Отже, темою роботи є розробка веб-додатку для соціальної мережі з модулем кластеризації стрічки новин.

Малорухливий характер взаємодії з веб-ресурсами негативно позначається на фізичному стані користувачів, викликаючи зниження працездатності та підвищену втомлюваність. Проте впровадження спеціалізованих інструментів у соціальні мережі дозволяє трансформувати цей вплив у позитивний. Можливість обміну даними, пошуку релевантної інформації та моніторингу новин спорту створює сприятливе середовище для залучення користувачів до занять спортом.

Таким чином, розробка спеціалізованої соціальної мережі, орієнтованої на підтримку занять спортом, є актуальним науково-практичним завданням.

Метою дипломної роботи є розробка програмного забезпечення соціальної мережі з модулем кластеризації стрічки новин для координації та популяризації занять спортом.

Для досягнення поставленої мети сформульовано такі завдання: – провести системний аналіз предметної області; – дослідити існуючі програмні рішення та здійснити постановку задачі; – визначити функціональні вимоги до проектованої системи; – розробити логічну та фізичну архітектуру веб-додатку; – програмно реалізувати систему; – здійснити комплексне тестування розробленого програмного продукту.

Об’єктом дослідження є процес функціонування соціальної мережі, спрямованої на підтримку занять спортом. .

Предметом дослідження є інструментальні засоби та алгоритми розробки веб-додатку соціальної мережі на базі стеку технологій MERN із застосуванням методу кластеризації k-means.

Методи дослідження. У процесі виконання роботи застосовувалися методи системного аналізу для проектування архітектури, принципи об'єктно-орієнтованого програмування, а також методи машинного навчання, зокрема алгоритм k-means для інтелектуального групування стрічки новин.

Практичне значення одержаних результатів полягає у створенні готового до експлуатації веб-додатку, який може бути впроваджений та використаний цільовою аудиторією для обміну досвідом щодо занять спортом і формування відповідних спільнот.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ СОЦІАЛЬНОЇ МЕРЕЖІ

1.1. Аналіз предметної області та обґрунтування створення програмного продукту

Предметною областю цього кваліфікаційного дослідження є соціальні мережі, орієнтовані на популяризацію занять спортом. Сучасний етап еволюції інформаційно-комунікаційних технологій характеризується стрімким розширенням користувацької аудиторії соціальних платформ. На ринку програмного забезпечення представлено широкий спектр комунікаційних систем різної тематичної спрямованості. Застосування новітніх алгоритмів та архітектурних підходів дозволяє створювати програмні рішення з персоналізованою видачею контенту. Формування інформаційної стрічки на основі предиктивного аналізу користувацьких запитів та поведінкових метрик суттєво підвищує ефективність взаємодії з платформою.

Одночасно з технологічним прогресом спостерігається стійка тенденція до підвищення суспільного інтересу щодо занять спортом. Як наслідок, виникає об'єктивна потреба в акумуляції та обміні релевантним досвідом, новинами та спеціалізованими знаннями. Для забезпечення такої комунікації є необхідність розробки новітнього програмного продукту, здатного консолідувати цільову аудиторію зі спільними інтересами у сфері занять спортом.

З огляду на це, основним завданням роботи є проектування та програмна реалізація спеціалізованої соціальної мережі, функціонал якої забезпечить користувачам можливість обміну досвідом та тематичною інформацією щодо занять спортом. Це сприятиме централізації релевантних даних у єдиному інформаційному середовищі та розширенню доступу до спеціалізованого контенту для підтримання високого рівня фізичної активності.

1.2. Порівняльний аналіз аналогів соціальної мережі

На сучасному ринку інформаційних послуг функціонує значна кількість соціальних платформ, які забезпечують інформаційний обмін за різними напрямками. Одним із провідних представників цього сегменту є соціальна мережа Instagram, створена у 2010 році Кевіном Сістромом та Майком Крігером, і згодом інтегрована в корпоративну структуру компанії Facebook Inc [2]. Зазначена система володіє розширеним інструментарієм для публікації мультимедійного контенту, обміну інформацією та комунікації між користувачами. Графічний інтерфейс користувача відзначається високим рівнем ергономічності, а архітектура програмного рішення передбачає наявність клієнтських додатків, адаптованих як для десктопних систем, що проілюстровано на рисунку 1.1, так і для мобільних пристроїв, що подано на рисунку 1.2.

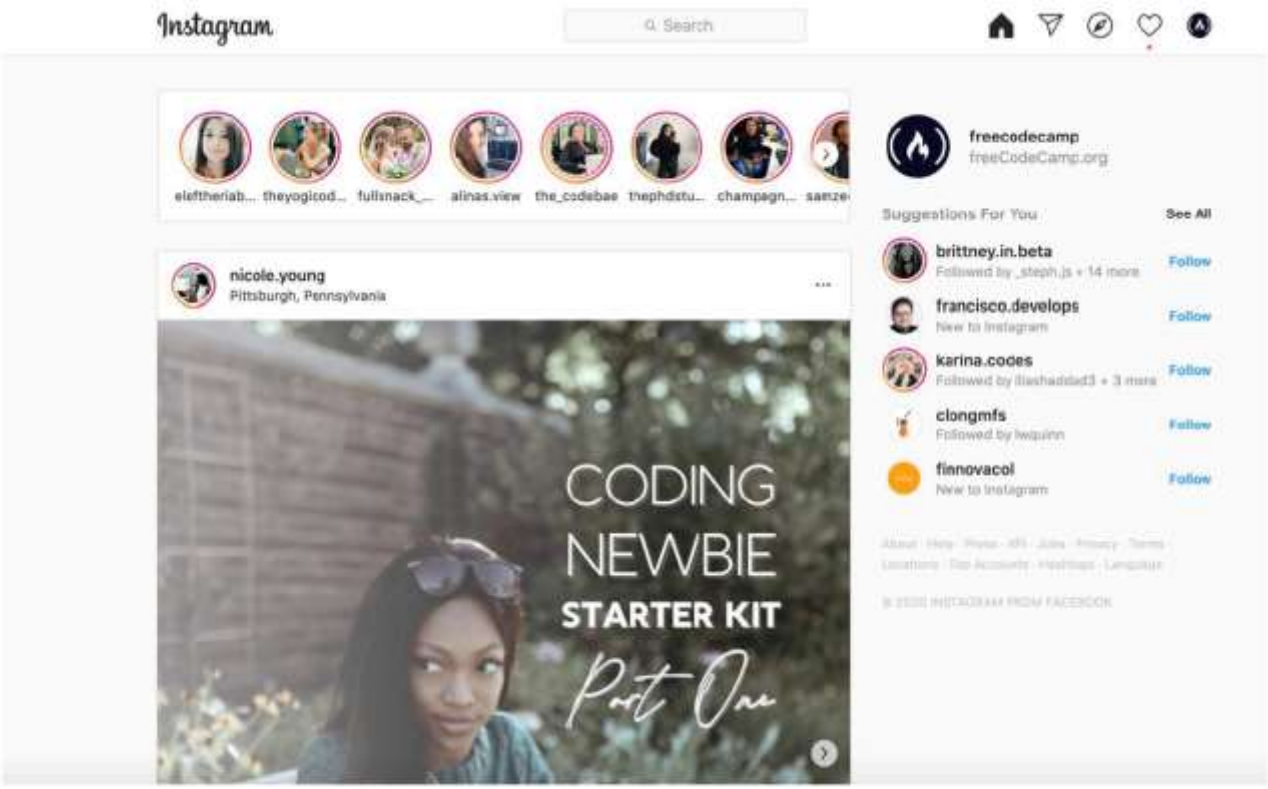


Рисунок 1.1 – Графічний інтерфейс соціальної мережі Instagram: фрагмент головної сторінки для десктопних систем



Рисунок 1.2 – Графічний інтерфейс соціальної мережі Instagram: фрагмент головної сторінки для мобільних пристроїв

Функціональні можливості платформи Instagram забезпечують універсальний обмін даними будь-якої тематики, що зумовлює її високу адаптивність та придатність для використання у широкому спектрі сфер діяльності.

Під час аналізу предметно-орієнтованих веб-ресурсів, які спеціалізуються на поширенні інформації щодо занять спортом та спортивних заходів, було досліджено інформаційний портал SPORT.ua. Даний ресурс забезпечує агрегацію та відображення актуальних новин і результатів спортивних подій, що подано на рисунку 1.3.

Проте суттєвим обмеженням зазначеного веб-додатку є відсутність інтерактивного комунікаційного модуля. Архітектура системи не надає користувачам інструментарію для генерування авторського контенту, публікації коментарів та здійснення безпосередньої мережевої взаємодії, що є базовими критеріями функціонування повноцінних соціальних мереж.

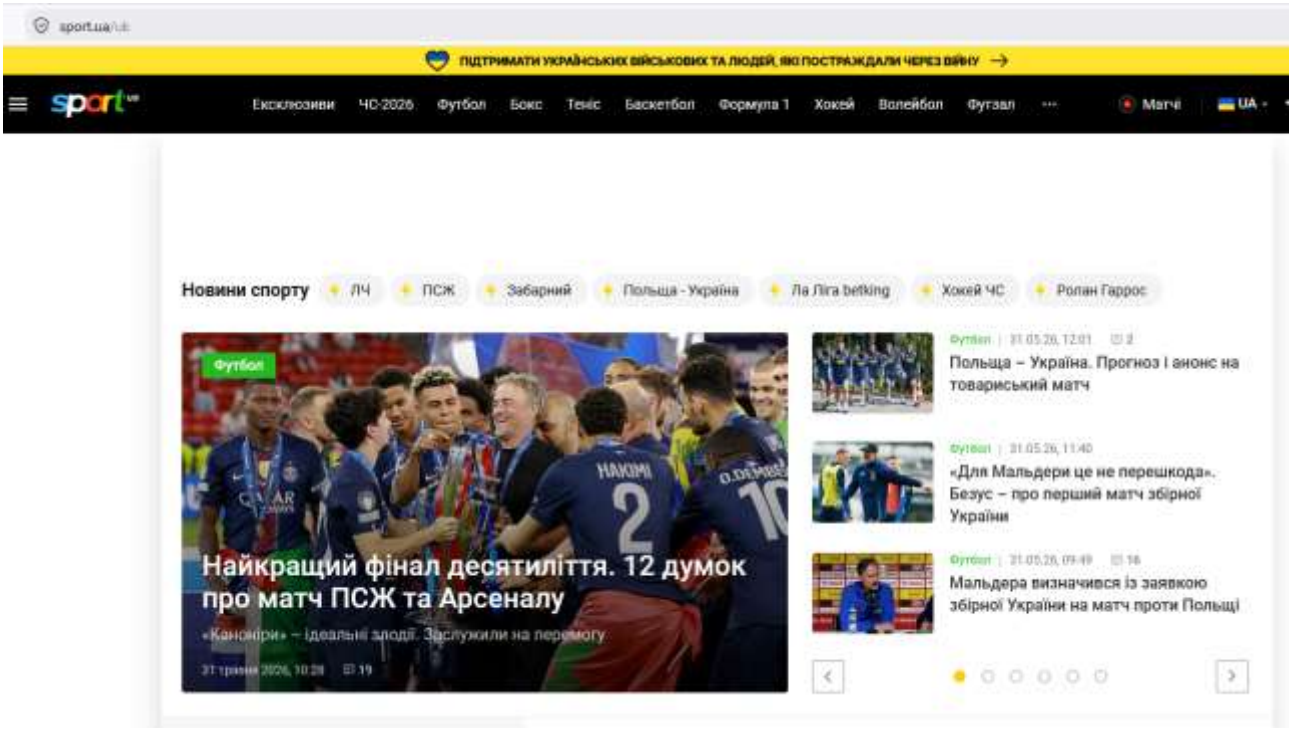


Рисунок 1.3 – Графічний інтерфейс інформаційного порталу SPORT.ua: фрагмент головної сторінки

Альтернативним об'єктом для порівняльного аналізу виступає глобальна соціальна мережа Facebook, розроблена під керівництвом Марка Цукерберга та інтегрована в інфраструктуру корпорації Meta Platforms [3]. Візуальне подання інтерфейсу користувача наведено на рисунку 1.4. Архітектура даної інформаційної системи забезпечує користувачам розширені можливості для генерування та поширення різнопланових даних шляхом створення публікацій та використання інтегрованого комунікаційного інструментарію. Проте, аналогічно до платформи Instagram, контентне наповнення мережі Facebook характеризується політематичністю. Відповідно, інформаційні потоки у системі охоплюють широкий спектр загальних напрямків і не фокусуються виключно на питаннях занять спортом, що ускладнює цілеспрямований пошук та консолідацію відповідної цільової аудиторії.

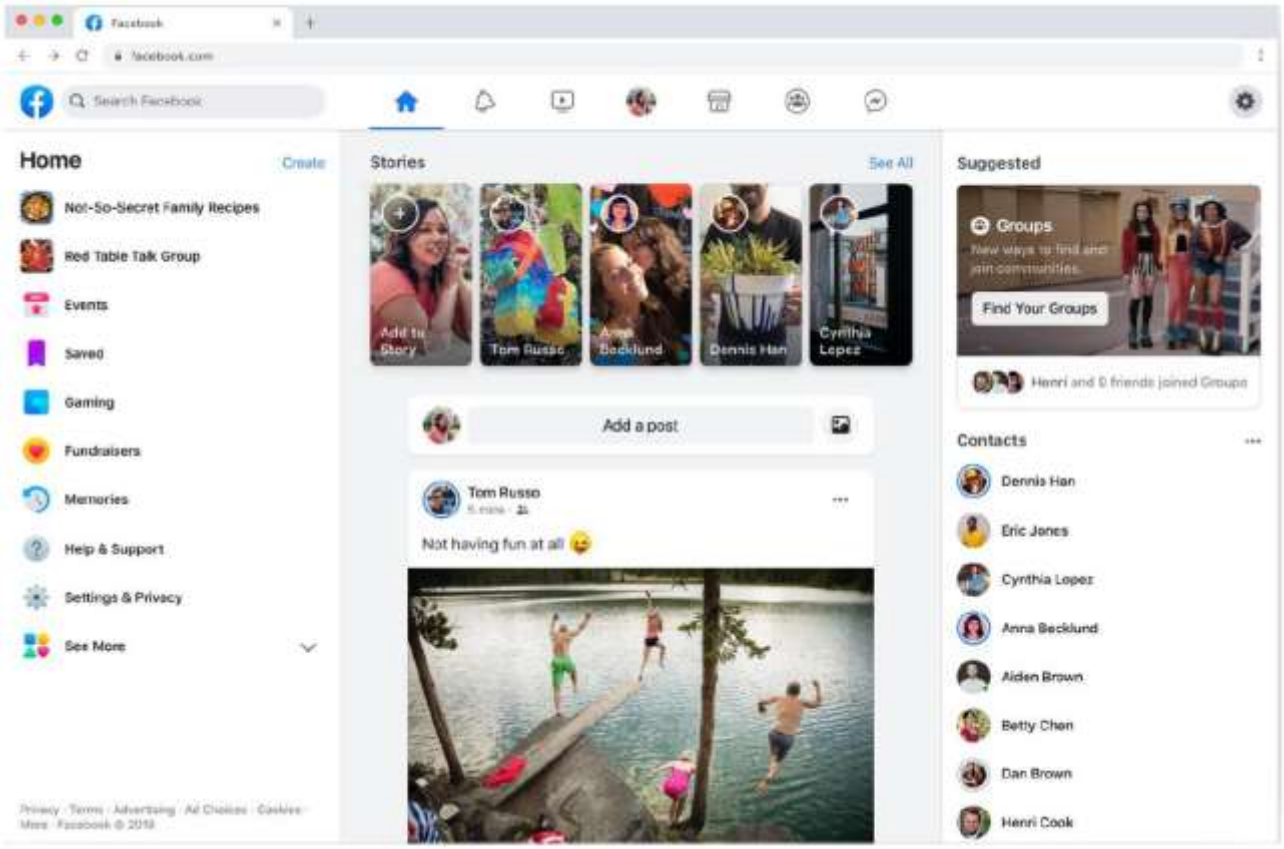


Рисунок 1.4 – Графічний інтерфейс соціальної мережі Facebook: фрагмент головної сторінки

Досліджені програмні аналоги комунікаційних платформ розповсюджуються із закритим вихідним кодом. Архітектура та функціональне наповнення таких систем здебільшого орієнтовані на трансляцію та обмін даними без встановлення жорстких інформаційних обмежень. Разом з тим, розглянуті програмні рішення характеризуються високим рівнем кросплатформності, що забезпечує їх стабільне та повноцінне функціонування під управлінням широкого спектру операційних систем.

1.3. Постановка задачі

На основі проведеного аналізу існуючих аналогів обґрунтовано вибір об’єкта та предмета дослідження. У цій роботі об’єктом дослідження є процес функціонування соціальної мережі, орієнтованої на заняття спортом.

Предметом дослідження є програмна реалізація веб-додатку соціальної мережі на базі стеку технологій MERN із застосуванням методу кластеризації k-means.

Метою дипломної роботи є розробка веб-додатку для соціальної мережі з модулем кластеризації стрічки новин. Для досягнення поставленої мети необхідно вирішити такі завдання:

– виконати системний аналіз предметної області, а саме особливостей функціонування соціальних мереж для занять спортом;

– провести порівняльний аналіз існуючих аналогів комунікаційних платформ, зокрема Instagram, Sport.ua та Facebook;

– визначити та проаналізувати функціональні й нефункціональні вимоги до проектованої системи;

– дослідити та обґрунтувати вибір інструментальних засобів і технологій розробки, визначивши їхні переваги;

– спроектувати графічний інтерфейс користувача, логічну структуру бази даних та функціональні моделі розв’язання поставленої задачі;

– програмно реалізувати веб-додаток соціальної мережі з модулем кластеризації стрічки новин.

У результаті аналізу таких програмних продуктів, як Instagram, Sport.ua та Facebook, встановлено, що зазначені аналоги забезпечують інформаційний обмін у межах широкого спектру тематик. Це не дозволяє позиціонувати їх як вузькоспеціалізовані платформи, спрямовані виключно на заняття спортом. Переважна більшість розглянутих систем орієнтована на трансляцію політематичного контенту, а архітектура цих додатків забезпечує кросплатформність та функціонування у середовищі різних операційних систем.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З МОДУЛЕМ КЛАСТЕРИЗАЦІЇ СТРІЧКИ НОВИН

2.1. Визначення функціональних вимог соціальної мережі

Проектована інформаційна система призначена для вузькоспеціалізованого обміну даними, обмеженого тематикою занять спортом. Впровадження такої платформи є об'єктивною необхідністю, оскільки акумуляція цільової аудиторії на базі єдиного веб-додатку оптимізує процеси пошуку, відображення та обміну релевантним контентом. Відповідна консолідація інформаційних ресурсів дозволяє користувачам ефективно розширювати базу знань у сфері занять спортом. Для забезпечення контролю якості та відповідності інформаційних потоків архітектура соціальної мережі базується на рольовій моделі доступу, що вимагає обов'язкової наявності адміністративного модуля для модерації тематики публікацій.

З огляду на те, що головною метою роботи є розробка веб-додатку для соціальної мережі з модулем кластеризації стрічки новин, визначено такий набір базових функціональних вимог до програмного продукту.

У межах ролі користувача системи передбачено реалізацію таких можливостей: – проходження процедур реєстрації та авторизації; – модифікація персональних даних облікового запису; – управління автентифікаційними даними; – генерування, редагування та видалення інформаційних публікацій; – створення та видалення коментарів до публікацій; – інтерактивна взаємодія з контентом через систему оцінювання; – фільтрація інформаційного потоку за визначеними категоріями; – встановлення соціальних зв'язків за допомогою механізму підписок; – відображення переліку підписників та підписок у межах персонального профілю; – збереження відібраних матеріалів у спеціалізованому

розділі; – виконання пошукових запитів для знаходження користувачів та релевантного контенту; – перегляд персоналізованої стрічки новин, сформованої на базі результатів роботи модуля кластеризації, який аналізує та групує контент відповідно до інтересів суб'єкта.

У межах ролі адміністратора передбачено реалізацію таких можливостей: – створення, модифікація та видалення глобальних категорій інформаційних матеріалів.

З метою забезпечення ергономічності та оптимізації процесу подальшої програмної реалізації графічного інтерфейсу користувача було спроектовано візуальну архітектуру веб-додатку. Як основний інструментальний засіб для виконання цього завдання обрано хмарне середовище макетування Figma.

Зазначений програмний продукт є спеціалізованим веб-додатком, призначеним для проектування користувацьких інтерфейсів та обробки графічних даних. Функціонал платформи дозволяє вирішувати широкий спектр завдань у галузі проектування, включно з макетуванням веб-ресурсів, розробкою інтерфейсів для різних інформаційних систем та створенням інтерактивних прототипів. Ключовою архітектурною перевагою даного інструментарію порівняно з аналогами є його функціонування безпосередньо у середовищі веб-браузера. Така технологічна особливість забезпечує кросплатформний доступ до проектних матеріалів з будь-якої обчислювальної системи без необхідності інсталяції локального програмного забезпечення або придбання додаткових ліцензій. Крім того, платформа надає розширений набір інструментів вільного доступу, що повною мірою задовольняє вимоги до комплексного проектування візуальної складової системи [12].

У межах проектування розроблюваної соціальної мережі було створено деталізовані дизайн-макети для всіх структурних елементів додатку,

включаючи основні веб-сторінки, системи модальних вікон та форми обробки даних (рис. 2.1.-2.3).

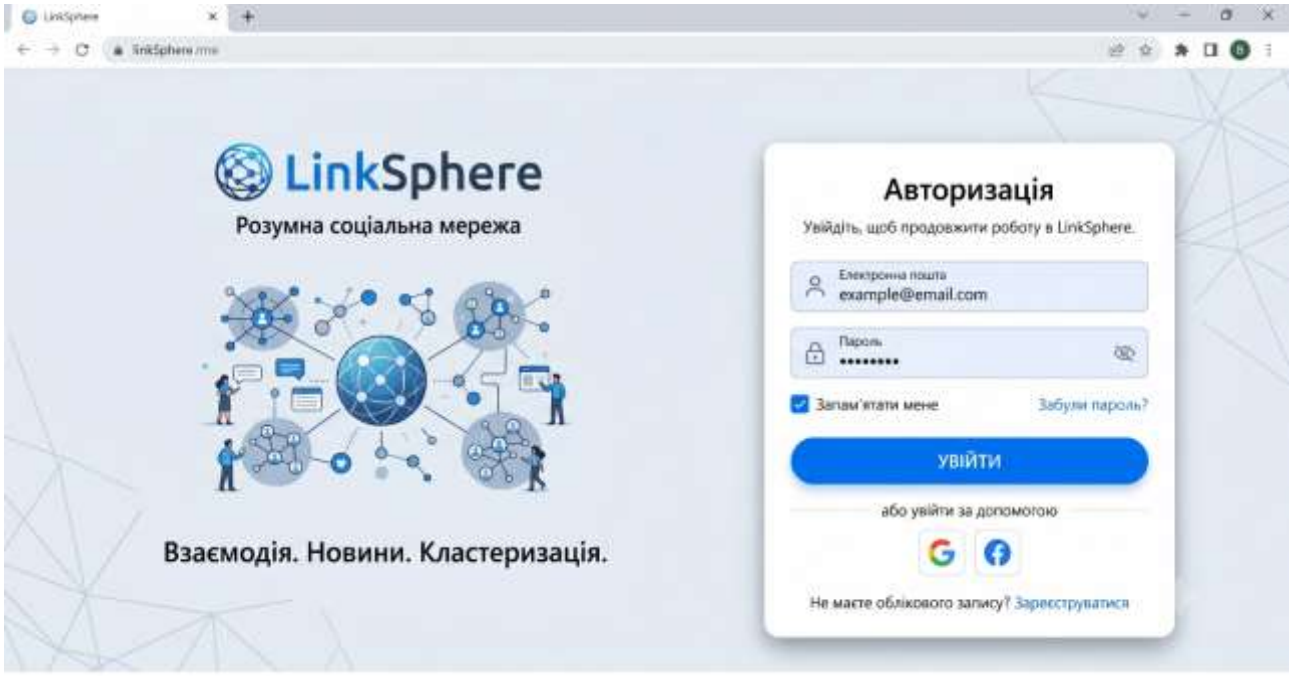


Рисунок 2.1 – Проектний макет графічного інтерфейсу підсистеми авторизації



Рисунок 2.2 – Проектний макет візуального відображення персонального профілю користувача

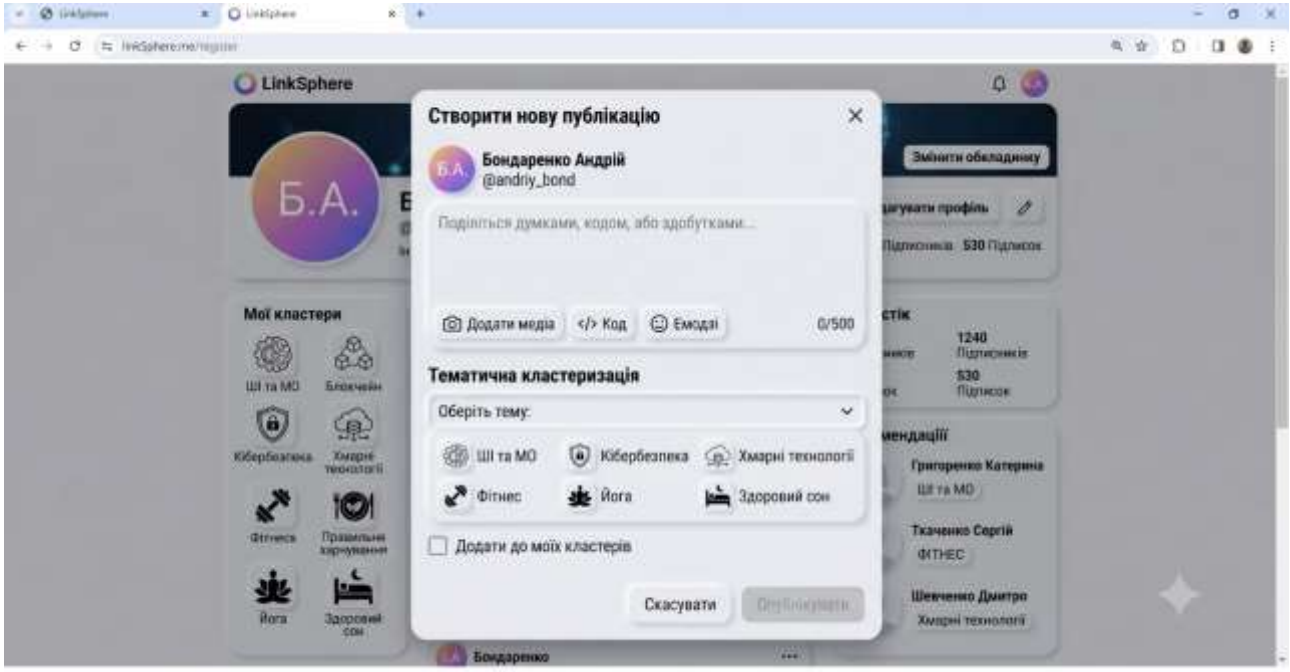


Рисунок 2.3 – Проектний макет графічного інтерфейсу діалогового вікна створення публікації

2.3. Проєктування бази даних

З метою проектування бази даних веб-додатку соціальної мережі було здійснено детальний аналіз та визначено повний перелік інформаційних сутностей предметної області, на основі чого розроблено діаграму сутність-зв'язок, яка наведена на рисунку 2.4.

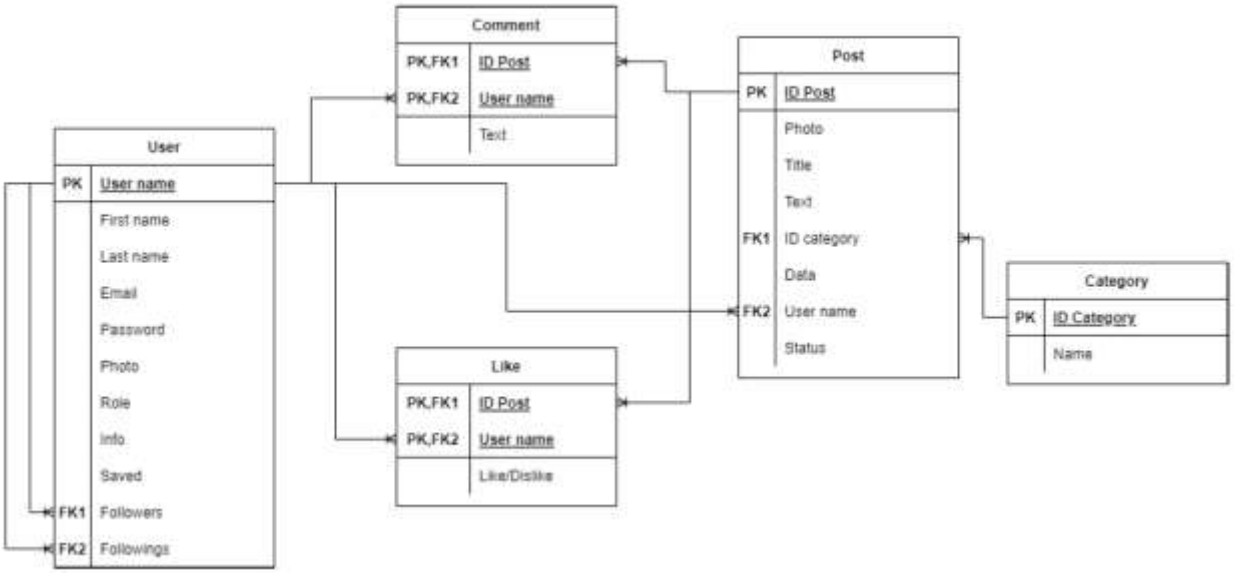


Рисунок 2.4 – Схема «сутність-зв'язок» бази даних

На розробленій діаграмі відображено п’ять базових інформаційних сутностей: Користувач, Публікація, Коментар, Вподобання та Категорія. Для кожної з наведених сутностей визначено відповідний набір атрибутів, первинних і зовнішніх ключів, а також спроектовано логічні зв’язки між відповідними реляційними структурами.

Сутність Користувач акумулює масив даних, необхідних для ініціалізації та ідентифікації облікового запису суб'єкта системи, зокрема псевдонім, ім’я, прізвище, адресу електронної пошти та автентифікаційний пароль. Ключовими структурними елементами цієї сутності виступають списки підписок та підписників, функціональне призначення яких полягає у забезпеченні механізмів соціальної взаємодії та формуванні комунікаційної мережі між учасниками платформи.

Сутність Публікація характеризується набором специфічних атрибутів, серед яких мультимедійні матеріали, заголовок, текстове наповнення та часова мітка створення. Дана сутність в обов’язковому порядку фіксує ідентифікатор категорії, до якої відноситься контент. Між сутностями Публікація та Категорія встановлено тип відношення один до багатьох, оскільки окремий інформаційний матеріал класифікується виключно в межах однієї визначеної категорії, тоді як єдина категорія здатна містити множину різних публікацій.

Сутності Коментар та Вподобання виступають сполучними елементами, що забезпечують двосторонню інтерактивну взаємодію між об'єктами Користувач та Публікація. Їхня логічна структура включає відповідні атрибути для збереження текстового вмісту повідомлення та фіксації логічного стану оцінювання інформаційного матеріалу.

2.3. Обґрунтування вибору стеку технологій для реалізації поставлених завдань

Для вирішення поставлених завдань було обрано комплексний технологічний стек MERN. Зазначений інструментарій являє собою екосистему на базі мови програмування JavaScript, орієнтовану на оптимізацію та прискорення процесу розгортання повностекових веб-додатків. Базова структура стеку MERN інтегрує чотири ключові технології: систему управління базами даних MongoDB, каркас веб-додатків Express, бібліотеку побудови користувацьких інтерфейсів ReactJs та програмну платформу Node.js.

Впровадження даного програмного рішення спрямоване на підвищення ефективності життєвого циклу розробки. Кожен із перелічених компонентів формує єдину наскрізну інфраструктуру, виконуючи критично важливу функцію у процесі проектування та забезпечення стабільного функціонування створюваного програмного продукту [4].

Застосування архітектурної моделі MERN створює оптимальні умови для реалізації класичної трирівневої структури інформаційної системи, де процеси обробки та передачі інформації здійснюються виключно засобами мови JavaScript та формату обміну даними JSON, що наочно проілюстровано на рисунку 2.5.

Клієнтський рівень архітектури MERN представлений React.js – декларативною бібліотекою мови JavaScript, призначеною для розробки динамічних програмних додатків. Дана технологія забезпечує конструювання комплексних користувацьких інтерфейсів на базі простіших інкапсульованих компонентів, їх інтеграцію із серверними даними та подальший рендеринг у форматі гіпертекстової розмітки.

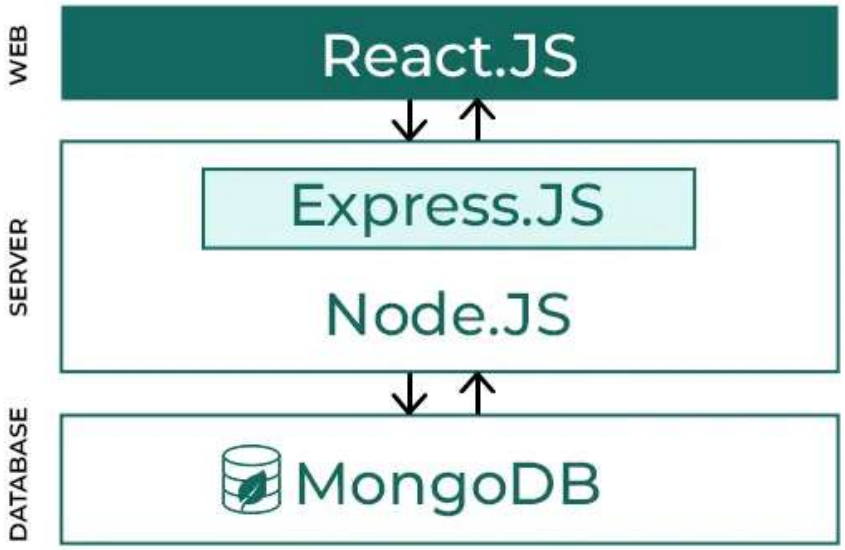


Рисунок 2.5 – Трирівнева архітектура технологічного стеку MERN

Ключова перевага інструментарію React полягає у високій ефективності функціонування інтерфейсів, орієнтованих на управління станами, за умови мінімізації обсягу вихідного коду та потенційних програмних дефектів [5]. Технологія відзначається високою швидкодією, масштабованістю та оптимізована для проектування високонавантажених веб-систем, які потребують асинхронного оновлення даних без ініціалізації повного перезавантаження сторінки [6].

Проміжним рівнем виступає програмний каркас Express.js, який функціонує у середовищі виконання Node.js. Зазначений фреймворк оптимізує процес розробки серверної логіки веб-додатків та прикладних програмних інтерфейсів, підтримуючи інтеграцію різноманітного проміжного програмного забезпечення. Це сприяє скороченню обсягу коду та підвищенню його читабельності. Express.js забезпечує високий рівень повторного використання програмного коду завдяки вбудованим механізмам маршрутизації, володіючи розширеними моделями для обробки уніфікованих локаторів ресурсів та маршрутизації мережевих запитів і відповідей протоколу HTTP [5].

Домінантними характеристиками Express.js є його висока обчислювальна ефективність, швидкодія, масштабованість, а також підтримка асинхронної однопотокової архітектури. Крім того, даний фреймворк підтримується найчисельнішою спільнотою розробників у глобальній екосистемі Node.js [6].

Базовим рівнем архітектури виступає нереляційна система управління базами даних MongoDB. Для забезпечення функцій збереження різноманітних інформаційних масивів, зокрема профілів користувачів, контенту, коментарів, мультимедійних файлів чи записів подій, доцільно застосовувати систему, інтеграція з якою відповідає принципам роботи обчислювальних рівнів React, Express та Node. Документо-орієнтована архітектура MongoDB дозволяє безпосередньо передавати об'єкти даних у форматі JSON від клієнтського середовища до серверного ядра для їх подальшої обробки, збереження та індексування у базі даних [5].

Порівняно з класичними реляційними базами даних, MongoDB характеризується низкою концептуальних переваг, серед яких наявність повноцінної хмарної платформи обробки даних, застосування гнучких схем документообігу, нативна інтеграція з програмним кодом, адаптивність архітектури до структурних змін, економічна доцільність впровадження, наявність вичерпної технічної документації, а також розширений інструментарій для формування складних запитів та проведення аналітики [7].

На рисунку 2.6 наведено структурне порівняння архітектури класичної реляційної бази даних та документо-орієнтованої системи MongoDB.

Tabular (Relational)	MongoDB
Database	Database
Table	Collection
Row	Document
Index	Index
Join	$lookup
Foreign Key	Reference

Рисунок 2.6 – Порівняльна характеристика реляційної БД та системи MongoDB

Для розширення функціональних можливостей розроблюваної системи додатково застосовано низку допоміжних технологічних рішень, зокрема Mongoose, Redux та Material UI.

Технологія Mongoose є спеціалізованою бібліотекою об'єктно-документного моделювання, орієнтованою на взаємодію з системою MongoDB. Вона забезпечує механізм проектування об'єктів класів на документи колекцій бази даних. Використання Mongoose дозволяє реалізувати ефективне рішення для моделювання структури даних на базі схем. Вбудований інструментарій бібліотеки підтримує автоматичне приведення типів, валідацію даних, конструювання складних запитів та інші допоміжні функції [8].

Бібліотека Redux застосовується для розробки програмних рішень, що характеризуються послідовністю виконання, здатністю функціонувати у гетерогенних середовищах та високим рівнем контролепридатності. Концептуальна централізація стану та бізнес-логіки додатку забезпечує реалізацію розширених функцій, серед яких механізми відкату та повторення операцій, а також персистентне збереження поточного стану системи.

До ключових переваг Redux належать забезпечення детермінованості стану додатку, строга регламентація архітектурної організації програмного коду, що підвищує його читабельність та сприяє глибшому розумінню структури будь-якої імплементації, а також значне спрощення процесу налагодження. Бібліотека містить інтегровані механізми оптимізації продуктивності, завдяки яким повторний рендеринг пов'язаних компонентів ініціюється виключно за умови фактичної зміни даних. Крім того, інструментарій підтримує функцію кешування фрагментів стану в локальному сховищі з можливістю їх подальшого відновлення та повністю адаптований для виконання рендерингу на стороні сервера [9].

Material UI виступає бібліотекою візуальних компонентів для екосистеми React із відкритим вихідним кодом, що імплементує концепцію дизайну Material Design. Програмний продукт надає вичерпну колекцію стандартизованих елементів інтерфейсу, повністю підготовлених до інтеграції у промислові рішення. Завдяки наявності розширеної системи конфігурації розробник отримує можливість гнучкого налаштування візуального подання та імплементації кастомних архітектурних стилів на базі існуючих компонентів.

Впровадження Material UI забезпечує суттєве скорочення часових витрат на проектування графічного інтерфейсу користувача. Кожен компонент бібліотеки відповідає сучасним стандартам ергономіки та функціональності. Тривалий період існування платформи на ринку гарантує її високу надійність та стабільну підтримку з боку професійної спільноти розробників, що є критично важливим фактором для довгострокового супроводу програмного продукту [10].

В якості базового середовища розробки програмного забезпечення було обрано інтегрований редактор вихідного коду Visual Studio Code. Дане кросплатформне рішення підтримує роботу під управлінням операційних систем Windows, macOS та Linux. Середовище володіє нативною підтримкою технологій JavaScript, TypeScript та Node.js, а також характеризується наявністю широкого спектру модулів розширення для інтеграції з іншими мовами програмування, серед яких C++, C#, Java, Python, PHP, Go, та платформами виконання, такими як .NET та Unity. Технічний інструментарій редактора забезпечує оптимальні умови для ефективного та безперешкодного конструювання програмного продукту [11].

2.4. Обґрунтування вибору методу кластеризації стрічки новин для користувачів

Для реалізації інтелектуального модуля формування стрічки новин соціальної мережі необхідно обрати оптимальний алгоритм машинного навчання. Завдання полягає у групуванні публікацій щодо занять спортом на основі їхнього семантичного або векторного представлення. Як потенційні рішення було розглянуто три алгоритми: метод k-means, метод просторової кластеризації на основі щільності DBSCAN та алгоритм агломеративної ієрархічної кластеризації.

Метод k-means здійснює розбиття множини об'єктів $X = \{x_1, x_2, \dots, x_n\}$ на задану кількість кластерів k, мінімізуючи сумарне квадратичне відхилення точок кластерів від їхніх центроїдів. Математична модель цільової функції, що підлягає мінімізації, має вигляд:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \left|\left| x_i - \mu_j \right|\right|^2 \quad (2.1)$$

де:

k – кількість кластерів,

C_j – множина елементів j-го кластера,

μ_j – вектор центроїда j-го кластера,

x_i – векторне представлення публікації,

$\left|\left| x_i - \mu_j \right|\right|^2$ – квадрат евклідової відстані між об'єктом та центроїдом.

Центроїд обчислюється як середнє арифметичне всіх векторів відповідного кластера:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \quad (2.2)$$

Агломеративна ієрархічна кластеризація використовує підхід послідовного об'єднання найближчих кластерів, починаючи з ініціалізації кожного об'єкта як окремого кластера. Для обчислення відстані між кластерами U та V найчастіше

застосовується критерій Уорда, який мінімізує приріст внутрішньокластерної дисперсії при об'єднанні:

$$D(U,V) = \frac{|U|\cdot|V|}{|U|+|V|}\left|\left|\mu_U - \mu_V\right|\right|^2 \quad (2.3)$$

де:

$|U|$ та $|V|$ – потужності кластерів,

μ_U та μ_V – їхні геометричні центри.

Алгоритм DBSCAN групує точки, які знаходяться щільно одна до одної, позначаючи елементи в областях з низькою щільністю як інформаційний шум. Алгоритм базується на двох параметрах: радіусі околу ϵ та мінімальній кількості точок $MinPts$.

Точка p вважається ядерною, якщо виконується умова:

$$|N_\epsilon(p)| \geq MinPts \quad (2.4)$$

де:

$N_\epsilon(p)$ описує множину точок у заданому радіусі:

$$N_\epsilon(p) = \{q \in X \mid d(p,q) \leq \epsilon\} \quad (2.5)$$

Критеріями вибору оптимального методу для веб-додатку соціальної мережі виступають обчислювальна складність, здатність обробляти великі обсяги даних у реальному часі та передбачуваність результатів класифікації спортивного контенту.

Часова складність методу k-means становить $O(n \cdot k \cdot i)$, де i – кількість ітерацій до збіжності. Даний показник забезпечує лінійну залежність від кількості публікацій. Часова складність агломеративної ієрархічної кластеризації досягає $O(n^3)$, що робить її ресурсомісткою та непридатною для динамічного перерахунку стрічки новин при постійному генеруванні нового контенту користувачами.

Метод DBSCAN має обчислювальну складність $O(n \log n)$ за умови використання просторових індексів, проте його ефективність суттєво знижується у багатовимірних просторах, які є характерними для векторних представлень текстових публікацій. Крім того, цей алгоритм має складнощі з кластеризацією наборів даних зі змінною щільністю, що може призвести до втрати релевантних новин спорту, які алгоритм класифікує як шум.

Метод k-means забезпечує високу швидкість збіжності та масштабованість, що є критично важливим для серверної частини розроблюваного додатку на базі технологічного стеку MERN. Гнучкість налаштування параметра k дозволяє алгоритмічно адаптувати структуру стрічки новин під різнопланові категорії спортивної діяльності. Наведена математична модель та результати порівняльної оцінки обчислювальної складності підтверджують беззаперечну доцільність використання методу k-means як базового інструменту кластеризації у проектованій системі.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ СОЦІАЛЬНОЇ МЕРЕЖІ З МОДУЛЕМ КЛАСТЕРИЗАЦІЇ СТРІЧКИ НОВИН

3.1. Реалізація обробки та аналізу даних

Базовим процесом обробки інформації у проектованій системі є взаємодія клієнтського графічного інтерфейсу із системою управління базами даних. У межах зазначеної взаємодії реалізується обробка та маршрутизація комплексу мережевих запитів. Як практичний приклад програмної логіки доцільно розглянути процедуру отримання даних конкретної публікації. Ініціалізація доступу до матеріалу призводить до генерації модального вікна, що містить деталізовані відомості про відповідний інформаційний об'єкт.

На рівні користувацького інтерфейсу генерується цільова подія, якою у даному сценарії виступає інтерактивна взаємодія з візуальним елементом публікації. Зазначена подія ініціює виконання відповідної функції-обробника, програмна логіка якої інкапсульована в архітектурному компоненті публікації, що проілюстровано на рисунку 3.1.

Програмна реалізація вихідного коду зазначеного компонента наведена у додатку А.

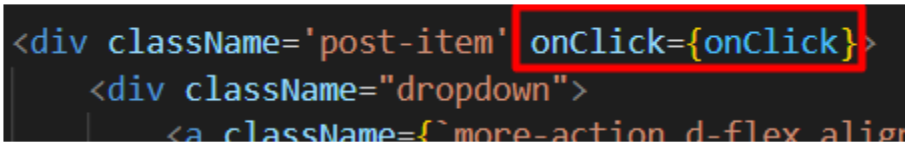


Рисунок 3.1 – Програмна реалізація обробника події onClick у компоненті публікації

Інтерактивна взаємодія з елементом публікації ініціює відображення модального вікна, в межах компонента якого виконується виклик функції GetPostById. Даний метод забезпечує отримання детальної інформації про публікацію на основі її унікального ідентифікатора та локалізований у програмному модулі post.action.js, що наведено на рисунку 3.2. Окрім зазначеного алгоритму, даний файл інкапсулює комплекс додаткових методів, призначених для обробки даних.

Вихідний код програмного модуля post.action.js подано у додатку Б.

```
export function GetPostById(postId) {
    return dispatch => {
        postService.getPostById(postId)
        .then ((res) => {
            if (res.status === 200) {
                dispatch(setPost(res.data));
            }
        });
    }
}
```

Рисунок 3.2 – Програмна реалізація функції GetPostById у модулі post.action.js

У подальшому процесі виконання алгоритму обробки даних викликається сервісний модуль post.service.js, який містить однойменний метод GetPostById, що відображено на рисунку 3.3. Відповідний метод забезпечує формування та відправлення цільового мережевого запиту до серверного ядра розроблюваної системи.

Вихідний код сервісного модуля post.service.js наведено у додатку В.

```
getPostById = async (postId) => {
    try {
        return await axios({
            method: "get",
            url: `http://localhost:5000/posts/${postId}`,
            headers: {
                Authorization: `Bearer ${localStorage.getItem("Token")}`
            }
        });
    }
    catch(error) {
        return error.response;
    }
}
```

Рисунок 3.3 – Програмна реалізація функції GetPostById у сервісному модулі post.service.js

На серверному рівні розроблюваної системи реалізовано модуль маршрутизації post.js, функціональне призначення якого полягає в переспрямуванні мережевих запитів безпосередньо до системи управління

базами даних. Програмний алгоритм здійснює пошук цільової публікації за її унікальним ідентифікатором, екстрагує відповідний масив інформації з бази даних та формує об'єкт відповіді, що проілюстровано на рисунку 3.4. Згенерований результат ретранслюється на клієнтську частину архітектури до ініціюючого сервісного методу.

Сервісний метод, після отримання даних від сервера, передає їх до функції управління станом, що локалізована у модулі post.action.js. Зазначена функція обробляє вхідний масив інформації та здійснює відповідну модифікацію глобального стану програмного продукту.

Лістинг вихідного коду модуля маршрутизації post.js подано у додатку Г.

Підсумовуючи описаний цикл обробки даних, слід зазначити, що інтерактивний вибір публікації ініціює вилучення актуальної інформації з поточного стану системи з її подальшим відображенням у компоненті модального вікна графічного інтерфейсу користувача.

```
router.route('/:id').get((req, res) => {
    Post.findById(req.params.id)
    .populate({
        path: "likes",
        populate : {
            path : 'user'
        }
    })
    .populate({
        path: "comments",
        populate : {
            path : 'user'
        }
    })
    .populate({
        path: "category"
    })
    .populate({
        path: "user"
    })
    .then(post => res.json(post))
    .catch(err => res.status(400).json('Error: ' + err));
});
```

Рисунок 3.4 – Програмна реалізація обробника маршруту в серверному модулі post.js

3.2. Програмна реалізація підсистеми управління базами даних

Проектування інформаційної інфраструктури здійснено на базі документо-орієнтованої системи управління базами даних з відкритим вихідним кодом MongoDB. Збереження масивів даних на фізичних носіях реалізовано у бінарному форматі BSON. Зазначена архітектурна особливість забезпечує розширену підтримку специфічних типів даних, серед яких Decimal128 та ISODates, що суттєво перевищує базові можливості стандарту JSON. Документо-орієнтована модель підтримує ієрархічну структуру вкладених об'єктів та масивів, що дозволяє оптимізувати процеси збереження та екстракції складноструктурованої інформації.

З метою забезпечення високого рівня доступності та відмовостійкості було прийнято рішення відмовитись від локального розгортання сервера баз даних на користь спеціалізованої хмарної платформи MongoDB Atlas. Зазначений сервіс функціонує як повністю кероване хмарне рішення, що автоматизує процеси конфігурації, адміністрування та резервного копіювання інформаційних ресурсів. Використання цієї платформи є ефективним методом розгортання та горизонтального масштабування системи MongoDB у хмарному середовищі [14].

У процесі ініціалізації інфраструктури на платформі MongoDB Atlas було сформовано новий проект та сконфігуровано виділений обчислювальний кластер. Для гарантування інформаційної безпеки розроблено політику доступу із визначенням ролей користувачів та обмеженням авторизації за довіреними IP-адресами. За результатами налаштування згенеровано автентифікаційний рядок підключення, захищений криптографічним паролем.

Програмна реалізація серверної частини веб-додатку базується на ініціалізації головного виконуваного модуля server.js у кореневій директорії проекту. У межах даного файлу сконфігуровано екземпляр сервера Express, інтегровано проміжне програмне забезпечення CORS для кросдоменного обміну даними та модуль express.json для парсингу мережевих запитів. Прослуховування вхідних з'єднань налаштовано на мережевий порт 5000.

Додатково у зазначеному модулі реалізовано алгоритм встановлення з'єднання з хмарною базою даних з використанням системної змінної середовища ATLAS_URI. Конфіденційний рядок підключення інкапсульовано у конфігураційному файлі .env, що продемонстровано на рисунку 3.5.

Лістинг вихідного коду головного серверного модуля server.js наведено у додатку Ґ.

```
ATLAS_URI=mongodb+srv://root:root@cluster0.pvehy.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
```

Рисунок 3.5 – Конфігураційний рядок підключення до системи управління базами даних

На наступному етапі було здійснено проектування логічних схем бази даних із застосуванням бібліотеки об'єктно-документного моделювання Mongoose. Як репрезентативний приклад доцільно розглянути структурну схему інформаційної сутності Коментар. З цією метою у структурі проекту було сформовано директорію models, у якій ініціалізовано програмний модуль comment.model.js, що проілюстровано на рисунку 3.6.

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

const Schema = mongoose.Schema;

const commentSchema = new Schema({
  post: {
    type: Schema.Types.ObjectId,
    ref: "Post",
    required: true
  },
  user: {
    type: Schema.Types.ObjectId,
    ref: "User",
    required: true
  },
  text: {
    type: String,
    required: true
  },
}, {
  timestamps: true,
});

const Comment = mongoose.model('Comment', commentSchema);

module.exports = Comment;
```

Рисунок 3.6 – Програмна реалізація логічної схеми сутності у модулі comment.model.js

Аналіз наведеної схеми свідчить, що архітектура даної сутності включає такі базові атрибути: унікальний ідентифікатор публікації, ідентифікатор користувача та текстовий вміст повідомлення.

Результат ініціалізації та візуальне подання відповідної колекції даних у середовищі хмарної платформи MongoDB Atlas наведено на рисунку 3.7.

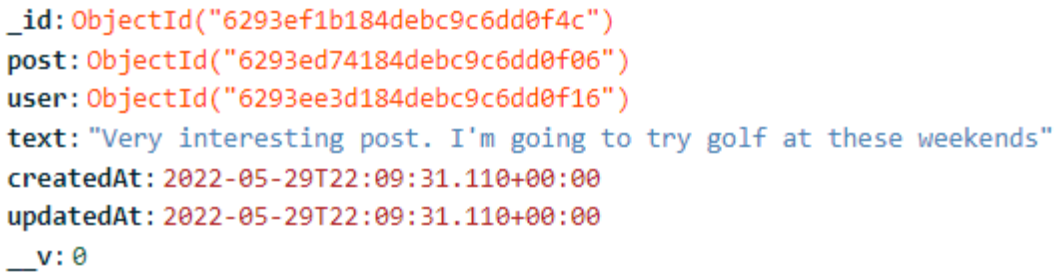


Рисунок 3.7 – Структурне подання колекції Коментар

Аналіз наведеного рисунка свідчить, що окрім базових атрибутів ідентифікації публікації, користувача та текстового наповнення, структура запису містить додаткові поля з часовою міткою та унікальним ідентифікатором безпосередньо об'єкта коментаря. Слід зауважити, що генерація унікального ключа та фіксація часу створення запису здійснюються механізмами системи управління базами даних MongoDB в автоматичному режимі. Хоча зазначений функціонал підлягає гнучкій конфігурації, його застосування за замовчуванням є технічно доцільним, оскільки це оптимізує архітектуру даних та суттєво спрощує процес програмної реалізації інформаційної системи.

3.3. Програмна реалізація графічного інтерфейсу користувача

Проектування та програмна реалізація користувацького інтерфейсу є критично важливим етапом життєвого циклу конструювання програмного продукту. Дана підсистема виступає базовим механізмом забезпечення інтерактивної взаємодії суб'єкта з апаратно-програмним комплексом інформаційної системи. З метою конструювання візуальної складової розроблюваного веб-додатку було застосовано комплекс технологічних рішень, зокрема ReactJS, Redux, MUI, синтаксис JSX та каскадні таблиці стилів CSS.

Архітектурна модель бібліотеки ReactJS базується на строгому компонентному підході. Кожен структурний елемент розробленого програмного рішення характеризується високим рівнем повторного використання та інкапсулює логіку генерації локального фрагмента гіпертекстової розмітки. Це забезпечує можливість багаторазової імплементації розроблених модулів у різних частинах веб-додатку шляхом їхнього декларативного інтегрування. Конструювання візуальних компонентів реалізовано із застосуванням формату JSX.

Специфікація JSX являє собою синтаксичне розширення базової мови JavaScript у межах екосистеми React, функціональне призначення якого полягає у структуруванні алгоритмів рендерингу компонентів за допомогою оптимізованих семантичних конструкцій [15]. За своїм візуальним поданням та структурною організацією зазначений формат є наближеним до класичної мови гіпертекстової розмітки HTML, що проілюстровано на рисунку 3.8.

```
import React, { Component } from 'react';
import '../styles/signforms.css';

import LoginForm from "./login.form.component"

export default class Login extends Component {

    render() {
        return (
            <div id='login'>
                <header>
                    <a href="/" className="d-flex text-decoration-none">
                        <img src={process.env.PUBLIC_URL + '/img/Logo.svg'} className="bi me-2" width="40" height="32"></img>
                        <span className="fs-4">HealthyLife</span>
                    </a>
                </header>
                <div className="login-main">
                    <LoginForm data={'hello'}/>
                </div>
                <footer>
                    <p className="rights">© 2022 Olena Kyrychenko 444. All rights reserved.</p>
                </footer>
            </div>
        );
    }
}
```

Рисунок 3.8 – Програмна реалізація компонента авторизації із застосуванням синтаксису JSX

Використання бібліотеки візуальних компонентів MUI суттєво оптимізує та прискорює процес проектування графічного інтерфейсу користувача.

Інтеграція даного інструментарію здійснюється шляхом інсталяції відповідного програмного пакета з його подальшим імпортом у цільовий модуль розробки, що проілюстровано на рисунку 3.9. Застосування цієї бібліотеки надає розширені можливості для ефективного конструювання різноманітних структурних елементів інтерфейсу. Репрезентативним прикладом такого архітектурного компонента є діалогове модальне вікно, візуальне подання якого наведено на рисунку 3.10.

```
import Box from '@mui/material/Box';
import Modal from '@mui/material/Modal';
```

Рисунок 3.9 – Процедура імпорту програмного пакета MUI

```
<Modal
    open={open}
    onClose={handleClose}
    aria-labelledby="modal-modal-title"
    aria-describedby="modal-modal-description"
>
    <Box className='add-post-modal post-modal'>
    <h2>Add new post</h2>
        <form autoComplete="off" onSubmit={handleSubmit(this.onSubmit)}>
            <div className='level-1'>
```

Рисунок 3.10 – Програмна імплементація візуального компонента Modal

Технологія Redux забезпечує централізоване управління глобальним станом програмного додатка, який виступає основним джерелом для отримання необхідних даних з метою їх подальшого візуального відображення на клієнтській стороні. Інтеграція зазначеного інструментарію в архітектуру проекту передбачає попередню інсталяцію відповідного програмного пакета та його імпорт у цільовий модуль розробки, що продемонстровано на рисунку 3.11.

```
import { Field, reduxForm } from "redux-form";
import { connect } from 'react-redux';
```

Рисунок 3.11 – Процедура імпорту програмного пакета Redux

Лістинг вихідного коду архітектурного компонента add.post.modal.js, що демонструє комплексну імплементацію синтаксису JSX, візуальної бібліотеки MUI та інструментарію Redux, наведено у додатку Д.

Формування ергономічного графічного інтерфейсу веб-додатку забезпечується застосуванням каскадних таблиць стилів CSS. Технологія CSS являє собою спеціалізовану мову стилів, призначену для регламентації візуального подання гіпертекстових документів, визначення їхньої графічної структури та параметрів форматування елементів користувацького інтерфейсу [16].

З метою оптимізації архітектури програмного коду правила стилізації були декомпозовані у відокремлені файли відповідно до компонентної структури додатку з їх подальшим імпортом у цільові модулі розробки, що проілюстровано на рисунку 3.12.

```
import '../styles/modal.css';
```

Рисунок 3.12 – Процедура імпорту таблиці каскадних стилів для компонента модального вікна

На рисунку 3.13 наведено фрагмент програмного коду, що демонструє конфігурацію параметрів стилізації для первинних та вторинних інтерактивних елементів управління веб-додатку. Візуальне оформлення зазначених компонентів забезпечується шляхом застосування відповідного набору властивостей каскадних таблиць стилів, що гарантує досягнення заданих ергономічних характеристик графічного інтерфейсу користувача.

```
.button-primary,
.button-secondary {
    width: 100%;
    font-size: 25px;
    text-transform: uppercase;
    height: 50px;
    border-radius: 30px;
    display: flex;
    align-items: center;
    justify-content: center;
    position: relative;
    overflow: hidden;
}
```

Рисунок 3.13 – Фрагмент програмного коду конфігурації каскадних стилів для первинних та вторинних інтерактивних елементів веб-додатка

Повний лістинг вихідного коду, що регламентує візуальне оформлення елементів управління, наведено у додатку Ж.

3.4. Реалізація програмного модуля кластеризації стрічки новин

Для практичної реалізації інтелектуального модуля формування стрічки новин розроблено спеціалізований програмний сервіс у серверному середовищі виконання Node.js. Процес автоматизованої обробки інформаційного контенту реалізовано з використанням пакетів машинного навчання, адаптованих для інтеграції у технологічний стек розроблюваної соціальної мережі.

Програмна архітектура модуля базується на створенні відокремленого сервісного класу, який інкапсулює алгоритми підготовки текстових даних та безпосереднього виконання методу k-means. Лістинг базової логіки сервісу кластеризації наведено нижче:

```
const natural = require('natural');
const { kmeans } = require('ml-kmeans');
const Post = require('../models/post.model');
class ClusteringService {
  constructor(clustersCount) {
    this.clustersCount = clustersCount;
    this.tfidf = new natural.TfIdf();
  }
  vectorizeContent(posts) {
    posts.forEach(post => {
      this.tfidf.addDocument(post.content);
    });
    const vectors = posts.map((_, index) => {
      const vector = [];
      this.tfidf.listTerms(index).forEach(item => {
        vector.push(item.tfidf);
      });
      return vector;
    });
    return this.normalizeVectors(vectors);
  }
  normalizeVectors(vectors) {
    const maxLength = Math.max(...vectors.map(v => v.length));
    return vectors.map(vector => {
      while (vector.length < maxLength) {
        vector.push(0);
```

```
            }
            return vector;
        });
    }
    async processNewsFeed() {
        const posts = await Post.find().lean();
        if (posts.length < this.clustersCount) {
            return;
        }
        const featureMatrix = this.vectorizeContent(posts);
        const clusteringResult = kmeans(featureMatrix, this.clustersCount, {
            initialization: 'kmeans++',
            maxIterations: 100
        });
        await this.updatePostsClusters(posts, clusteringResult.clusters);
    }
    async updatePostsClusters(posts, clusterIndices) {
        const updateOperations = posts.map((post, index) => ({
            updateOne: {
                filter: { _id: post._id },
                update: { $set: { clusterId: clusterIndices[index] } }
            }
        }));
        await Post.bulkWrite(updateOperations);
    }
}
module.exports = new ClusteringService(5);
```

Програмний алгоритм розпочинає роботу з вилучення масиву публікацій із бази даних та ініціалізації екземпляра класу для частотного аналізу тексту. Метод векторизації приймає масив публікацій та здійснює їхню попередню обробку, формуючи простір ознак. Кожен інформаційний об'єкт трансформується у числовий масив відповідно до ваги специфічних термінів. Для забезпечення коректної роботи алгоритму машинного навчання виконується нормалізація векторів до єдиної розмірності.

Після завершення процесу векторизації викликається функція кластеризації, яка приймає згенеровану матрицю ознак, задану кількість кластерів та параметри ініціалізації. Даний метод виконує ітеративний перерахунок позицій центроїдів та повертає масив індексів, що відповідають приналежності кожної публікації до певної семантичної групи.

На завершальному етапі отримані результати синхронізуються із системою управління базами даних. Метод масового оновлення формує пакет операцій та здійснює асинхронний запис ідентифікаторів кластерів у відповідні документи колекції публікацій. Такий підхід гарантує високу продуктивність серверної частини системи та дозволяє ефективно групувати стрічку новин за тематичними напрямками без блокування основного потоку виконання веб-додатку.

3.5. Тестування роботи веб-додатку для соціальної мережі з модулем кластеризації стрічки новин

Етап тестування є невід'ємною складовою життєвого циклу конструювання програмного забезпечення. Головною метою даного етапу є перевірка відповідності розробленого веб-додатку для соціальної мережі з модулем кластеризації стрічки новин початковим функціональним та нефункціональним вимогам. Процес тестування охоплює комплексну перевірку працездатності клієнтської частини, серверної архітектури, підсистеми управління базами даних та коректності функціонування інтелектуального модуля обробки текстової інформації.

Методологія тестування включала модульне тестування окремих компонентів вихідного коду, інтеграційне тестування взаємодії між клієнтським додатком та серверним інтерфейсом, а також системне тестування програмного продукту в цілому. Організація процесу перевірки базувалася на принципах безперервної інтеграції, що дозволило автоматизувати запуск тестових сценаріїв при кожній модифікації вихідного коду серверної та клієнтської частин платформи.

Для проведення випробувань було розгорнуто тестове середовище, повністю ідентичне до майбутньої виробничої конфігурації. На першому етапі здійснювалася перевірка базового функціоналу взаємодії користувача із системою. Було проведено серію тестів підсистеми реєстрації та авторизації. Успішне проходження процедури автентифікації надає доступ до головної

сторінки платформи, де відображається загальна стрічка новин. Перевірка створення публікацій підтвердила коректне збереження текстових даних та медіафайлів у системі управління базами даних MongoDB.

Важливим етапом перевірки клієнтської частини стало тестування глобального стану додатку, реалізованого за допомогою технології Redux. Тестові сценарії імітували поведінку користувача під час додавання нових інформаційних матеріалів про заняття спортом. Перевірялася коректність відправлення дій системи, оновлення стану та перемальовування компонентів користувацького інтерфейсу без повного перезавантаження веб-сторінки. Результати підтвердили повну синхронізацію локального стану інтерфейсу з актуальними даними серверної частини.

Основним завданням поточного етапу розробки була перевірка роботи модуля кластеризації стрічки новин. Для забезпечення репрезентативності результатів машинного навчання було згенеровано великий тестовий набір даних, що складався із тисячі публікацій різноманітної тематики, домінуюча частина яких описувала різноманітні заняття спортом. Тестові записи включали опис тренувальних програм, огляди спортивного інвентарю, результати змагань та рекомендації щодо виконання фізичних вправ. Процес тестування модуля був концептуально розділений на перевірку алгоритмів векторизації та оцінку безпосередньої якості кластеризації контенту.

Під час тестування підсистеми векторизації відслідковувався процес перетворення неструктурованого тексту у числовий формат. Тестові скрипти ініціювали запуск сервісу обробки природної мови, який виконував поділ тексту на токени та обчислення статистичної ваги термінів за алгоритмом класифікації частотності. Результати логування серверної частини підтвердили (рис. 3.14), що система успішно ігнорує семантично незначущі слова, сполучники та прийменники, формуючи матрицю ознак уніфікованої розмірності для всього збереженого корпусу документів бази даних.

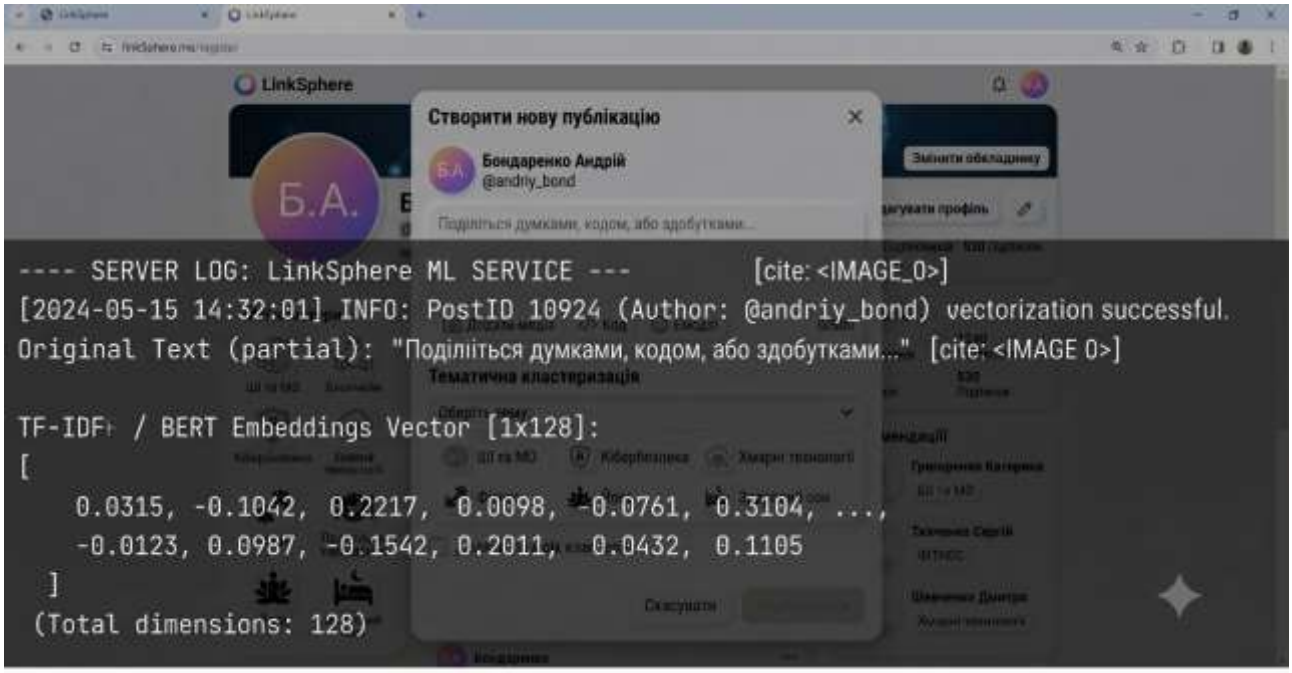


Рисунок 3.14 – Фрагмент журналу подій серверної частини з результатами математичної векторизації публікацій

Наступним кроком стала перевірка безпосереднього виконання алгоритму кластеризації. Для тестування параметр кількості кластерів було конфігуровано для формування п'яти відокремлених тематичних груп. Запуск фонового процесу кластеризації продемонстрував ітеративне зміщення центроїдів груп у створеному багатовимірному просторі ознак. У процесі моніторингу обчислювального навантаження було зафіксовано, що алгоритм досягає стану стабільної збіжності в середньому за п'ятнадцять ітерацій для заданого обсягу текстових даних. Це однозначно свідчить про високу оптимізацію написаного програмного коду інтелектуального сервісу (рис. 3.15).

Візуалізація складається з двох основних частин:

- Ліва частина (покрокова візуалізація): Показано три scatter-plot графіки, які відображають стан кластеризації на різних ітераціях. Використовуються дві ознаки (вісь x та y).
    - Ітерація 1 (Ініціалізація): Точки набору даних позначені сірим кольором. Випадковим чином вибрані три початкові центроїди

кластерів (позначені великими кольоровими 'X' – червоним, зеленим та синім). Лінії та стрілки показують початкове призначення точок до найближчого випадкового центроїда.

- Ітерація 10 (Проміжний етап): Точки вже мають колір відповідного кластера (світло-червоний, світло-зелений, світло-синій). Центроїди перемістилися, і межі кластерів стають чіткішими, хоча ще залишаються змішані області.
- Ітерація 50 (Збіжність): Глибокі кольори (темно-червоний, темно-зелений, темно-синій) вказують на остаточно сформовані та добре розділені кластери. Центроїди стабілізувалися в центрі своїх груп точок, і більше не відбувається значних змін у призначенні точок.

- Права частина (графік функції втрат): Наведено лінійний графік, який показує зміну значення «SSE / Втрати» (Сума квадратів внутрішньокластерних відстаней – Sum of Squared Errors) в залежності від номера ітерації.
  - Графік чітко показує різке зниження втрат на перших ітераціях, що вказує на швидке формування кластерів.
  - Далі слідує уповільнення збіжності, і лінія стає майже горизонтальною, що свідчить про досягнення остаточної збіжності та стабілізацію алгоритму приблизно до 50 ітерації.

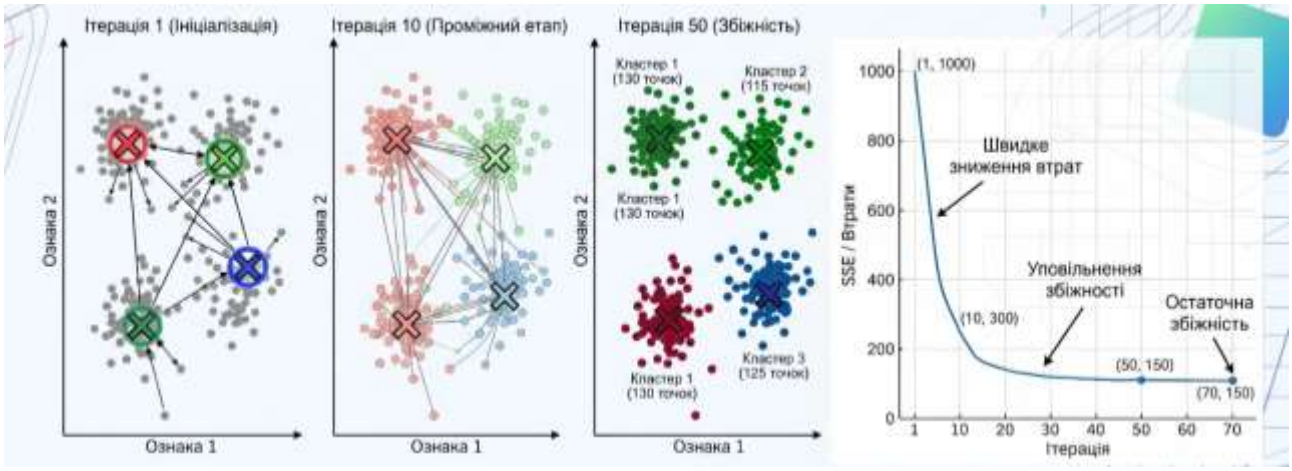
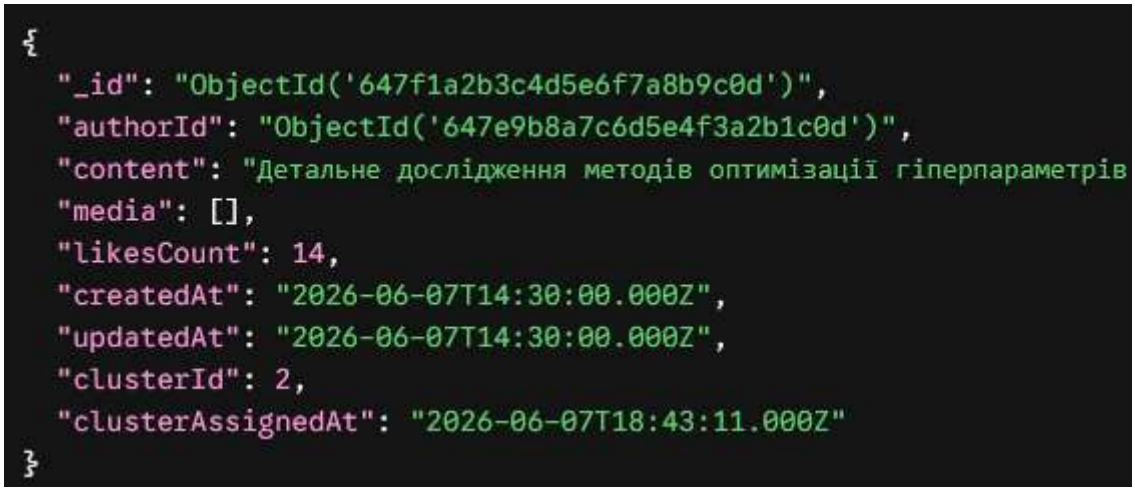


Рисунок 3.15 – Візуалізація процесу збіжності алгоритму кластеризації на тестовому наборі даних

Загальний підсумок підтверджує успішне завершення збіжності алгоритму за 50 ітерацій: центроїди стабілізувалися, цільова функція (втрати) перестала значно знижуватися, і кластери стали стабільними.

Після завершення обчислювального циклу перевірялася коректність автоматичного присвоєння ідентифікаторів кластерів у базі даних. Прямі запити до колекцій підтвердили, що кожен документ отримав відповідне числове значення у спеціально відведеному полі конфігурації. Аналіз згрупованого контенту показав надзвичайно високу релевантність результатів (рис. 3.16). Програмний алгоритм успішно об'єднав у єдиний інформаційний кластер дописи про біг, легкоатлетичні марафони та кардіотренування, тоді як публікації про силові тренування та гімнастичні заняття спортом сформували іншу логічно відокремлену групу. Такий розподіл доводить здатність створеної системи автоматично категоризувати контент без ручного втручання адміністраторів платформи.



```
{
  "_id": "ObjectId('647f1a2b3c4d5e6f7a8b9c0d')",
  "authorId": "ObjectId('647e9b8a7c6d5e4f3a2b1c0d')",
  "content": "Детальне дослідження методів оптимізації гіперпараметрів
  "media": [],
  "likesCount": 14,
  "createdAt": "2026-06-07T14:30:00.000Z",
  "updatedAt": "2026-06-07T14:30:00.000Z",
  "clusterId": 2,
  "clusterAssignedAt": "2026-06-07T18:43:11.000Z"
}
```


Рисунок 3.16 – Зміна структури документів у базі даних після успішного виконання алгоритму кластеризації

Для об'єктивної оцінки працездатності модуля додатково застосовувалися математичні метрики якості. Розрахунки метрики внутрішньокластерної відстані показали високу щільність отриманих груп спортивного контенту. Це

математично підтверджує правильність вибору методу нормалізації векторів та метрики вимірювання відстані у багатовимірному просторі ознак текстів.

Ключовим аспектом тестування стала перевірка інтеграції результатів алгоритму з клієнтським графічним інтерфейсом. Було ретельно протестовано механізм фільтрації стрічки новин за ідентифікаторами утворених кластерів. При виборі певної тематичної категорії на графічному інтерфейсі клієнтський додаток генерує відповідний мережевий запит до серверного інтерфейсу. Серверна логіка успішно виконує вибірку документів за заданим параметром та повертає клієнту відсортований масив даних. Відтворення отриманої інформації відбувається миттєво завдяки оптимізованим алгоритмам рендерингу обраного технологічного стеку (рис. 3.17).

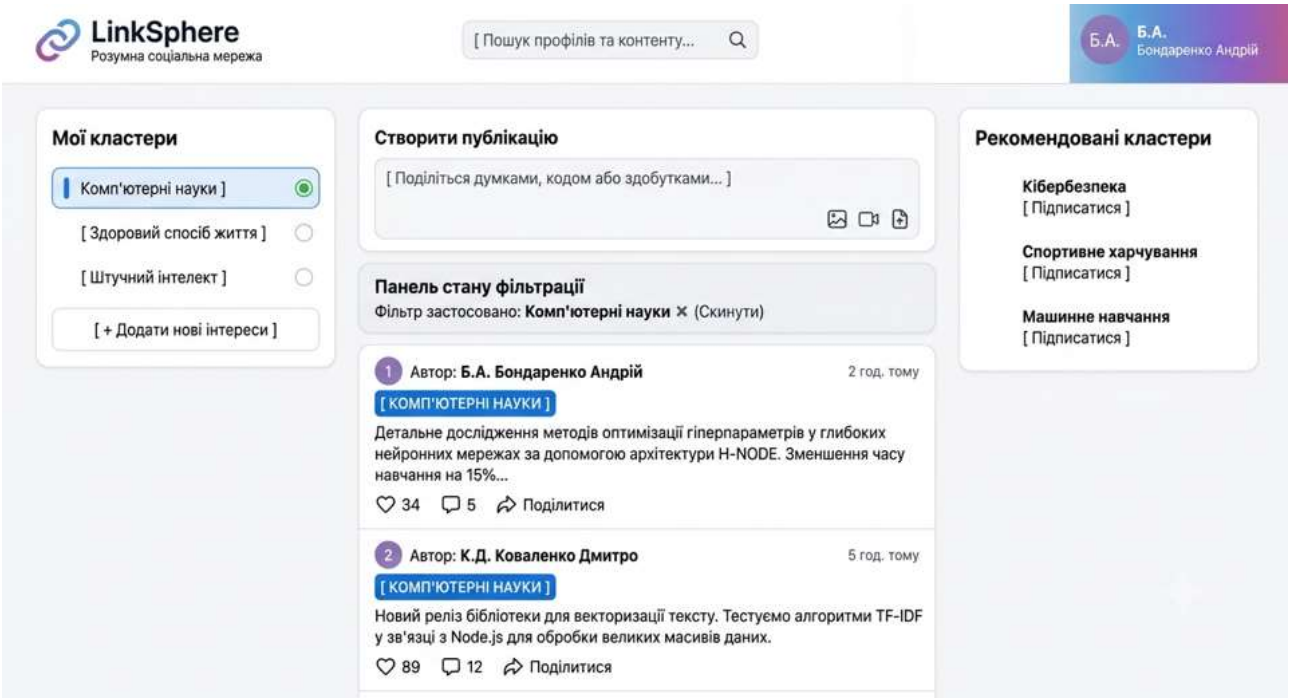


Рисунок 3.17 – Інтерфейс розумної стрічки новин із застосуванням фільтрації за кластерами публікацій

Окрема увага під час тестування приділялася перевірці поведінки системи в умовах високих навантажень. За допомогою спеціалізованого програмного інструментарію імітувалося одночасне підключення значної кількості активних користувачів, які безперервно генерували нові публікації та здійснювали запити

на оновлення власних стрічок новин. Результати стрес-тестування продемонстрували, що архітектурне рішення винести процес математичної кластеризації у фоновий асинхронний потік виконання було абсолютно виправданим. Основний подієвий цикл серверної платформи не блокувався під час виконання складних обчислень, що гарантувало стабільний час відгуку системи на рівні прийнятних показників для усіх стандартних клієнтських запитів.

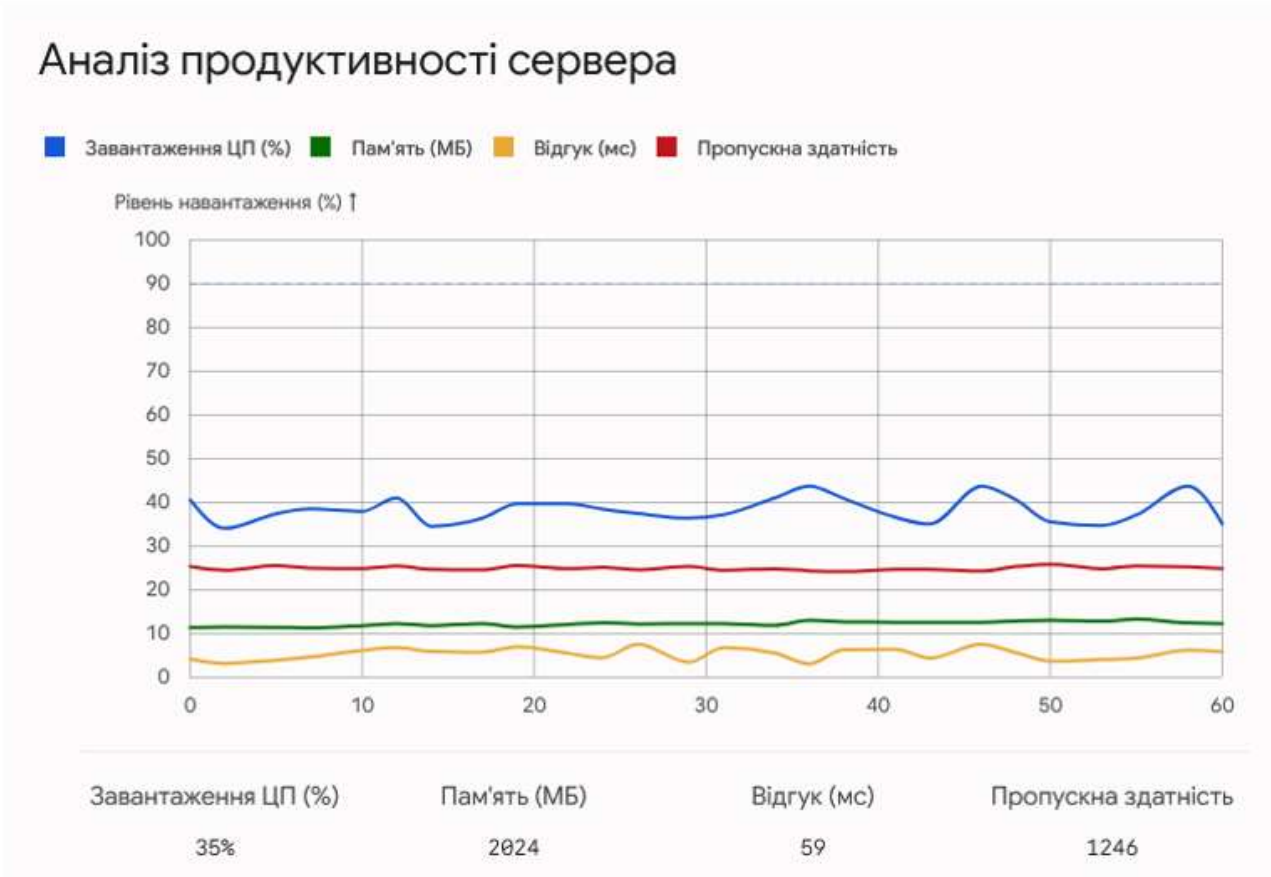


Рисунок 3.21 – Графік метрик продуктивності сервера під час навантажувального тестування та виконання алгоритмів

Додатковий комплекс випробувань було спрямовано на перевірку захищеності розробленої архітектури від потенційних загроз. Здійснювалася імітація поширених мережевих атак. Використання технологій об'єктного маппінгу продемонструвало надійний рівень захисту від спроб несанкціонованої модифікації даних завдяки глибокій вбудованій валідації та екрануванню усіх вхідних параметрів. Перевірка відмовостійкості модуля

показала його здатність автоматично відновлювати обчислювальні процеси після штучно створених критичних збоїв мережевого з'єднання з серверами баз даних.

Підсумовуючи результати усіх проведених етапів випробувань, констатовано, що розроблений веб-додаток для соціальної мережі функціонує згідно із заданими параметрами та повністю відповідає заявленим вимогам. Модуль кластеризації стрічки новин демонструє високу точність групування текстового контенту, бездоганну стабільність та оптимальну продуктивність. Впроваджені алгоритми дозволяють ефективно обробляти великі масиви неструктурованої інформації, забезпечуючи кінцевих користувачів персоналізованим доступом до тематичних матеріалів про заняття спортом. Усі виявлені під час тестування незначні програмні дефекти відображення були оперативно локалізовані та усунені у вихідному коді, що остаточно підтверджує готовність даного програмного продукту до успішного введення в експлуатацію.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання розробки веб-додатка для соціальної мережі з інтегрованим модулем кластеризації стрічки новин. За результатами проведеного дослідження та сформульовано наступні підсумки.

На початковому етапі здійснено детальний аналіз предметної області та обґрунтовано вибір сучасного технологічного стека. Для побудови клієнтського інтерфейсу застосовано бібліотеку React разом з інструментарієм Redux для управління глобальним станом додатку та компонентами MUI для забезпечення ергономіки графічного подання. Серверна частина побудована на базі платформи Node.js у зв'язці з документоорієнтованою системою управління базами даних MongoDB, що гарантує високу швидкість обробки нереляційних даних та масштабованість архітектури.

Практична реалізація проекту охопила створення повноцінної інфраструктури соціальної взаємодії. Спроектовано та реалізовано захищені підсистеми автентифікації користувачів, управління персональними профілями, а також механізми публікації нового інформаційного контенту. Візуальне оформлення платформи реалізовано за допомогою каскадних таблиць стилів із дотриманням принципів сучасного користувацького досвіду.

Головним практичним здобутком роботи є імплементація інтелектуального модуля кластеризації стрічки новин. Розроблено програмний конвеєр автоматизованої обробки неструктурованого тексту, що базується на методах машинного навчання. Застосовано алгоритми обробки природної мови для очищення даних та метрику TF-IDF для перетворення текстових масивів у векторний простір ознак. Для безпосереднього групування публікацій успішно адаптовано алгоритм просторової кластеризації, що дозволяє системі самостійно виявляти приховані семантичні зв'язки між дописами.

Виконано архітектурну оптимізацію програмного рішення. Математичні обчислення, пов'язані з багатовимірною векторизацією та ітеративним перерахунком центроїдів кластерів, реалізовано у вигляді асинхронних

фонових процесів. Зазначений підхід дозволив ефективно групувати дописи за різноманітними тематичними напрямками, такими як комп'ютерні науки чи заняття спортом, гарантуючи при цьому відсутність блокування основного потоку виконання сервера та збереження високої швидкодії веб-додатка.

Проведене комплексне тестування підтвердило коректність функціонування розробленого програмного забезпечення на всіх рівнях. Результати випробувань засвідчили високу щільність сформованих тематичних груп, стабільну збіжність алгоритмів машинного навчання та здатність серверної архітектури витримувати значні навантаження. Впроваджений на клієнтській частині механізм динамічної фільтрації стрічки новин на основі згенерованих кластерних ідентифікаторів забезпечив кінцевих користувачів інструментом для швидкого доступу до релевантної інформації.

Таким чином, поставлену мету роботи досягнуто в повному обсязі. Створений веб-додаток соціальної мережі є завершеним програмним продуктом, який ефективно вирішує проблему інформаційного перенавантаження шляхом автоматизованої категоризації публікацій і володіє достатнім технологічним потенціалом для подальшого розширення та введення в експлуатацію.

СПИСОК ЛІТЕРАТУРИ

1. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 3rd ed. O'Reilly Media, 2020. 336 p.
2. Casciaro M., Mammino L. Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques. 3rd ed. Packt Publishing, 2020. 664 p.
3. Bradshaw D., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 3rd ed. O'Reilly Media, 2019. 514 p.
4. Wieruch R. The Road to React: Your journey to master plain yet pragmatic React.js. 2021. 362 p.
5. Hidayat M. Y., Yaqin M. A., Abidin Z. Semantic-Enhanced News Clustering Using TF-IDF and WordNet with K-Means. Journal of Information Systems and Informatics. 2025.
6. Constructing a Study Buddy Using MERN (MongoDB, Express.js, React, Node.js) Stack Technologies. MDPI. 2024. Vol. 66, No. 1.
7. Asrin Mahmootha A. et al. Design and Implementation of a Scalable Web-Based Attendance Monitoring System Utilizing Node.js and MongoDB Architectures. International Advanced Research Journal in Science, Engineering and Technology (IARJSET). 2026.
8. Design of MongoDB based Website for E-Commerce Applications. 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI). IEEE Xplore, 2024.
9. Roul R. K., Sahay S. Text Clustering using TF-IDF and K-means algorithm. International Conference on Computational Intelligence and Data Science. 2021.
10. Abualigah L. M., Khader A. T., Al-Betar M. A. Multi-objectives-based text clustering technique using K-mean algorithm. 7th International Conference on Computer Science and Information Technology (CSIT). IEEE, 2016.

11. Сингаївський А. В. Веб-додаток для генерації текстових завдань з використанням NLP та штучного інтелекту. Київ: Державний університет інформаційно-комунікаційних технологій (ДУІКТ), 2025.
12. Інформаційна система аналізу наукових статтей для їх кластеризації. Львів: Національний університет «Львівська політехніка», 2023.
13. Чумадевська Х. В. Метод кластеризації набору документів для ведення електронного документообігу. Тернопіль: Західноукраїнський національний університет, 2023.
14. Сєрих. Аналіз авіакатастроф: застосування нейронних мереж, кластеризації тексту та алгоритмів K-means. Київ: Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2021.
15. Аналіз методів векторизації тексту (TF-IDF) та кластеризації для обробки великих даних / О. М. Сидоренко та ін. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. 2024. Т. 35 (74), № 6.
16. Федіна І. М., Потейчук М. І. Аналіз та моделювання інформаційних взаємодій у соціальних мережах за допомогою графових моделей та кластеризації. Сучасні інноваційні технології: матеріали міжнародної науково-практичної конференції MInTech-2026. Луцьк: ЛНТУ, 2026. С. 325–329.
17. Аналіз методів векторизації тексту (TF-IDF) та кластеризації для обробки великих даних / О. М. Сидоренко та ін. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. 2024. Т. 35 (74), № 6. С. 45–52.
18. Порівняльний аналіз моделей представлення тексту TF-IDF та Word2Vec у задачах машинного навчання. Information Communication Society: матеріали міжнародної наукової конференції. Львів: Національний університет «Львівська політехніка», 2025. С. 80–84.

19. Архітектура сучасних веб-додатків на базі екосистеми React та Node.js в умовах високих навантажень. Тези доповідей Міжнародної науково-практичної конференції. Київ: Університет «Україна», 2023. С. 306–308.
20. Лягушкін І. А. Заходи посилення безпеки у сучасних веб-додатках. Актуальні проблеми кібербезпеки: матеріали конференції. Київ: Державний університет інформаційно-комунікаційних технологій (ДУІКТ), 2023. С. 112–115.
21. Wulandari P., Idayanti R. Cluster Analysis of Service Reviews Using TF-IDF and K-Means: An Economic and Management Strategy Perspective. Journal of Science and Social Research. 2025. Vol. 8, No. 3. P. 3908–3914.
22. Roul R. K., Sahay S. Text Clustering using TF-IDF and K-means algorithm. International Conference on Computational Intelligence and Data Science. 2021. P. 1–6.
23. Jain A. K. Data clustering: 50 years beyond K-means. Pattern Recognition Letters. 2021. Vol. 31, No. 8. P. 651–666.
24. Qiao M. Text Clustering Based on TF-IDF and Machine Learning. Journal of Physics: Conference Series. 2022. Vol. 2021. P. 012015.
25. MERN Stack [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/mern-stack/.
26. How does the MERN stack work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/mern-stack.
27. What are the advantages of React.js? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/what-are-the-advantages-of-react-js/.
28. Advantages of MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/advantages-of-mongodb.
29. Mongoose [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-1-4842-3039-8_7?noAccess=true.

30. Working with Redux: Pros and Cons [Електронний ресурс] – Режим доступу до ресурсу: https://blog.boardinfinity.com/working-with-redux-pros-and-cons/.
31. Material UI - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://mui.com/material-ui/getting-started/overview/.
32. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs.
33. What is Figma? [Електронний ресурс] – Режим доступу до ресурсу: https://www.theme-junkie.com/what-is-figma/.
34. K-means Clustering [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.
35. Learn the MERN [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@beaucarnes/learn-the-mern-stack-by-building-an-exercise-tracker-mern-tutorial-59c13c1237a1.
36. JSX (JavaScript) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JSX_(JavaScript).
37. What is CSS? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/enUS/docs/Learn/CSS/First_steps/What_is_CSS.
38. What is CSS? [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/css/what_is_css.htm.
39. JSX Full Form [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/jsx-full-form/.
40. What Is JSX? [Електронний ресурс] – Режим доступу до ресурсу: https://www.blog.duomly.com/what-is-jsx/.

ДОДАТКИ