МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 2001–“С” 01.11.2023. 021 ПЗ

РЯБЧЕНКА МИКОЛИ МИКОЛАЙОВИЧА

2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Болбот І. М., д.т.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2024 р.	**____ _____ 2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему __Інтелектуальна система обліку веб проектів_____

Спеціальність ___122 - Комп'ютерні науки_____
(код і назва)

Освітня програма ____Інформаційно управляючі системи та технології_
(назва)

Орієнтація освітньої програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____кандидат технічних наук, доцент_____ _____ Голуб Б.Л.__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____к.т.н., доцент_____ _____ _____Кириченко В.В._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Рябченко М.М._____
(підпис) (ПІБ студента)

КИЇВ - 2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"16" вересня 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Рябченко Микола Миколайович
(прізвище, ім'я, по батькові)

Спеціальність 122 - Комп'ютерні науки
(код і назва)

Освітня програма Інформаційно управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Інтелектуальна система обліку веб проектів

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру 28 листопада 2024р.
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи отримання списку проектів на обрані теми у вигляді посилань на тематичні сайти або сторінки github

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області
2. Моделювання системи
3. Розробка системи
4. Результати дослідження

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "16" вересня 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Кириченко В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Рябченко М.М.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Швидке зростання та ускладнення веб-проектів підкреслили **актуальність** в інноваційних рішеннях у сфері обліку та управління ресурсами. Традиційні методи часто не справляються з динамічною та багатогранною природою сучасної веб-розробки. Ця дипломна робота присвячена створенню інтелектуальної системи обліку, яка оптимізує процеси, підвищує точність і підтримує прийняття рішень на основі даних.

Предметом даного дослідження є дослідження передових методів та інструментів обліку в контексті веб-проектів. Зокрема, **об'єктом** дослідження є робочі процеси та системи управління ресурсами, властиві середовищам веб-проектів. Звертаючись до цих аспектів, дослідження має на меті створити надійну основу для інтелектуальних систем обліку, адаптованих до унікальних вимог цієї сфери.

Метою дослідження є покращення галузі створення веб-проектів, розробивши систему яка замінює застарілі ручні процеси складною автоматизованою альтернативою. Запропонована система інтегрує інтелектуальні алгоритми, прогнозні моделі та аналітичні можливості, забезпечуючи безперебійне відстеження та звітування. Ця зміна не тільки

зменшує кількість помилок, але й підвищує ефективність, дозволяючи керівникам проектів швидко приймати обґрунтовані рішення.

Завдання роботи поєднує системний аналіз, моделювання, проектування та експериментальну перевірку. Дослідження починається з поглибленого аналізу існуючої практики обліку, визначення ключових обмежень і областей для вдосконалення. Потім розробляється концептуальна модель, що детально описує архітектуру та компоненти запропонованої системи. Система впроваджується за допомогою сучасних інструментів і технологій з подальшим ретельним тестуванням для оцінки її ефективності в реальних сценаріях.

Наукова новизна даного дослідження полягає в підході до інтеграції інтелектуальних систем в систему обліку веб-проектів. На відміну від звичайних методів, запропоноване рішення використовує передову аналітику даних і автоматизацію для адаптації до мінливих вимог проекту, пропонуючи рівень точності та гнучкості, який раніше був недосяжний. Етап тестування продемонстрував здатність системи покращувати розподіл ресурсів, покращувати відстеження завдань і оптимізувати процеси звітності, підтверджуючи її практичну корисність.

Дана робота поділена на кілька ключових розділів, щоб забезпечити повне розуміння дослідження та його результатів. Він починається з детального аналізу предметної області, вивчення поточної практики та визначення критичної потреби в інноваціях. Після цього розробляється концептуальна модель, яка служить основою для інтелектуальної системи. Етап проектування та реалізації зосереджується на створенні інтуїтивно зрозумілої ефективної системи, яка включає модулі для введення, обробки та візуалізації даних. Результати дослідження демонструють ефективність системи, представляючи кількісні та якісні докази її впливу на облік веб-проектів.

Це дослідження пропонує значний внесок у сферу управління проектами та обліку, представляючи практичне та інноваційне рішення

проблем, з якими стикаються під час управління веб-проектами. Інтелектуальна система, розроблена в результаті цього дослідження, являє собою великий крок вперед, поєднуючи передові технології з глибоким розумінням складності сучасної веб-розробки.

Магістерська робота містить 76 сторінок, в тому числі 19 рисунків, 2 додатків. Вона посилається на 34 джерела, включаючи наукові статті, книги та електронні ресурси.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Швидкий розвиток веб-технологій значно збільшив обсяг і складність веб-проектів. Ці проекти, які варіюються від веб-сайтів малого бізнесу до великомасштабних веб-додатків, включають багато зацікавлених сторін, різноманітні ресурси та складні робочі процеси. Ефективне управління такими проектами вимагає точного обліку завдань, часових рамок, бюджетів і людських ресурсів, що має вирішальне значення для підтримки ефективності, дотримання термінів і контролю витрат.

Традиційні методи обліку та відстеження проектів часто покладаються на ручні процеси або загальні інструменти, яким бракує гнучкості, щоб адаптуватися до динамічної природи веб-розробки. Ці підходи схильні до помилок, неефективності та вузьких місць, особливо в середовищах, де зміни відбуваються часто, і швидкі рішення важливі. Крім того, вони не забезпечують рівень аналізу даних і розуміння, необхідних для проактивного прийняття рішень і оптимізації ресурсів.

Сучасні веб-проекти вимагають інтелектуальних систем, здатних автоматизувати повторювані завдання, аналізувати дані та надавати корисну

інформацію в реальному часі. Такі системи повинні вирішувати ключові проблеми, включаючи розподіл завдань, моніторинг ефективності та використання ресурсів, при цьому бути адаптованими до конкретних вимог окремих проектів.

Проблемна область також включає інтеграцію багатьох джерел даних, таких як відстеження часу, фінансові дані та етапи проекту, у єдину систему. Забезпечення точності та узгодженості цих вхідних даних має вирішальне значення для забезпечення надійної основи для прийняття рішень. Крім того, потреба в зручних для користувача інтерфейсах і настроюваних інструментах звітності є важливою для того, щоб всі зацікавлені сторони, від керівників проектів до членів команди, могли ефективно взаємодіяти з системою.

Ця робота спрямована на вирішення цих проблем, пропонуючи інтелектуальну систему обліку, спеціально розроблену для веб-проектів. Завдяки використанню передових технологій, таких як машинне навчання, прогнозна аналітика та автоматизація, система забезпечить надійне рішення для відстеження та оптимізації робочих процесів проекту, забезпечуючи більшу точність, ефективність і адаптивність у управлінні веб-проектами.

Веб-проекти стали наріжним каменем сучасної цифрової інфраструктури, що охоплює веб-сайти, веб-додатки та різні онлайн-сервіси. У цих проектах часто беруть участь різноманітні зацікавлені сторони, включаючи розробників, дизайнерів, творців контенту та менеджерів проектів. Вони охоплюють численні галузі, такі як охорона здоров’я, електронна комерція, фінанси та освіта, з унікальними вимогами до функціональності, масштабованості та обслуговування. Ефективне управління та класифікація цих проектів має вирішальне значення для забезпечення їх успіху, особливо враховуючи те, що обсяг і складність веб-проектів постійно зростають.

Інтелектуальна система обліку веб-проектів служить централізованим рішенням для збору, класифікації та керування даними, пов’язаними з цими проектами. Він розроблений для вирішення проблем традиційних ручних систем, які часто борються з неефективністю, непослідовністю та нездатністю

зберегти видалену або історичну інформацію. Інтелектуальна система надає розширені можливості, такі як автоматизований збір даних, тематична класифікація та пошукові записи, що робить її безцінним інструментом для розробників, керівників проектів та інвесторів.

Ключові аспекти предметної області включають:

Автоматизований збір даних

Веб-проекти часто розповсюджуються на кількох платформах, таких як GitHub, GitLab та інші служби контролю версій або хостингу. Ці платформи надають велику кількість метаданих, включаючи описи проектів, учасників, використані технології та оновлення. Автоматизація збору цих даних забезпечує повноту й точність, одночасно зменшуючи необхідні ручні зусилля.

Тематична класифікація

Важливим аспектом обліку веб-проектів є можливість класифікувати проекти на основі їх домену чи теми. Наприклад, проекти можуть бути пов’язані з такими темами, як «електронна комерція», «медицина» або «криптовалюта». Використовуючи обробку природної мови (NLP) і машинне навчання, система може аналізувати метадані та описи проекту, щоб точно класифікувати проекти[1].

Інтеграція бази даних і відстеження історії

Інтелектуальна система оснащена надійною базою даних, здатною зберігати не тільки активні проекти, але й історичні дані, включаючи інформацію про проекти, які більше не можуть бути доступні в Інтернеті. Це гарантує збереження цінної інформації, що дозволяє аналізувати тенденції та використовувати в майбутньому посилання.

Можливість пошуку та доступність

Інвесторам, менеджерам проектів або зацікавленим сторонам часто потрібно ідентифікувати проекти, які відповідають певним критеріям, наприклад проекти в певній галузі або з використанням певних технологій. Система надає розширені можливості пошуку, що дозволяє швидко й

ефективно отримувати відповідні проекти на основі тематичних або технічних атрибутів.

Застосування в різних галузях

Додатки системи поширюються не тільки на розробників і менеджерів проектів, але й на інвесторів і дослідників. Наприклад, інвестор, який хоче фінансувати інноваційні проекти в сфері охорони здоров'я, може використовувати систему для виявлення перспективних веб-проектів. Так само дослідники можуть використовувати його для вивчення тенденцій і досягнень у різних областях.

Завдяки автоматизації та покращенню традиційних процесів ця система пропонує оптимізоване, масштабоване рішення для управління дедалі складнішими веб-проектами.

1.2 Огляд існуючих рішень

Розглянемо існуючі рішення предметної області.

AngelList — це динамічна платформа, створена для об'єднання стартапів, інвесторів і шукачів роботи в екосистемі стартапів, що швидко розвивається. Запущений у 2010 році, він зарекомендував себе як життєво важливий інструмент для бізнесу на ранніх стадіях, надаючи ресурси для забезпечення фінансування, найму талантів і сприяння зростанню. Його функціональність охоплює кілька сфер, пропонуючи стартапам можливість продемонструвати своє бачення та прогрес за допомогою детальних профілів. Ці профілі не лише висвітлюють місію та продукти компанії, але й служать магнітом для інвесторів і потенційних працівників, які відповідають цілям стартапу.

Для інвесторів AngelList пропонує інструменти, які спрощують процес виявлення та оцінки потенційних інвестицій. За допомогою таких функцій, як синдикати, інвестори можуть об'єднувати ресурси для спільної підтримки стартапів, часто під керівництвом досвідченого провідного інвестора. Крім

того, підібрані фонди надають різноманітні інвестиційні можливості, а інструменти для виявлення угод дозволяють інвесторам досліджувати підприємства на основі галузі, географії чи стадії фінансування[2].

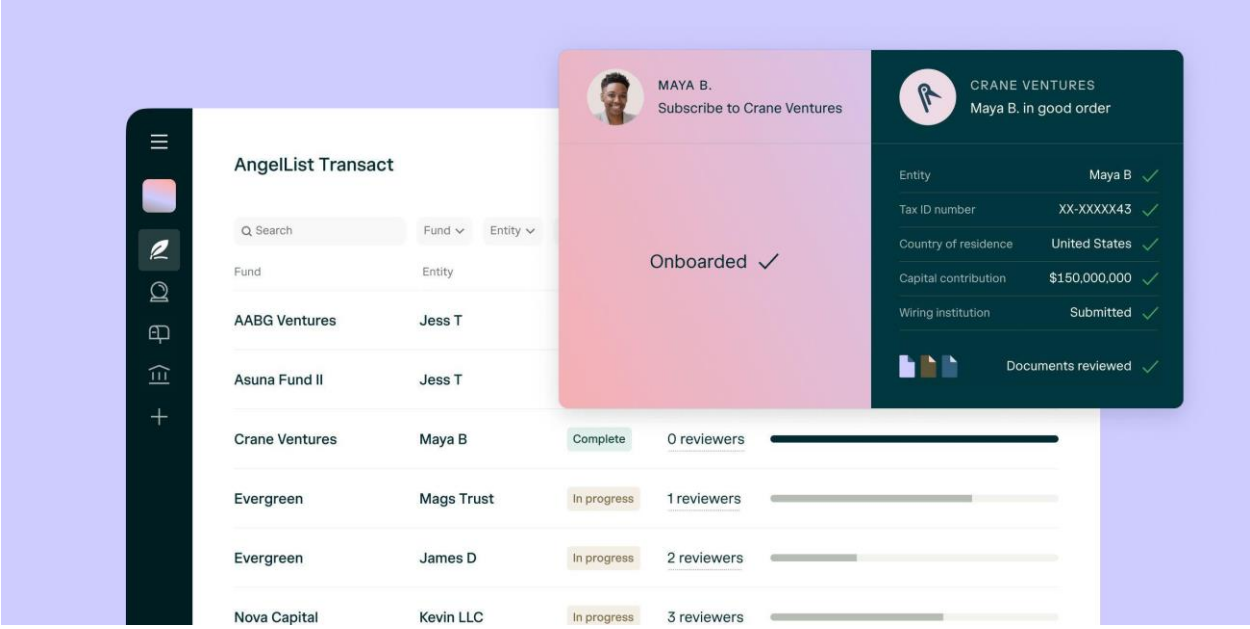


Рис. 1 Програмне забезпечення «AngelList»

Шукачі роботи отримують переваги від величезного ринку, де вони можуть досліджувати можливості в різних галузях, починаючи від технологій до охорони здоров'я та фінтех. AngelList Talent забезпечує прямий доступ до тисяч ролей стартапів, що дозволяє кандидатам співпрацювати з інноваційними компаніями без традиційних бар'єрів при наймі. Зі збільшенням уваги до віддаленої роботи платформа також адаптувала свої пропозиції, щоб включити фільтри для віддалених вакансій, що полегшує підключення стартапів і шукачів роботи до цього все більш гнучкого робочого середовища.

Підприємці отримують видимість і доступ до ресурсів, необхідних для масштабування своїх підприємств. У той же час інвестори можуть виявити багатообіцяючі можливості в компаніях на ранній стадії, хоча вони повинні орієнтуватися в невід'ємних ризиках, пов'язаних з інвестиціями в стартапи. Хоча AngelList працює в усьому світі, його найвидатніші функції обслуговують регіони з потужними екосистемами стартапів, наприклад Сполучені Штати.

Подолаючи розрив між амбітними стартапами, далекоглядними інвесторами та талановитими професіоналами, AngelList продовжує процвітати як важливий центр для інновацій. Його здатність адаптуватися до мінливого ландшафту підприємництва та віддаленої роботи підкреслює його актуальність і стійкий успіх у підтримці спільноти стартапів.

Bitbucket — це веб-платформа, призначена для контролю версій і співпраці, насамперед орієнтована на команди розробників, які працюють над програмними проектами. Належить Atlassian, він легко інтегрується з іншими інструментами Atlassian, такими як Jira та Confluence, що робить його популярним вибором для команд, які прагнуть оптимізувати свої робочі процеси розробки[5].

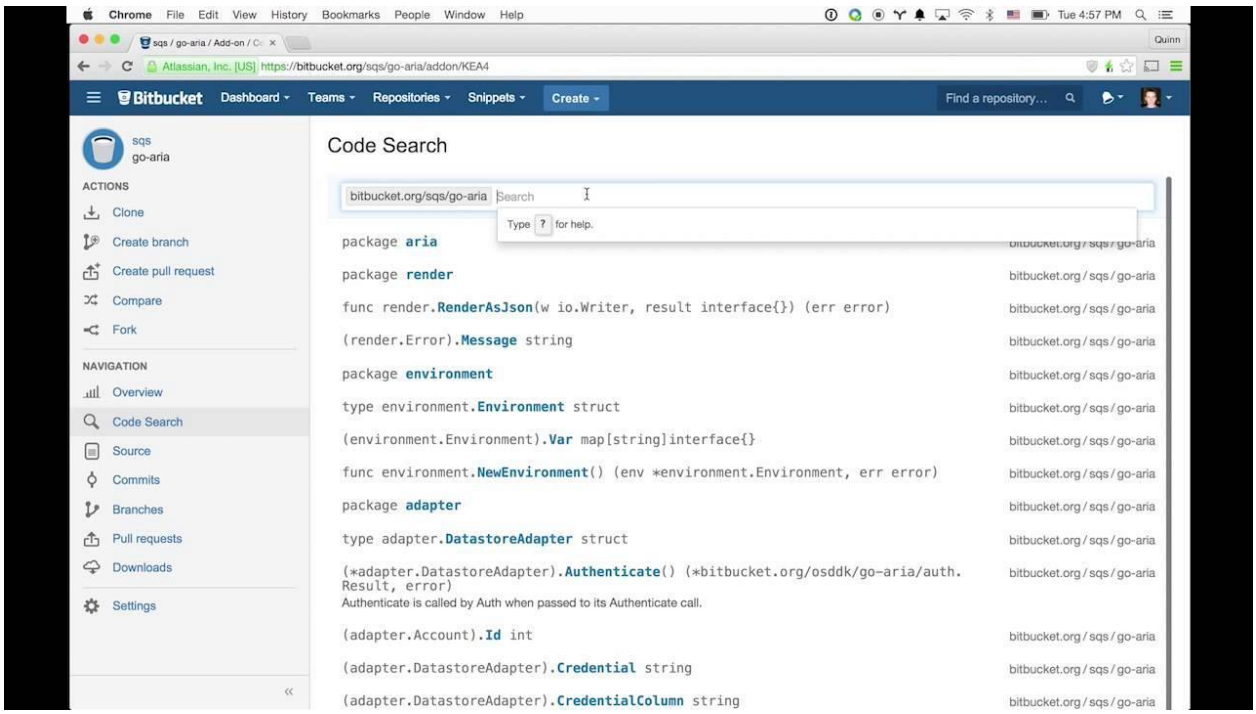


Рис. 2 Програмне забезпечення «Bitbucket»

За своєю суттю Bitbucket надає репозиторії для керування та розміщення проектів на основі Git. Розробники можуть безпечно зберігати свій код, відстежувати зміни та співпрацювати над оновленнями за допомогою надійних функцій контролю версій. Bitbucket підтримує як приватні, так і публічні репозиторії, з акцентом на безпеку для приватних проектів, які віддають перевагу організаціям і командам, яким потрібен контрольований доступ до їх кодової бази.

Однією з видатних особливостей Bitbucket є його здатність полегшувати командну співпрацю. Запити на витягування дозволяють розробникам переглядати та обговорювати зміни коду перед їх об’єднанням, забезпечуючи вищу якість коду та менше помилок. Вбудовані інструменти перегляду коду заохочують спільне прийняття рішень, тоді як вбудовані коментарі забезпечують простий спосіб обговорення окремих рядків коду.

Платформа також вирізняється можливостями інтеграції. Bitbucket легко з’єднується з конвеєрами CI/CD, дозволяючи командам автоматизувати збирання, тестування та розгортання безпосередньо зі своїх репозиторіїв. Для розробників, які використовують Jira, інтеграція забезпечує відстеження між проблемами та кодом, спрощуючи керування проектами та покращуючи видимість.

Підтримка Bitbucket для різних робочих процесів робить його адаптованим до команд будь-якого розміру та методології, від невеликих стартапів до великих підприємств. Він пропонує такі функції, як дозволи на розгалуження та перевірки злиття для підтримки стандартів кодування та захисту критичних розгалужень. Крім того, його хмарна пропозиція забезпечує доступність з будь-якого місця, тоді як локальна версія (Bitbucket Data Center) забезпечує гнучкість для організацій із суворішими вимогами відповідності.

У умовах дедалі більшої конкуренції Bitbucket виділяється своєю інтеграцією в екосистему Atlassian і фокусом на безпеці та співпраці корпоративного рівня. Він залишається основним рішенням для команд, які прагнуть підтримувати ефективні робочі процеси та створювати високоякісний код у безпечному та масштабованому середовищі.

1.3 Постановка завдання

Основною метою цієї магістерської роботи є розробка інтелектуальної системи для обліку та управління веб-проектами. Система призначена для

відстеження та організації різних аспектів веб-проектів, таких як деталі проекту, пов’язані домени та керування вмістом. Центральним компонентом цієї системи є надійна база даних, де зберігається та класифікується інформація про проект. Учасники можуть вводити дані проекту, а система інтелектуально класифікує та отримує деталі на основі тем проекту або конкретних атрибутів.

Основна мета цієї системи — надати користувачам — таким як менеджери проектів, інвестори та розробники — швидкий доступ до важливої інформації про веб-проекти. Вони можуть містити деталі щодо тем проекту (наприклад, електронна комерція, охорона здоров’я, криптовалюта), зв’язок проекту з певними доменами чи каналами та статус різних етапів проекту (наприклад, активний, заархівований, видалений). Розподіляючи проекти за категоріями, система полегшує пошук за певними темами, дозволяючи користувачам знаходити відповідні проекти на таких платформах, як GitHub, GitLab тощо.

Система працює в інтерактивному режимі зі зручним інтерфейсом. За допомогою серії меню користувачі можуть вибирати різні дії, такі як пошук проектів за темою або отримання видалених даних проекту. Програма реагує на вибір користувача, надаючи відповідні результати та пропонуючи додаткові дії залежно від контексту.

Ця інтелектуальна система обліку веб-проектів розроблена для підвищення ефективності для окремих осіб або організацій, які беруть участь в управлінні проектами, пропонуючи швидкий, організований і відповідний доступ до даних проекту.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Загальні відомості

UML (Unified Modeling Language) — це стандартизована візуальна мова, яка широко використовується в розробці програмного забезпечення для представлення та документування дизайну програмних систем. Він пропонує різноманітні методи побудови діаграм, які допомагають як розробникам, так і зацікавленим сторонам візуалізувати, визначити та повідомити про функціональність і структуру системи. UML, який в основному використовується в об’єктно-орієнтованій розробці, підтримує моделювання як статичної структури, так і динамічної поведінки системи[6].

UML відомий у всьому світі завдяки своїй послідовності та універсальності. Розроблений Object Management Group (OMG), він забезпечує уніфікований підхід до моделювання, який можна застосовувати в різних методологіях розробки програмного забезпечення, включаючи Agile та Waterfall. Його візуальний характер робить його інтуїтивно зрозумілим інструментом для зображення складних систем, допомагаючи покращити розуміння як технічними, так і нетехнічними зацікавленими сторонами.

Діаграми UML зазвичай поділяються на дві категорії: структурні та поведінкові. Структурні діаграми зосереджуються на статичних компонентах системи, ілюструючи, як елементи системи пов’язані один з одним. До них належать діаграми класів, які показують зв’язки між класами, діаграми компонентів, що ілюструють модульну структуру системи, та діаграми об’єктів, які фіксують екземпляри класів у певний момент часу. Інші структурні діаграми, як-от діаграми пакетів і розгортання, організовують систему в логічні одиниці та показують, як компоненти програмного забезпечення розподіляються між обладнанням.

З іншого боку, діаграми поведінки описують динамічні аспекти системи, наприклад те, як вона поводиться у відповідь на різні події. Діаграми

варіантів використання тут є звичайним інструментом, оскільки вони визначають взаємодію між користувачами та системою. Діаграми послідовності та зв’язку показують, як об’єкти взаємодіють протягом певного часу, тоді як діаграми діяльності відображають потік дій і рішень у системі. Діаграми станів моделюють, як об’єкт переходить між різними станами на основі певних подій. Тимчасові діаграми використовуються для представлення станів об’єктів у часі, що особливо корисно в системах реального часу.

Однією з ключових переваг UML є те, що він сприяє чіткій комунікації між зацікавленими сторонами, дозволяючи як технічним, так і нетехнічним особам зрозуміти дизайн системи. Він також надає спосіб візуалізації складних систем, що може допомогти командам виявити проблеми на ранніх стадіях процесу розробки. UML підтримує послідовну та організовану документацію, що полегшує підтримку та масштабування систем у міру їх розвитку. Крім того, він забезпечує гнучкість, оскільки його можна застосовувати до широкого діапазону типів систем, від об’єктно-орієнтованих до сервіс-орієнтованих архітектур[7].

На практиці UML є цінним інструментом протягом життєвого циклу розробки програмного забезпечення, особливо на етапах проектування та аналізу. Це допомагає розробникам програмного забезпечення визначити високорівневу архітектуру системи, створити детальний дизайн компонентів і виявити потенційні проблеми на ранніх стадіях розробки. UML також відіграє важливу роль у спілкуванні, забезпечуючи узгодженість усіх зацікавлених сторін і чіткість і точність документації системи.

Загалом, UML є важливою мовою для розробки програмного забезпечення, пропонуючи стандартизований метод проектування, візуалізації та документування систем. Використовуючи UML, розробники можуть переконатися, що всі аспекти системи добре зрозумілі, що призводить до більш ефективної розробки та кращої якості програмного забезпечення.

Інтелектуальний аналіз даних – це процес аналізу великих наборів даних для виявлення прихованих закономірностей, кореляцій і цінної інформації за допомогою передових методів статистики, машинного навчання та систем баз даних. Ця практика виходить за рамки простого аналізу даних, застосовуючи складні алгоритми для виявлення тенденцій, які можуть допомогти приймати більш обґрунтовані рішення, прогнозувати майбутні результати та вдосконалювати бізнес-стратегії.

За своєю суттю інтелектуальний аналіз даних передбачає отримання корисної інформації з величезних обсягів даних, які можуть бути неочевидними. Мета полягає в тому, щоб визначити значущі відносини, які можуть інформувати про стратегічні дії та бізнес-рішення. Його можна застосовувати як до структурованих даних (наприклад, баз даних), так і до неструктурованих даних (таких як текст, зображення або вміст соціальних мереж). Методи, які зазвичай використовуються в інтелектуальному аналізі даних, включають класифікацію, кластеризацію, регресію, аналіз правил асоціації, виявлення аномалій і послідовний аналіз шаблонів.

У класифікації дані класифікуються в попередньо визначені групи на основі спільних характеристик, які часто використовуються в таких програмах, як виявлення шахрайства, медичні діагнози або прогнозування поведінки клієнтів. З іншого боку, кластеризація групує схожі точки даних разом, допомагаючи компаніям сегментувати клієнтів або виявляти шаблони в даних, які раніше не були відомі. Регресійний аналіз використовується для прогнозування постійного значення на основі даних, що робить його корисним для прогнозування продажів, цін на акції чи інших тенденцій. Інтелектуальний аналіз правил асоціації визначає взаємозв’язки між різними змінними, наприклад виявлення того, які товари часто купують разом у налаштуваннях роздрібної торгівлі.

Виявлення аномалій зосереджено на виявленні незвичайних точок даних, які відхиляються від норми, що особливо важливо в сферах безпеки, таких як виявлення шахрайства або виявлення несправного обладнання в

промислових умовах. Послідовний аналіз шаблонів визначає шаблони, які виникають у певних послідовностях, які можна застосовувати в таких областях, як аналіз поведінки споживачів або шаблони веб-навігації.

Процес інтелектуального аналізу даних зазвичай складається з кількох етапів. Спочатку виконується очищення даних, щоб усунути будь-які невідповідності або помилки. Дані з різних джерел потім інтегруються, а відповідні набори даних вибираються для аналізу. Ці дані перетворюються у відповідні формати для майнінгу. Нарешті, результати оцінюються та інтерпретуються для отримання значущої інформації.

Інтелектуальний аналіз даних має широке застосування в різних галузях промисловості. У маркетингу компанії використовують його, щоб зрозуміти поведінку споживачів і створити персоналізовані маркетингові стратегії. У сфері охорони здоров’я це допомагає передбачити результати пацієнтів і оптимізувати плани лікування. Фінансові установи використовують інтелектуальний аналіз даних для оцінки кредитних ризиків, виявлення шахрайства та прогнозування ринкових тенденцій. Крім того, він підтримує оптимізацію процесів і прогнозне обслуговування на виробництві.

Незважаючи на свої переваги, інтелектуальний аналіз даних викликає серйозні проблеми щодо конфіденційності та безпеки, особливо при роботі з конфіденційною інформацією. Організаціям важливо дотримуватися етичних принципів і правил захисту даних, щоб забезпечити довіру та уникнути неправомірного використання даних[10].

Підсумовуючи, інтелектуальний аналіз даних є потужним інструментом, який допомагає організаціям отримати цінну інформацію з великих обсягів даних. Застосовуючи складні алгоритми, компанії та дослідники можуть виявляти тенденції, які керують прийняттям рішень, підвищують ефективність та стимулюють інновації. Однак, щоб забезпечити відповідальне використання даних, завжди слід враховувати етичні міркування та міркування конфіденційності.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це візуальне представлення, яке використовується в розробці програмного забезпечення для опису функціональних вимог системи з точки зору кінцевого користувача. Він є частиною Уніфікованої мови моделювання (UML) і в основному використовується на ранніх етапах розробки програмного забезпечення, щоб проілюструвати, як система взаємодіє із зовнішніми об’єктами, відомими як «актори». Діаграма варіантів використання допомагає зацікавленим сторонам зрозуміти функціональність системи та взаємодію, яка відбувається між користувачами та системою.

На діаграмі варіантів використання система представлена прямокутником, а актори зображені у вигляді фігурок за межами системи. Актори можуть представляти різні типи користувачів, зовнішні системи або пристрої, які взаємодіють із системою. Кожен актор пов’язаний з одним або кількома варіантами використання, які представлені овалами. Ці варіанти

використання визначають конкретні дії або функції, які система виконуватиме на основі введення актора.

Рис. 3 Діаграма прецедентів

Основна мета діаграми варіантів використання полягає в тому, щоб охопити функціональні вимоги системи простим, високорівневим способом. Він показує, що буде робити система і як різні актори взаємодіятимуть з нею, але не вказує, як ці взаємодії будуть реалізовані. Замість цього він зосереджений на описі «що» функціональності системи, що робить його чудовим інструментом для розуміння вимог до системи та

передачі їх зацікавленим сторонам, включаючи розробників, тестувальників і бізнес-аналітиків.

Створена діаграма прецедентів містить акторів:

1. Відвідувач

Опис: відвідувач є основним користувачем системи, який відвідує веб-сайт для перегляду та пошуку проєктів.

Операції:

- фільтрування проєктів: відвідувач може фільтрувати проєкти за різними критеріями, такими як категорія, підкатегорія, тощо;
- пошук проєктів: можливість виконати пошук проєктів за ключовими словами або іншими параметрами;
- перегляд проєктів: можливість перегляду деталей проєктів, їхніх метрик та джерел.

2. Система

Опис: система є центральним елементом, який забезпечує збір, обробку, категоризацію та зберігання даних про проєкти.

Операції:

- збір даних про проєкти: Автоматизоване отримання даних про нові та існуючі проєкти;
- обробка даних про проєкти: аналіз та обробка отриманих даних для подальшого використання та аналізу;
- категоризація проєктів: визначення категорій та підкатегорій для класифікації проєктів;
- зберігання проєктів: забезпечення ефективного зберігання даних про проєкти для подальшого використання.

3. Адміністратор

Опис: адміністратор має повноваження керувати системою, моніторити діяльність та взаємодіяти з іншими користувачами.

Операції:

- авторизація: виконання процедури авторизації для доступу до адміністративних функцій;
- керування контентом: здійснення редагування, додавання або видалення інформації про проєкти та інших ресурсів системи;
- моніторинг: слідкування за діяльністю користувачів, забезпечення безпеки та вирішення потенційних проблем.

Ці актори та їхні операції створюють основу для подальшого розроблення функціональності та інтерфейсу вашої системи управління веб-проєктами.

Назва випадку використання: Пошук проекту

Актор: відвідувач

Мета: дозволити користувачеві шукати веб-проекти на основі конкретних критеріїв, таких як домен, технологія, галузь або статус проекту.

Передумови: Користувач повинен увійти в систему. Користувач має доступ до бази даних проекту з відповідними критеріями пошуку.

Основний потік:

1. Користувач входить у систему та переходить до розділу «Пошук проекту»;
2. Користувачеві пропонуються різні фільтри або параметри пошуку;
3. Після заповнення потрібних фільтрів користувач натискає кнопку «Пошук», щоб відправити пошуковий запит;
4. Система обробляє запит, запитуючи свою базу даних про проекти, які відповідають вибраним фільтрам. Система також може ранжувати або сортувати проекти на основі

релевантності, причому проекти з більшою відповідністю відображаються вище в списку;

5. Користувачеві пропонується список веб-проектів, які відповідають критеріям пошуку. Кожен проект у списку відображається з ключовими деталями;
6. Користувач може натиснути будь-який проект, щоб переглянути більш детальну інформацію;
7. Користувач може уточнити пошук або застосувати додаткові фільтри, щоб звузити результати;
8. Результати пошуку відображаються, і користувач може взаємодіяти з інформацією, або продовжуючи подальші дії (наприклад, збереження цікавих проектів, зв’язуючись із власниками проектів тощо), або починаючи новий пошук.

Альтернативні потоки:

1. Якщо користувач виконує порожній пошуковий запит або вводить недійсні дані (наприклад, непідтримувані символи в полі пошуку), система відобразить повідомлення про помилку, запропонувавши користувачеві скоригувати введені дані;
2. Якщо жоден проект не відповідає критеріям пошуку, система відобразить повідомлення на зразок «За вашими критеріями не знайдено проектів. Уточніть пошук».

Назва випадку використання: Категоризація проекту

Актор: Модуль збору та обробки даних

Мета: автоматично класифікувати веб-проекти на основі попередньо визначених критеріїв, таких як домен, стек технологій, розмір

проекту або статус, і призначити їх відповідним категоріям для легшого пошуку та керування.

Передумови:

Система повинна мати доступ до сховища веб-проектів із відповідними метаданими (наприклад, домен, технологія, опис тощо). Правила або алгоритми категоризації повинні бути заздалегідь визначені та інтегровані в систему (наприклад, правила для категоризації на основі ключових слів проекту, технологічного стеку або тегів, визначених користувачем). Модуль збору та обробки даних належним чином налаштований і активний у системі.

Основний потік:

1. Модуль збору й обробки даних запускає процес категоризації, зазвичай викликаний додаванням нових даних до системи (наприклад, нещодавно зареєстрованих проектів або оновлень існуючих даних проекту);
2. Модуль отримує всі необхідні дані проекту з бази даних системи або вхідного каналу даних. Це включає важливу інформацію, таку як опис проекту, теги, стек технологій, ключові слова та будь-які інші відповідні метадані;
3. Система застосовує заздалегідь визначені правила категоризації або алгоритми до кожного проекту;
4. На основі правил категоризації система автоматично відносить кожен проект до однієї або кількох категорій. Модуль позначатиме проект відповідними категоріями на основі критеріїв, знайдених у даних, і ці теги зберігаються в записі бази даних проекту;
5. Залежно від конфігурації системи може існувати додатковий етап перевірки, під час якого категоризовані проекти переглядаються (або автоматично, або користувачем/адміністратором). Цей крок гарантує, що

категоризація є точною та відповідає визначеним правилам. У разі виявлення розбіжностей система може змінити категорію проектів або позначити їх для перевірки вручну;

6. Після завершення процесу категоризації категоризовані проекти зберігаються назад у базу даних із оновленими тегами категорій. Це дозволяє системі легше отримувати класифіковані дані під час майбутніх пошуків або аналізу;
7. Класифіковані проекти тепер доступні в системі, що дозволяє користувачам (наприклад, інвесторам, адміністраторам) швидко шукати, фільтрувати та отримувати проекти на основі їх категорії. Це підвищує ефективність управління проектами та покращує взаємодію з користувачем.

Альтернативні потоки:

1. Якщо система виявить неповні або недійсні дані (наприклад, відсутність інформації про технологічний стек або домен проекту), модуль категоризації може позначити проект для перевірки або спробувати класифікувати його на основі доступних даних. Крім того, проект може бути виключений з процесу категоризації, доки не буде надана необхідна інформація.
2. Якщо система виявить проект, який не відповідає жодній із попередньо визначених категорій (наприклад, абсолютно нова технологія чи домен), система може створити нову тимчасову категорію та позначити проект для подальшого перегляду вручну. З часом нові категорії можуть бути додані до системи за потреби.

2.2.2 Діаграма послідовності. Діаграма послідовності — це тип діаграми взаємодії UML, яка візуально представляє, як різні системні

компоненти або об’єкти взаємодіють з часом. Це особливо корисно для моделювання поведінки системи, зосереджуючись на порядку, в якому відбуваються події. Діаграми послідовності показують потік повідомлень між акторами та об’єктами в системі, забезпечуючи ясність того, як взаємодіють різні частини системи.

На діаграмі послідовності актори представляють зовнішні сутності, такі як користувачі або інші системи, які взаємодіють із системою. Ці актори зазвичай відображаються у верхній частині діаграми. Об’єкти або компоненти — це різні частини системи, залучені до взаємодії, зображені вертикальними пунктирними лініями, які називаються лініями життя. На діаграмі також показано повідомлення між цими компонентами, стрілки вказують напрямок і тип зв’язку. Ці повідомлення можуть бути синхронними, коли відправник чекає на відповідь, або асинхронними, коли відправник продовжує роботу, не чекаючи відповіді. Зворотні повідомлення, які показані пунктирними стрілками, вказують на те, що керування або дані повертаються відправнику після виконання процесу[9].

Кожен учасник діаграми послідовності представлений лінією життя, яка показує тривалість участі об'єкта у взаємодії. Коли об’єкт виконує завдання, він позначається полем активації, прямокутником, розміщеним на лінії життя, щоб вказати період активності. Умови або цикли також можуть бути представлені на діаграмі послідовності, щоб проілюструвати такі сценарії, як умовне розгалуження або повторювані дії. Крім того, знищення об’єктів можна зобразити, поставивши «X» на кінці лінії життя, вказуючи, що об’єкт більше не використовується після послідовності.

Діаграми послідовності зазвичай використовуються для пояснення взаємодії в системі, документуючи, як компоненти обмінюються інформацією в певному порядку. Вони також відіграють важливу роль у

проектуванні поведінки системи, оскільки дозволяють командам візуалізувати потенційну неефективність або проблеми перед впровадженням. Крім того, ці діаграми служать цінним інструментом для створення тестових випадків на основі детального потоку взаємодій, які вони представляють.

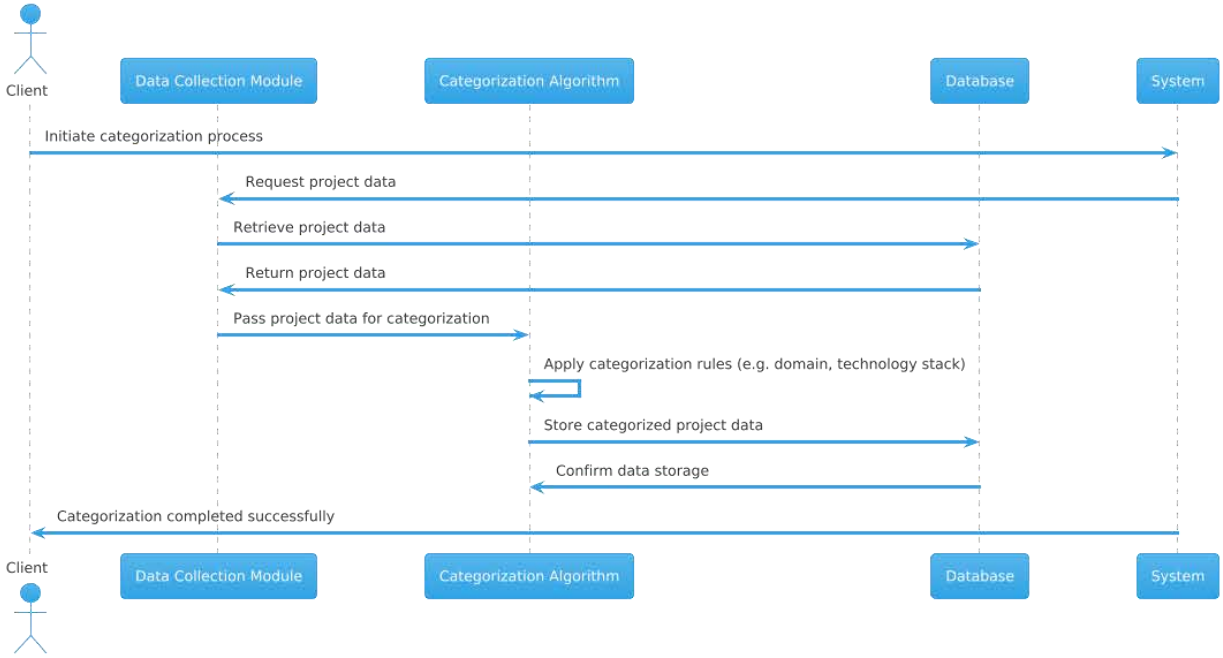


Рис. 4 Діаграма послідовності

Користувач починає процес, запускаючи функцію «Категоризація проекту» в системі. Система запитує необхідні дані проекту з модуля збору даних. Модуль збору даних отримує дані проекту з бази даних. Після збору даних модуль збору даних передає інформацію в алгоритм категоризації. Алгоритм категоризації застосовує попередньо визначені правила категоризації (наприклад, на основі домену, стеку технологій тощо). Після категоризації алгоритм категоризації зберігає категоризовані дані проекту назад у базу даних. Система сповіщає Користувача про успішне завершення процесу категоризації.

2.2.3 Діаграма активності. Діаграма діяльності — це діаграма UML (уніфікована мова моделювання), яка використовується для представлення потоку керування або даних у системі чи процесі. Це особливо цінно для візуалізації динамічних аспектів системи, зображення

послідовності дій, рішень і взаємодій, залучених у процес. Цей тип діаграми допомагає моделювати робочі процеси, бізнес-процеси та поведінку складної системи, пропонуючи чітке уявлення про те, як виконуються завдання та як вони взаємодіють у системі.

Основні компоненти діаграми діяльності включають дії, які представлені у вигляді округлених прямокутників і вказують на дії або завдання в рамках процесу. Діаграма починається з початкового вузла, позначеного зафарбованим колом, який позначає початок робочого процесу. Подібним чином він закінчується кінцевим вузлом, зображеним у вигляді кола з рамкою, що сигналізує про завершення процесу. Діяльність поєднується стрілками, відомими як переходи, які показують напрямок потоку керування.

Рішення, представлені ромбоподібними формами, приймаються, коли вибір робиться на основі певних умов або логіки. І навпаки, злиття, також показано як ромб, поєднує кілька шляхів назад в один. Розгалуження, що відображаються у вигляді товстих горизонтальних або вертикальних смуг, розділяють процес на паралельні дії, тоді як з’єднання, також представлені товстими смугами, синхронізують паралельні шляхи назад у єдиний потік.

Плавні доріжки використовуються для організації діяльності на основі того, хто або що за них відповідає, групуючи їх в окремі секції для різних учасників або компонентів. Це допомагає прояснити ролі, які беруть участь у процесі, полегшуючи розуміння схеми.

Діаграми діяльності особливо корисні для моделювання робочих процесів, де вони зображують кроки, що беруть участь у процесі, наприклад між різними відділами або компонентами системи. Вони також служать інструментом для моделювання бізнес-процесів, дозволяючи візуалізувати

те, як виконуються завдання та як інформація переміщується системою. Крім того, діаграми активності можуть моделювати поведінку системи, ілюструючи потік від одного стану до іншого, і вони часто використовуються для опису алгоритмів, показуючи послідовність операцій у функції чи завданні[9].

Переваги використання діаграм діяльності включають забезпечення чіткого та наочного представлення складних робочих процесів, визначення неефективності або вузьких місць і покращення спілкування між членами команди та зацікавленими сторонами. Ці діаграми також служать цінною документацією для навчання, розробки та обслуговування системи, забезпечуючи спільне розуміння того, як працює процес.

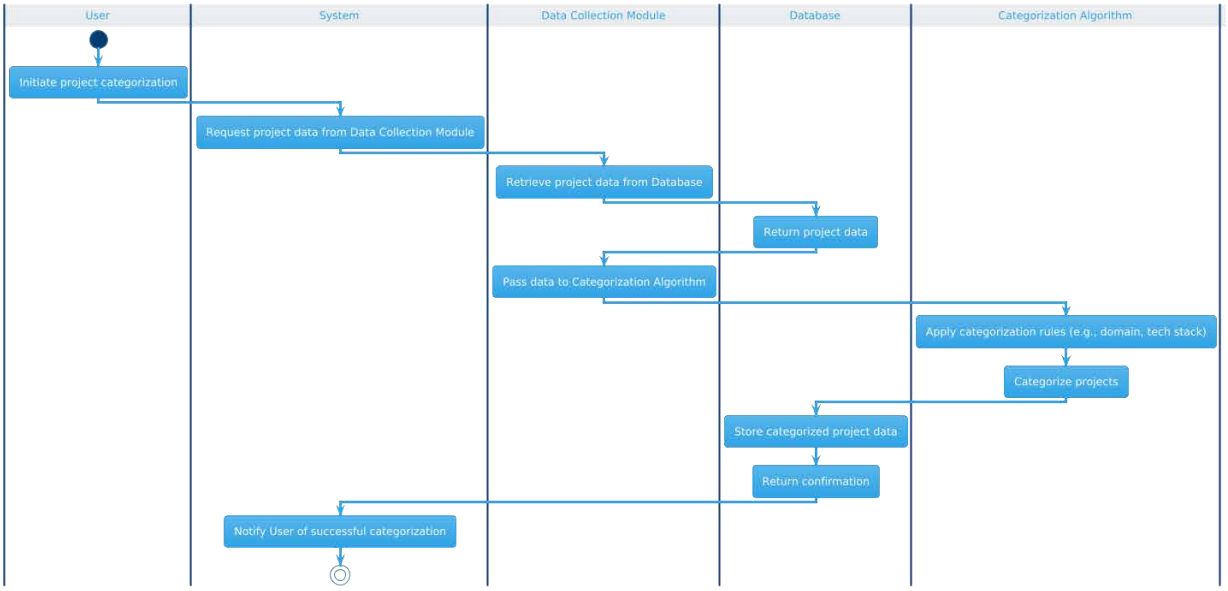


Рис. 5 Діаграма активності

Користувач починає процес, ініціюючи категоризацію веб-проектів. Система запитує необхідні дані проекту від модуля збору даних. Модуль збору даних отримує дані проекту з бази даних. База даних повертає запитані дані проекту до модуля збору даних. Потім модуль збору даних передає дані проекту в алгоритм категоризації. Алгоритм категоризації застосовує правила для категоризації, як-от визначення домену (наприклад, електронна комерція,

медицина) або використовуваний стек технологій. Після категоризації проектів алгоритм категоризації зберігає категоризовані дані назад у базу даних. База даних підтверджує успішне зберігання категоризованих даних проекту. Система сповіщає Користувача про успішне завершення процесу категоризації.

2.3 Огляд інструментів для реалізації завдань Data Mining

Інтелектуальний аналіз даних відіграє вирішальну роль в інтелектуальній системі обліку веб-проектів, оскільки передбачає вилучення корисних шаблонів і інформації з великих наборів даних. Для ефективного виконання завдань інтелектуального аналізу даних доступні різноманітні інструменти та фреймворки, кожен із яких пропонує окремі функції для збору, аналізу та обробки даних. Нижче наведено огляд деяких відомих інструментів, які можна використовувати для аналізу даних у контексті обліку веб-проектів.

RapidMiner — це потужна наукова платформа з відкритим вихідним кодом, розроблена для завдань інтелектуального аналізу даних і машинного навчання. Він підтримує широкий спектр методів аналізу даних, попередньої обробки та візуалізації. Платформа надає зручний графічний інтерфейс, який дозволяє користувачам створювати, оцінювати та розгортати моделі без глибоких знань програмування. Велика бібліотека попередньо створених операторів RapidMiner спрощує процес інтелектуального аналізу даних, що робить його ідеальним для таких завдань, як кластеризація, класифікація, регресія та виявлення аномалій у даних веб-проекту.

KNIME (Konstanz Information Miner) — ще одна популярна платформа з відкритим кодом для аналізу даних, звітності та інтеграції. Він підтримує робочі процеси інтелектуального аналізу даних і пропонує

модульну конструкцію, що дозволяє користувачам створювати власні робочі процеси шляхом підключення різних компонентів. Можливості інтелектуального аналізу даних KNIME включають інструменти для попередньої обробки даних, перетворення, кластеризації та аналізу правил асоціації, які можна застосовувати для аналізу веб-проектів і отримання цінної інформації зі структурованих і неструктурованих джерел даних. Його підтримка широкого спектру джерел даних, включаючи бази даних, електронні таблиці та веб-сервіси, робить його придатним для комплексного аналізу.

Weka (Waikato Environment for Knowledge Analysis) — це інструмент із відкритим вихідним кодом для завдань інтелектуального аналізу даних і машинного навчання. Він надає набір алгоритмів для класифікації, кластеризації, регресії та аналізу правил асоціації, що робить його корисним для аналізу даних веб-проекту для виявлення закономірностей, тенденцій і кореляцій. Зручний інтерфейс Weka дозволяє легко маніпулювати даними та візуалізувати їх, що робить його хорошим вибором як для початківців, так і для досвідчених користувачів. Його здатність обробляти як структуровані, так і неструктуровані дані робить його універсальним для ряду завдань інтелектуального аналізу даних.

Apache Mahout — це бібліотека машинного навчання з відкритим вихідним кодом, призначена для масштабованого аналізу даних і завдань машинного навчання. Побудований на основі фреймворку Apache Hadoop, Mahout забезпечує розподілену обробку, що робить його чудовим інструментом для обробки великомасштабних даних, типових для систем керування веб-проектами. Він підтримує алгоритми для класифікації, кластеризації, спільної фільтрації та рекомендацій, які можна

використовувати для визначення тенденцій у даних веб-проектів, поведінки користувачів або класифікації веб-проектів на основі певних критеріїв.

Незважаючи на те, що TensorFlow відомий перш за все як додатки глибокого навчання, він також є потужним інструментом для завдань інтелектуального аналізу даних, які вимагають передових моделей машинного навчання. Гнучкість TensorFlow дозволяє розробляти власні моделі для кластеризації, виявлення аномалій і прогнозної аналітики. Він підтримує високорівневі абстракції для тренувальних моделей на великих наборах даних, що може бути корисним для складних завдань інтелектуального аналізу даних, пов’язаних з аналізом веб-проектів, таких як категоризація проектів, аналіз настроїв або прогнозування тенденцій.

SAS Enterprise Miner — це комплексне програмне забезпечення для аналізу даних, розроблене для бізнес-аналітики та прогнозної аналітики. Він надає широкий спектр інструментів для дослідження даних, створення моделей і розгортання, що робить його потужним інструментом для організацій, яким потрібні рішення для аналізу даних на рівні підприємства. Завдяки підтримці розширених аналітичних методів, таких як дерева рішень, нейронні мережі та регресійний аналіз, SAS Enterprise Miner можна використовувати для аналізу даних веб-проектів на предмет тенденцій, ризиків і можливостей, надаючи зацікавленим сторонам практичну інформацію[11].

Orange — це простий і потужний набір інструментів для аналізу даних і машинного навчання з відкритим кодом. Він має простий у використанні графічний інтерфейс для створення робочих процесів інтелектуального аналізу даних, що робить його чудовим варіантом для користувачів, які віддають перевагу візуальним інструментам для дослідження даних. Orange підтримує різні завдання, включаючи класифікацію, кластеризацію, регресію та аналіз правил асоціації, і здатний

обробляти великі набори даних, які зазвичай зустрічаються в системах керування веб-проектами. Його інтеграція з Python також дозволяє розширені сценарії та налаштування моделі.

R — це мова програмування, яка широко використовується для статистичних обчислень і аналізу даних, із великою екосистемою пакетів для завдань інтелектуального аналізу даних. RStudio, інтегроване середовище розробки (IDE) для R, спрощує процес кодування та візуалізації даних. R надає такі пакети, як caret, randomForest і e1071, які підтримують класифікацію, регресію та кластеризацію, серед інших методів аналізу даних.

2.4 Структура джерела інформації для інтелектуального аналізу

Інтелектуальна система обліку веб-проектів базується на добре структурованому підході до обробки, зберігання та аналізу даних веб-проектів. Мета полягає в тому, щоб ефективно керувати даними та використовувати їх для прийняття рішень, категоризації проектів і аналізу ефективності. Ця система включає в себе кілька рівнів, кожен з яких сприяє загальній функціональності та ефективності.

В основі системи лежить рівень збору даних, який відповідає за збір важливої інформації, пов’язаної з проектом. Це включає основні метадані проекту, такі як назва проекту, опис, використані технології та пов’язані посилання, а також дані контролю версій з таких платформ, як GitHub і GitLab. У цей рівень інтегровано додаткові джерела даних, включаючи зовнішні API, відгуки користувачів і аналітику трафіку. Консолідуючи різноманітні джерела даних, система гарантує повне уявлення про проекти.

Після збору дані піддаються трансформації на рівні попередньої обробки даних. Цей етап має вирішальне значення для очищення даних

шляхом усунення дублікатів, виправлення невідповідностей і заповнення пропущених значень. Потім дані нормалізуються, щоб забезпечити однаковість за різними атрибутами. Процес виділення ознак визначає найбільш відповідні точки даних, які необхідні для ефективного аналізу. Рівень попередньої обробки також передбачає інтеграцію даних, гарантуючи, що інформація з багатьох джерел вирівнюється та узгоджена для подальшої обробки.

Після попередньої обробки рівень зберігання даних відіграє ключову роль в організації та збереженні оброблених даних. Він використовує реляційні бази даних для структурованої інформації про проект і бази даних NoSQL для неструктурованих даних, таких як журнали та відгуки користувачів. Хмарні рішення для зберігання даних можна використовувати для масштабування системи в міру зростання обсягу даних, забезпечуючи високу доступність і продуктивність.

Після збереження система переходить на рівень аналізу даних, де розблоковується справжня вартість зібраних даних. За допомогою методів інтелектуального аналізу даних, таких як кластеризація, класифікація та аналіз правил асоціації, система визначає шаблони та тенденції в даних. Інструменти прогнозної аналітики використовують моделі машинного навчання для прогнозування майбутніх тенденцій або потенційних результатів проекту. Техніки інтелектуального аналізу тексту та обробки природної мови (NLP) застосовуються для отримання інформації з неструктурованих даних, таких як описи проектів і відгуки користувачів. Цей аналіз допомагає класифікувати проекти, оцінювати продуктивність і виявляти аномалії, які можуть вказувати на нові проблеми[12].

Відомості, зібрані в результаті аналізу даних, потім упорядковуються та представляють на рівні представлення знань. Цей рівень структурує дані таким чином, щоб користувачам було легко їх

інтерпретувати та діяти. Онтології та таксономії класифікують веб-проекти на основі таких атрибутів, як стек технологій або галузь. Графіки знань відображають зв’язки між різними об’єктами, такими як проекти, учасники та технології, пропонуючи візуальне розуміння того, як різні елементи взаємопов’язані. Інформаційні панелі та звіти відображають ключові показники та тенденції, що дозволяє зацікавленим сторонам приймати обґрунтовані рішення.

На рівні підтримки прийняття рішень система проводить аналіз і перетворює його на дієві рекомендації. Цей рівень допомагає користувачам визначати області для вдосконалення, пропонує оптимальні технології чи стратегії та попереджає їх про потенційні ризики. Він генерує показники ефективності для відстеження ефективності проектів і використовує алгоритми оптимізації для підвищення ефективності розробки та управління проектами.

Щоб зробити ці функції доступними, рівень інтерфейсу користувача представляє функції системи в зручній для користувача формі. Завдяки можливостям пошуку та фільтрації користувачі можуть швидко знаходити проекти за різними критеріями, такими як галузь чи технологічний стек. Інформаційні панелі надають огляд ключових показників ефективності (KPI), тенденцій і статусів проектів, а інтерактивні візуалізації допомагають користувачам зрозуміти складні дані. Система також пропонує сповіщення та повідомлення, гарантуючи, що користувачі залишаються в курсі значних змін у проекті або нових ризиків.

Безпека має першочергове значення, а рівень безпеки та конфіденційності забезпечує цілісність конфіденційних даних. Механізми автентифікації та авторизації контролюють доступ до системи, гарантуючи, що лише авторизовані користувачі можуть отримати доступ до певних даних або виконувати певні дії. Методи шифрування захищають конфіденційність

конфіденційної інформації, а засоби контролю доступу обмежують можливості користувачів залежно від їхніх ролей у системі. Крім того, система дотримується правил конфіденційності, забезпечуючи дотримання таких законів, як GDPR.

Щоб задовольнити різноманітні вимоги до даних, у сховищі даних було ретельно розроблено набір таблиць. Сховище даних зображено на Рис. 6.

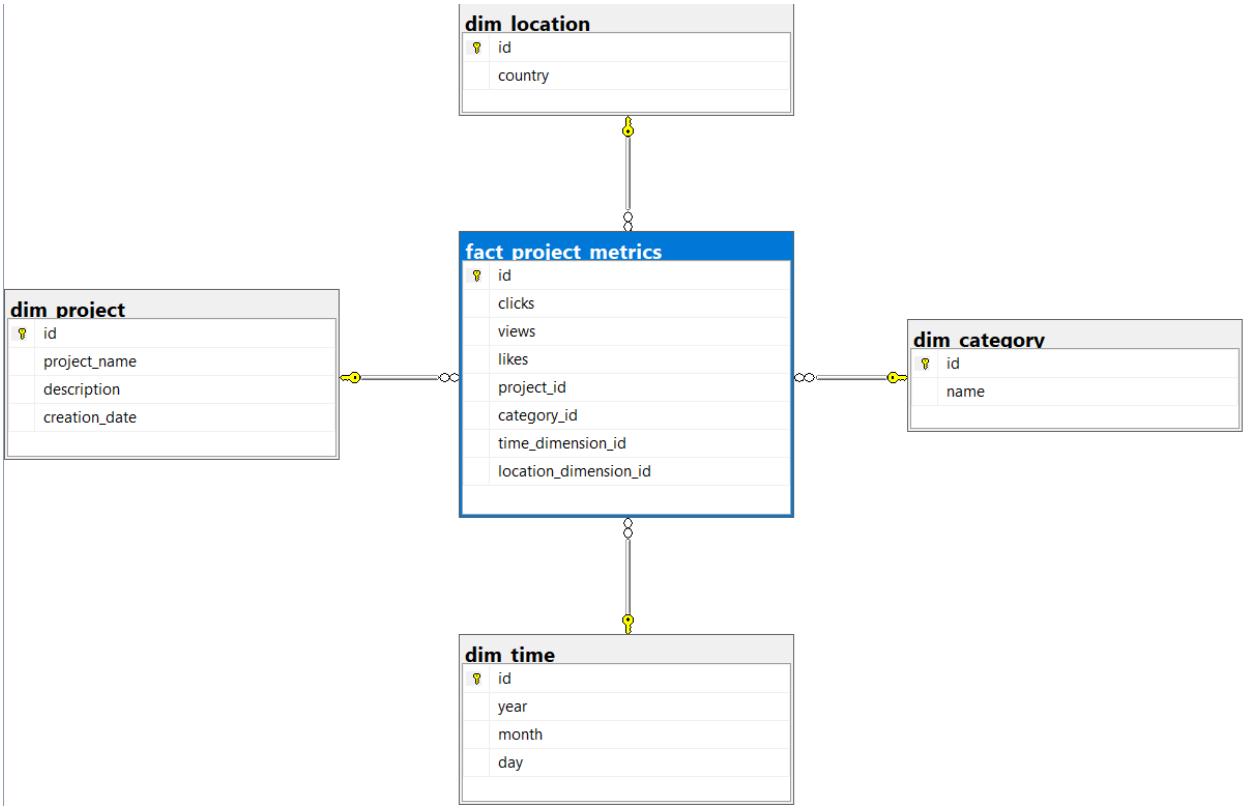


Рис. 6 Сховище даних системи

Сховище даних (data warehouse) — це тип системи керування даними, яка призначена для забезпечення та підтримки діяльності бізнес-аналітики (BI). Сховища даних призначені виключно для виконання запитів та аналізу і часто містять великі обсяги історичних даних. Дані в сховищі даних зазвичай отримують з широкого діапазону джерел, таких як файли журналів програм і програми транзакцій.

3 РОЗРОБКА СИСТЕМИ

3.1 Логічна модель даних

ERwin — це потужний інструмент моделювання даних, який широко використовується для проектування та керування архітектурами даних, насамперед у сферах розробки баз даних, бізнес-аналітики та керування корпоративними даними. ERwin відомий своєю здатністю полегшувати створення, візуалізацію та керування складними моделями даних, дозволяючи організаціям створювати послідовні та ефективні структури даних, забезпечуючи при цьому відповідність моделей даних вимогам бізнесу.

ERwin надає зручний графічний інтерфейс, який дозволяє користувачам легко проектувати логічні та фізичні моделі даних. Він підтримує різноманітні методи моделювання, включаючи діаграми сутності та зв’язку (ERD), які використовуються для візуального представлення структури даних у базі даних. Це допомагає організувати дані в сутності, визначити зв’язки між цими сутностями та описати атрибути та обмеження.

Ключовою особливістю ERwin є його здатність генерувати та реконструювати схеми баз даних, що полегшує підтримку та оновлення моделей даних у міру розвитку основної бази даних. Він також підтримує інтеграцію даних, керування даними та керування метаданими, гарантуючи, що моделі даних не лише точні, але й узгоджені в різних системах.

ERwin дозволяє співпрацювати між кількома користувачами, дозволяючи командам працювати над одним проектом одночасно. Він підтримує контроль версій, який допомагає відстежувати зміни з часом і гарантує, що всі члени команди працюють з останньою версією моделі. Крім того, ERwin надає інструменти звітності та документації, які можуть

створювати вичерпні модельні звіти, корисні для відповідності, аудиту та обміну інформацією із зацікавленими сторонами[15].

Однією з найбільш помітних переваг ERwin є його підтримка як реляційних, так і нереляційних баз даних, що робить його універсальним для низки середовищ баз даних. Він підтримує такі популярні платформи баз даних, як Microsoft SQL Server, Oracle, MySQL, PostgreSQL та інші, що дозволяє компаніям адаптувати свої моделі даних до своїх конкретних потреб баз даних.

Загалом, ERwin — це комплексне рішення для моделювання та керування базою даних, яке допомагає організаціям підвищити узгодженість даних, оптимізувати продуктивність бази даних і оптимізувати процес розробки. Він відіграє вирішальну роль у забезпеченні відповідності архітектури даних бізнес-цілям і можливості ефективного масштабування в міру зростання обсягів даних.

Логічна модель системи представлена на рис. 7.

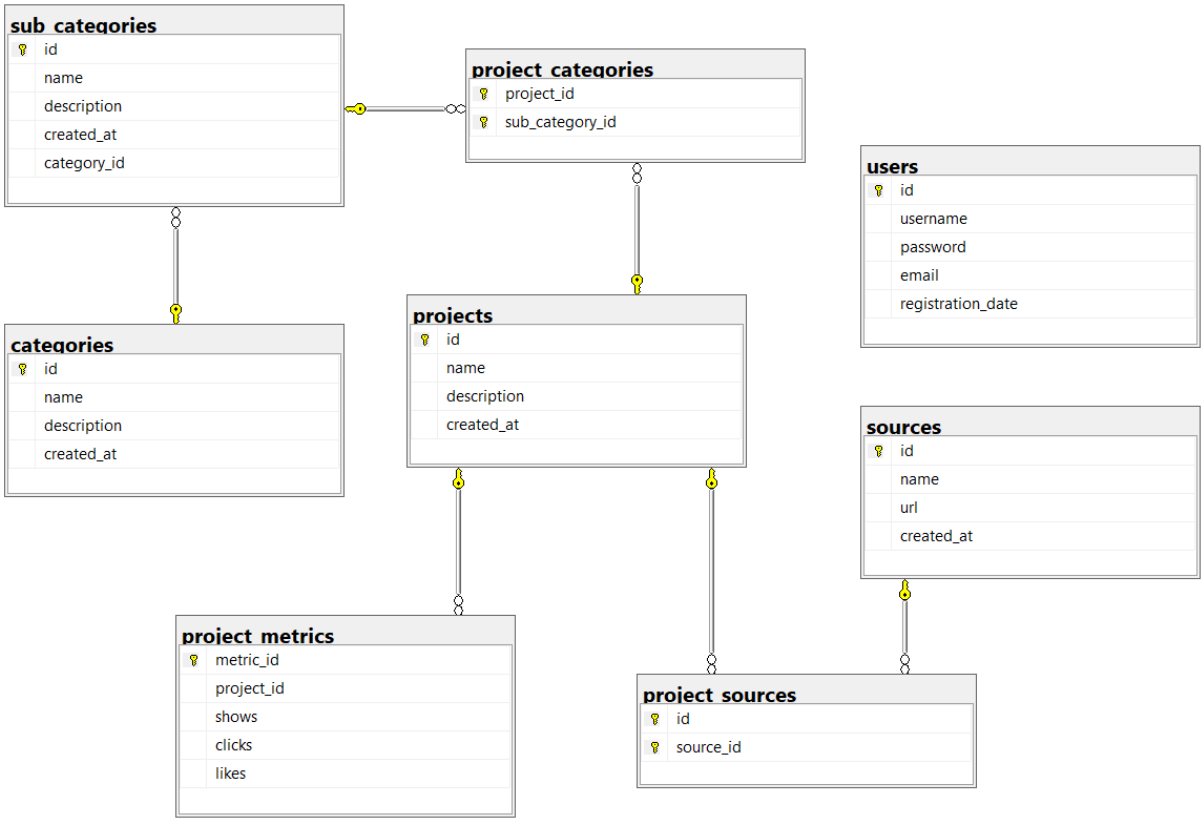


Рис. 7 ER-діаграма

На етапі розробки логічної моделі даних було виділено 8 сутностей:

1. Таблиця users:

   - id (INT): Унікальний ідентифікатор користувача;
   - username (NVARCHAR(255)): Ім'я користувача;
   - password (NVARCHAR(255)): Пароль користувача;
   - email (NVARCHAR(255)): Електронна пошта користувача;
   - registration_date (DATETIME): Дата реєстрації користувача (за замовчуванням - поточна дата).

2. Таблиця categories:

   - id (INT): Унікальний ідентифікатор категорії;
   - name (NVARCHAR(255)): Назва категорії;
   - description (NVARCHAR(MAX)): Опис категорії;
   - created_at (DATETIME): Дата створення категорії (за замовчуванням - поточна дата).

3. Таблиця sub_categories:

   - id (INT): Унікальний ідентифікатор підкатегорії;
   - name (NVARCHAR(255)): Назва підкатегорії;

- description (NVARCHAR(MAX)): Опис підкатегорії;
- created_at (DATETIME): Дата створення підкатегорії (за замовчуванням - поточна дата);
- category_id (INT): Зовнішній ключ, посилається на id в таблиці categories.

4. Таблиця projects:

- id (INT): Унікальний ідентифікатор проєкту;
- name (NVARCHAR(255)): Назва проєкту;
- description (NVARCHAR(MAX)): Опис проєкту;
- created_at (DATETIME): Дата створення проєкту (за замовчуванням - поточна дата).

5. Таблиця project_categories:

- project_id (INT): Зовнішній ключ, посилається на id в таблиці projects;
- sub_category_id (INT): Зовнішній ключ, посилається на id в таблиці sub_categories;
- Первинний ключ (PRIMARY KEY): Комбінація project_id та sub_category_id.

6. Таблиця sources:

- id (INT): Унікальний ідентифікатор джерела;
- name (NVARCHAR(255)): Назва джерела;
- url (NVARCHAR(MAX)): URL джерела;
- created_at (DATETIME): Дата створення джерела (за замовчуванням - поточна дата).

7. Таблиця project_sources:

- id (INT): Зовнішній ключ, посилається на id в таблиці projects;
- source_id (INT): Зовнішній ключ, посилається на id в таблиці sources;
- Первинний ключ (PRIMARY KEY): Комбінація id та source_id.

8. Таблиця project_metrics:

- id (INT): Унікальний ідентифікатор метрики проєкту;
- project_id (INT): Зовнішній ключ, посилається на id в таблиці projects;
- shows (INT): Кількість показів проєкту;
- clicks (INT): Кількість кліків на проєкт;
- likes (INT): Кількість лайків для проєкту.

Ця структура дозволяє вам організувати інформацію про користувачів, категорії, проєкти, джерела та метрики проєктів з використанням взаємозв'язків між таблицями.

3.2 Вибір системи управління базою даних та її реалізація

Система керування базами даних (СУБД) — це важливе програмне забезпечення, яке сприяє структурованому та ефективному зберіганню, організації та пошуку даних. Забезпечуючи інтерфейс для користувачів і програм, він абстрагує складність керування даними, дозволяючи легше взаємодіяти з основною інформацією. У середовищах, де необхідно зберігати, запитувати та оновлювати великі обсяги даних, СУБД відіграють вирішальну роль у забезпеченні безпеки, організації та доступності даних.

Основним призначенням СУБД є підтримка таких операцій, як створення, отримання, оновлення та видалення даних. Він досягає цього, пропонуючи структурований підхід до керування даними, який покращує продуктивність системи та забезпечує надійність і безпеку даних. Однією з основних функцій СУБД є її здатність визначати та запроваджувати структури даних за допомогою схем. Ці схеми допомагають забезпечити відповідність даних певним форматам і правилам, сприяючи їх цілісності та узгодженості[16].

На додаток до визначення даних, СУБД також дозволяє користувачам маніпулювати даними за допомогою мов запитів, таких як SQL, що полегшує вставлення, оновлення та отримання інформації. Контроль доступу є ще однією важливою особливістю СУБД, оскільки він регулює, хто може отримати доступ до даних і які дії вони можуть

виконувати. Це гарантує, що лише авторизовані користувачі можуть взаємодіяти з конфіденційною інформацією.

Управління кількома користувачами, які одночасно отримують доступ до системи, є ще одним ключовим аспектом СУБД. Він контролює взаємодію між користувачами, гарантуючи, що операції не конфліктують, а дані залишаються послідовними. Це особливо важливо в середовищах, де багато користувачів можуть оновлювати або запитувати дані одночасно. Крім того, СУБД забезпечує цілісність даних за допомогою таких обмежень, як первинні ключі та зовнішні ключі, гарантуючи, що зв'язки між елементами даних є дійсними, а дані не пошкоджені чи несумісні.

СУБД також надає функції резервного копіювання та відновлення, які необхідні для захисту даних у разі системних збоїв або катастроф. Це гарантує, що дані можуть бути відновлені до їх правильного стану, коли це необхідно, мінімізуючи ризик втрати даних. Ще однією ключовою перевагою СУБД є її здатність забезпечувати незалежність даних. Це означає, що користувачі можуть взаємодіяти з даними без необхідності розуміти деталі їх фізичного зберігання, що робить систему більш гнучкою та простішою в обслуговуванні.

Управління транзакціями є ще одним важливим аспектом функціональності СУБД. Транзакція в СУБД відноситься до набору операцій, які повинні бути виконані як єдине ціле. Система гарантує, що транзакції відповідають властивостям ACID (Atomicity, Consistency, Isolation, Durability), гарантуючи, що дані залишаються правильними та надійними навіть у разі збою.

СУБД бувають різних типів, кожна з яких підходить для різних потреб. Найбільш широко використовуються реляційні СУБД (RDBMS), які організовують дані в таблиці та використовують SQL для запитів. Такі

системи, як MySQL, Oracle і PostgreSQL, ідеально підходять для структурованих даних із чітко визначеними зв’язками. З іншого боку, СУБД NoSQL розроблені для обробки неструктурованих або напівструктурованих даних. Ці бази даних, включаючи MongoDB і Cassandra, пропонують більшу гнучкість і масштабованість, що робить їх придатними для великомасштабних програм з різними типами даних. Об’єктно-орієнтовані СУБД зберігають дані у формі об’єктів, подібно до об’єктно-орієнтованого програмування, і використовуються в додатках, де такі структури є корисними. Приклади включають ObjectDB і db4o. Ієрархічні та мережеві СУБД, хоч і менш поширені сьогодні, організовують дані в деревоподібних або графових структурах, дозволяючи різні види зв’язків між даними[17].

Зрештою, СУБД є фундаментальними для ефективного управління даними в сучасних обчислювальних середовищах. Вони зменшують надлишковість даних, підвищують продуктивність, забезпечують надійний захист і полегшують прийняття рішень на основі даних. Завдяки своїй здатності структурувати та підтримувати дані, СУБД є незамінними інструментами для компаній і розробників, які працюють з великими обсягами даних.

Для магістерської роботи «Інтелектуальна система обліку веб-проектів» я вибрав MySQL як систему керування базами даних (СУБД) завдяки численним перевагам, які добре відповідають вимогам проекту. MySQL — це надійна, масштабована та надійна система керування реляційною базою даних, яка широко використовується в веб-додатках, що робить її ідеальним вибором для цього проекту.

Перш за все, висока продуктивність MySQL є ключовим фактором при його виборі. MySQL відома своєю ефективністю в обробці великих наборів даних, що має вирішальне значення для системи обліку, яка буде керувати та аналізувати значні обсяги даних, наприклад записи веб-

проектів, їх метадані та взаємодії користувачів. Його здатність виконувати складні запити з низькою затримкою є головною перевагою, гарантуючи, що система може швидко отримувати та обробляти необхідну інформацію, навіть коли дані зростають.

Ще однією важливою причиною вибору MySQL є його потужна підтримка цілісності та безпеки даних. MySQL забезпечує застосування властивостей ACID (Atomicity, Consistency, Isolation, Durability), які є важливими для підтримки послідовності та надійності даних. В інтелектуальній системі обліку життєво важливо, щоб цілісність даних була забезпечена в будь-який час, особливо коли кілька користувачів взаємодіють із системою одночасно. MySQL надає різноманітні функції, такі як зовнішні ключі, обмеження та тригери, які допомагають підтримувати зв’язки та гарантувати достовірність даних, тим самим запобігаючи аномаліям і невідповідностям.

MySQL також пропонує чудову масштабованість. У міру розширення системи MySQL дозволяє легко обробляти збільшені навантаження та великі обсяги даних. Ця масштабованість має важливе значення для системи, якій з часом може знадобитися розмістити більше веб-проектів, категорій і користувачів. MySQL підтримує горизонтальне масштабування, що означає, що система може масштабуватися на кількох серверах, якщо це необхідно, таким чином зберігаючи високу продуктивність у міру зростання системи.

Крім того, MySQL дуже сумісний із веб-технологіями та фреймворками. З огляду на те, що інтелектуальна система обліку буде розгорнута як веб-додаток, бездоганна інтеграція MySQL з різними мовами програмування та фреймворками, включаючи PHP, Python і JavaScript, робить її відповідним вибором. Система може легко взаємодіяти з веб-

технологіями та фреймворками, які зазвичай використовуються у веб-розробці.

Крім того, MySQL є відкритим кодом і вільним для використання, що значно знижує вартість розробки та розгортання. Така природа відкритого вихідного коду означає, що існує велика спільнота розробників, які постійно роблять внесок у його вдосконалення та надають підтримку для усунення несправностей та оптимізації. Наявність багатої документації, навчальних посібників і форумів спільноти полегшує розробникам створення, усунення несправностей і оптимізацію системи, забезпечуючи довгострокову стійкість.

Важливими факторами при його виборі також є простота використання та простота керування базами даних MySQL. MySQL забезпечує зручний інтерфейс за допомогою таких інструментів, як phpMyAdmin і MySQL Workbench, які роблять завдання адміністрування бази даних, такі як резервне копіювання, міграція даних і оптимізація запитів, простими та доступними.

Підсумовуючи, MySQL було обрано для «Інтелектуальної системи обліку веб-проектів» завдяки її високій продуктивності, цілісності даних і функціям безпеки, масштабованості, сумісності з веб-технологіями, економічній ефективності та простоті використання. Ці функції ідеально відповідають вимогам проекту, гарантуючи, що система буде надійною, ефективною та здатною обробляти зростаючі обсяги даних з часом.

3.3 Архітектура програмного забезпечення

Архітектура програми «клієнт-сервер» — це широко використовувана конструкція, яка структурує взаємодію між клієнтом системи та компонентами сервера. У випадку «Інтелектуальної системи

обліку веб-проектів» ця архітектура обрана через її здатність ефективно розподіляти обов’язки, забезпечувати масштабованість і покращувати керування системою. Це дозволяє чітко розрізнити інтерфейс користувача та серверні процеси, сприяючи кращій продуктивності та зручності обслуговування.

Клієнтська сторона — це інтерфейс кінцевого користувача, де відбувається взаємодія. Він служить точкою доступу для взаємодії користувачів із системою та відповідає за відображення відповідних даних, таких як категорії веб-проектів, деталі проекту та результати пошуку. Створений з використанням таких веб-технологій, як HTML, CSS, JavaScript і Vue.js, клієнт дозволяє користувачам взаємодіяти з системою через браузер незалежно від їхнього фізичного розташування. Він виконує різноманітні завдання, включаючи представлення даних у зручному для користувача форматі, фіксацію введених користувачем даних, таких як критерії пошуку проекту та команди категоризації, і надсилання запитів на сервер.

На стороні сервера серверна частина відповідає за обробку запитів клієнтів, керування даними та виконання бізнес-логіки програми. Ця частина системи взаємодіє з базою даних MySQL для керування та зберігання даних проекту. Сервер отримує запити від клієнта, виконує необхідні операції, як-от оновлення, отримання або класифікацію даних проекту, і повертає відповіді. Для керування цими операціями сервер зазвичай використовує веб-фреймворк, наприклад Laravel для PHP або Node.js. Сервер забезпечує безперебійну роботу програми, перевіряючи дані, взаємодіючи з базою даних і надаючи клієнту відповідні відповіді.

Зв'язок між клієнтом і сервером відбувається в циклі запит-відповідь. Коли користувач взаємодіє з інтерфейсом клієнта, наприклад вводить пошуковий термін або класифікує проект, клієнт надсилає запит на сервер. Сервер обробляє запит, отримує необхідні дані з бази даних і

надсилає результат назад клієнту. Потім клієнт надає ці дані користувачеві, зберігаючи безперебійний інтерактивний досвід.

Система обліку веб-проектів використовує цю клієнт-серверну модель для вирішення ключових завдань, таких як пошук проектів, категоризація та керування. Наприклад, при пошуку проектів клієнт надсилає критерії пошуку на сервер. Сервер обробляє цей запит, запитує базу даних і повертає список проектів, які відповідають критеріям. Так само, коли класифікують проекти або оновлюють інформацію про проект, клієнт надсилає відповідні дані на сервер, який потім відповідно оновлює базу даних.

Ця клієнт-серверна архітектура забезпечує кілька переваг. По-перше, це покращує масштабованість, оскільки сервер можна масштабувати незалежно для обробки збільшеного трафіку або більш складних операцій. Клієнт залишається зосередженим на забезпеченні зручного для користувача інтерфейсу, а сервер виконує важку роботу. Поділ проблем також веде до кращої безпеки. Керування конфіденційними операціями та збереженням даних здійснюється на сервері, що гарантує безпеку особистої інформації користувачів і даних проекту. Зв’язок між клієнтом і сервером зашифрований (наприклад, через HTTPS), захищаючи дані під час передачі.

Крім того, ця архітектура підтримує модульність і ремонтопридатність. Розділивши клієнт і сервер, розробники можуть працювати над кожною частиною незалежно, що полегшує оновлення або покращення системи з часом. Нові функції можна додати до сервера або клієнта, не порушуючи роботу іншої частини системи[18].

Загалом клієнт-серверна архітектура ідеально підходить для «Інтелектуальної системи обліку веб-проектів», оскільки забезпечує чітке відокремлення взаємодії користувача та внутрішніх процесів, підтримує

масштабованість, підвищує безпеку та полегшує обслуговування та майбутні оновлення. Такий підхід гарантує, що система може розвиватися та адаптуватися до мінливих вимог, зберігаючи при цьому високий рівень продуктивності та надійності.

3.4 Використання 1-Rule для класифікації

Використання алгоритму 1-Rule[3] для класифікації передбачає ідентифікацію атрибута з найвищою точністю прогнозування та призначення найчастішої мітки класу екземплярам, які мають таке значення атрибута. Незважаючи на свою простоту, алгоритм 1-Rule служить базовим методом для оцінки більш складних моделей класифікації. Його простий підхід робить його особливо корисним для швидкої оцінки даних і початкового вивчення класифікаційних завдань. Однак його продуктивність може бути обмеженою, особливо на наборах даних із високою розмірністю або складними зв’язками між атрибутами та мітками класів. Незважаючи на це, алгоритм 1-Rule залишається цінним інструментом для розуміння основ класифікації та як еталон для порівняння з більш просунутими алгоритмами. Подальші дослідження продовжують вивчати вдосконалення та адаптацію

алгоритму 1-Rule для підвищення його ефективності в різних доменах і наборах даних.

Щоб проаналізувати свої дані за допомогою алгоритму 1-Rule, я створив програму в Microsoft Visual Studio та дослідив популярність веб-проектів за допомогою метрики «подобається».

Визначено залежну змінну: Клас - висока/низька популярність.

Якщо лайків >= 4949 - низька популярність

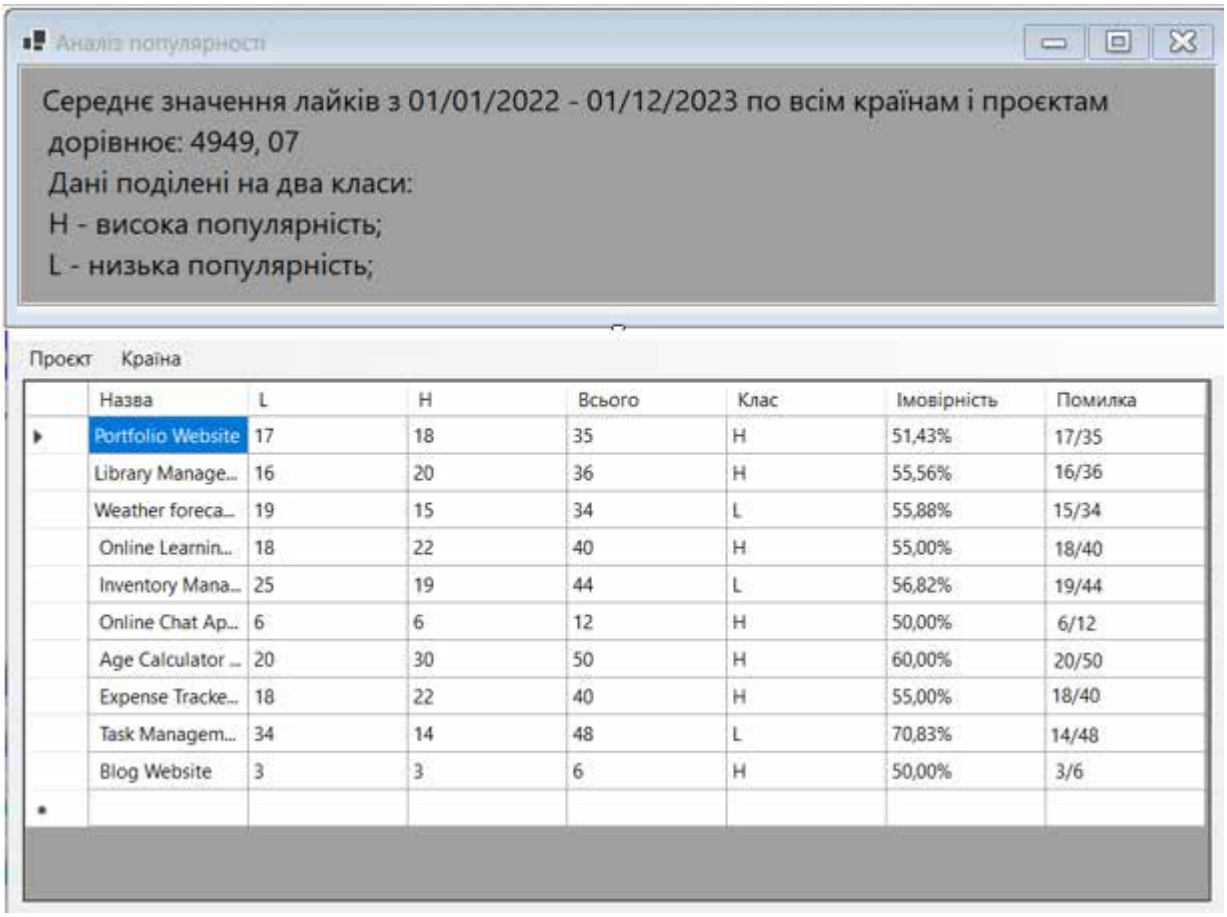
Якщо лайків < 4949 - висока популярність

Незалежні змінні: проект, країна.

На рисунку 8 показано результат роботи програми за назвою проекту.

Згідно з аналізом кількості лайків, найпопулярнішим проектом є додаток «Калькулятор віку» з високим рівнем популярності, який становить

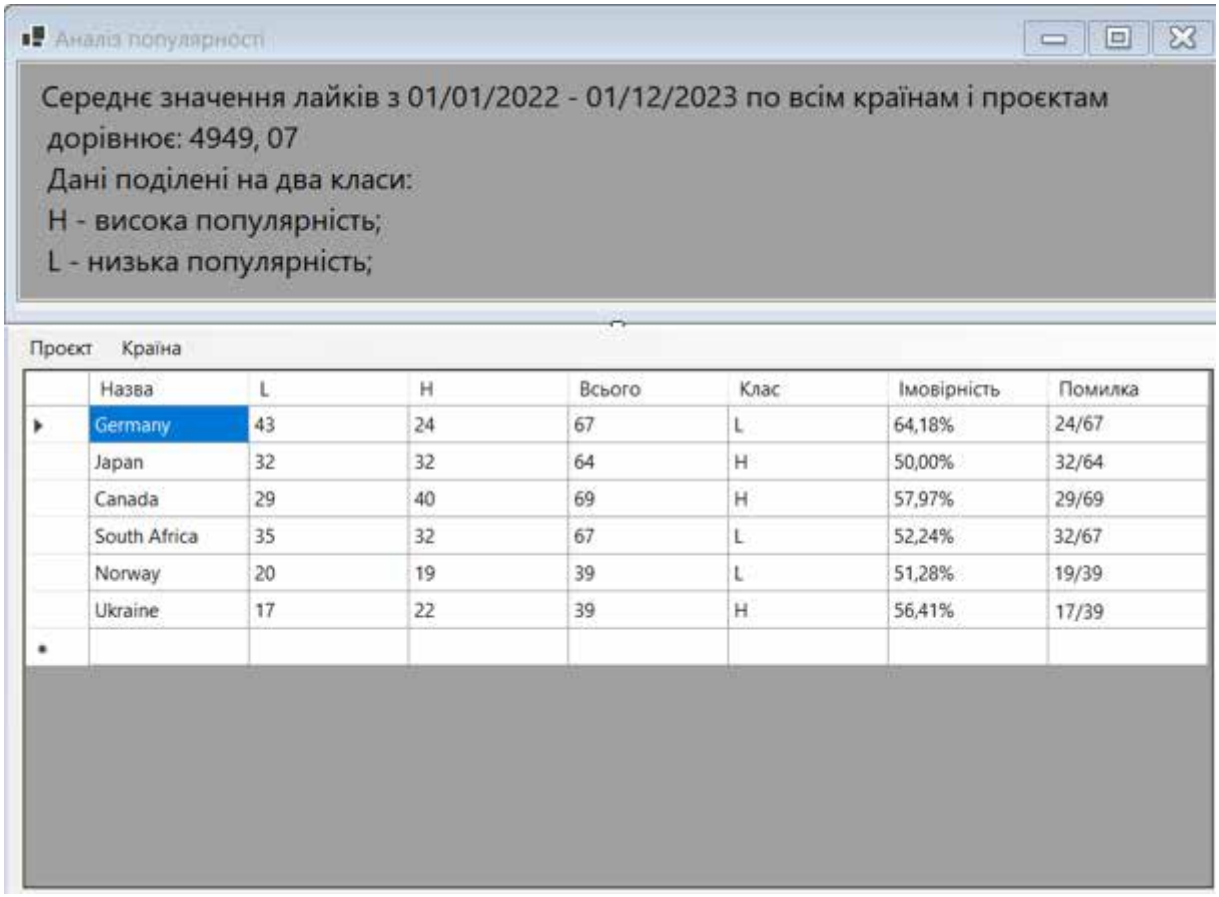
60%. Натомість найменш популярним проектом став Task Management System, який має низький рівень популярності, а саме 70,83%.



Назва	L	H	Всього	Клас	Імовірність	Помилка
Portfolio Website	17	18	35	H	51,43%	17/35
Library Manage...	16	20	36	H	55,56%	16/36
Weather foreca...	19	15	34	L	55,88%	15/34
Online Learnin...	18	22	40	H	55,00%	18/40
Inventory Mana...	25	19	44	L	56,82%	19/44
Online Chat Ap...	6	6	12	H	50,00%	6/12
Age Calculator ...	20	30	50	H	60,00%	20/50
Expense Tracke...	18	22	40	H	55,00%	18/40
Task Managem...	34	14	48	L	70,83%	14/48
Blog Website	3	3	6	H	50,00%	3/6

Рис. 8 Результат роботи програми за назвою проекту

На рисунку 9 показано результати програми за країнами.



Назва	L	H	Всього	Клас	Імовірність	Помилка
Germany	43	24	67	L	64,18%	24/67
Japan	32	32	64	H	50,00%	32/64
Canada	29	40	69	H	57,97%	29/69
South Africa	35	32	67	L	52,24%	32/67
Norway	20	19	39	L	51,28%	19/39
Ukraine	17	22	39	H	56,41%	17/39

Рис. 9 Результати програми по країнах

Згідно з аналізом даних по країнах, найбільше користувачам сподобався контент з Канади, оскільки він має високий рівень популярності, а саме 57,97%. З іншого боку, найменше лайків ставлять користувачі з Німеччини, яка має низький рівень популярності – 64,18%.

3.5 Вибір інструментарію для створення програмного забезпечення

Для розробки «Інтелектуальної системи обліку веб-проектів» було обрано ретельно відібраний набір інструментів, який забезпечує як ефективність, так і масштабованість протягом усього процесу. Набір інструментів включає ряд технологій, розроблених для обробки різних

аспектів програми, від внутрішньої розробки до зовнішньої реалізації та керування базами даних.

C# служить основною мовою програмування для внутрішньої розробки завдяки своїм надійним об’єктно-орієнтованим можливостям і розгалуженій екосистемі, що робить її ідеальною для створення масштабованих і підтримуваних програм. На доповнення до цього ASP.NET використовується як основа для створення веб-програми, пропонуючи потужну інфраструктуру для керування серверними операціями, такими як обробка даних, автентифікація користувачів і керування проектами.

Вибрана система керування базою даних MySQL, відома своєю високою продуктивністю, надійністю та масштабованістю. Як реляційна база даних із відкритим вихідним кодом, MySQL підтримує структуроване зберігання та керування даними веб-проекту. Його сумісність із C# і ASP.NET забезпечує безперебійну інтеграцію, що робить його підходящим вибором для керування великими наборами даних і виконання складних запитів.

Для взаємодії з базою даних використовується Entity Framework (EF) Core. Ця спрощена версія Microsoft ORM забезпечує ефективний засіб зіставлення моделей C# із таблицями MySQL, зменшуючи потребу в ручних запитах SQL. EF Core допомагає спростити керування даними, обробляючи операції CRUD і забезпечуючи плавну взаємодію між програмою та базою даних.

Щоб полегшити розробку, Visual Studio Code (VS Code) вибрано як інтегроване середовище розробки. Відомий своєю швидкістю, універсальністю та підтримкою різних мов програмування, VS Code дає змогу розробникам писати та налагоджувати код як для зовнішнього, так і для внутрішнього. Його широкий спектр розширень дозволяє налаштувати

його відповідно до конкретних потреб процесу розробки, забезпечуючи спрощений досвід розробки.

Для інтерфейсу HTML, CSS і JavaScript є основними технологіями, які використовуватимуться для створення адаптивного та інтерактивного інтерфейсу користувача. HTML формує структуру веб-сторінок, тоді як CSS обробляє стиль і макет, а JavaScript додає динамічну поведінку та інтерактивність, покращуючи взаємодію з користувачем.

Для розробки користувальницького інтерфейсу (UI) та взаємодії з користувачем (UX) обрано Figma, хмарний інструмент проектування. Це дозволяє дизайнерам і розробникам співпрацювати над візуалізацією та створенням прототипу інтерфейсу програми до того, як почнеться фактична розробка. З Figma дизайн можна повторювати, забезпечуючи відшліфований, інтуїтивно зрозумілий і зручний кінцевий продукт.

Інструменти, вибрані для цього проекту, забезпечують збалансований підхід до внутрішньої та зовнішньої розробки, керування базами даних і дизайну. C# і ASP.NET забезпечують міцну основу для створення масштабованих веб-додатків, тоді як MySQL забезпечує надійне зберігання даних. EF Core спрощує керування базами даних і підвищує продуктивність розробників. VS Code пропонує легке, настроюване середовище розробки, а HTML, CSS і JavaScript дозволяють створювати адаптивний і динамічний інтерфейс користувача. Нарешті, Figma забезпечує

безперебійну співпрацю в дизайні UI/UX, гарантуючи, що програма відповідає очікуванням дизайну до початку розробки.

Загалом цей набір інструментів гарантує комплексний та ефективний процес розробки, що веде до надійної та масштабованої системи для обліку веб-проектів.

3.6 Використання методу Наївного Байєса

Використання наївного методу Байєса[4] для класифікації передбачає застосування теореми Байєса з «наївним» припущенням незалежності ознак. Він обчислює ймовірність мітки класу з урахуванням особливостей екземпляра шляхом множення умовних ймовірностей кожної функції з міткою класу. Незважаючи на свою простоту та припущення про незалежність функцій, Naive Bayes часто добре працює на практиці, особливо для класифікації тексту та інших масивів даних великої розмірності. Він ефективний з точки зору обчислень і потребує невеликої кількості навчальних даних, що робить його придатним для додатків у реальному часі та великомасштабних наборів даних. Однак він може страждати від «проблеми нульової частоти», коли стикається з невидимими комбінаціями функцій під час прогнозування, і його продуктивність може погіршитися, якщо порушується припущення незалежності. Тим не менш, Naive Bayes залишається популярним вибором для завдань класифікації в різних областях, з реалізаціями, доступними в бібліотеках, як-от scikit-learn у Python.

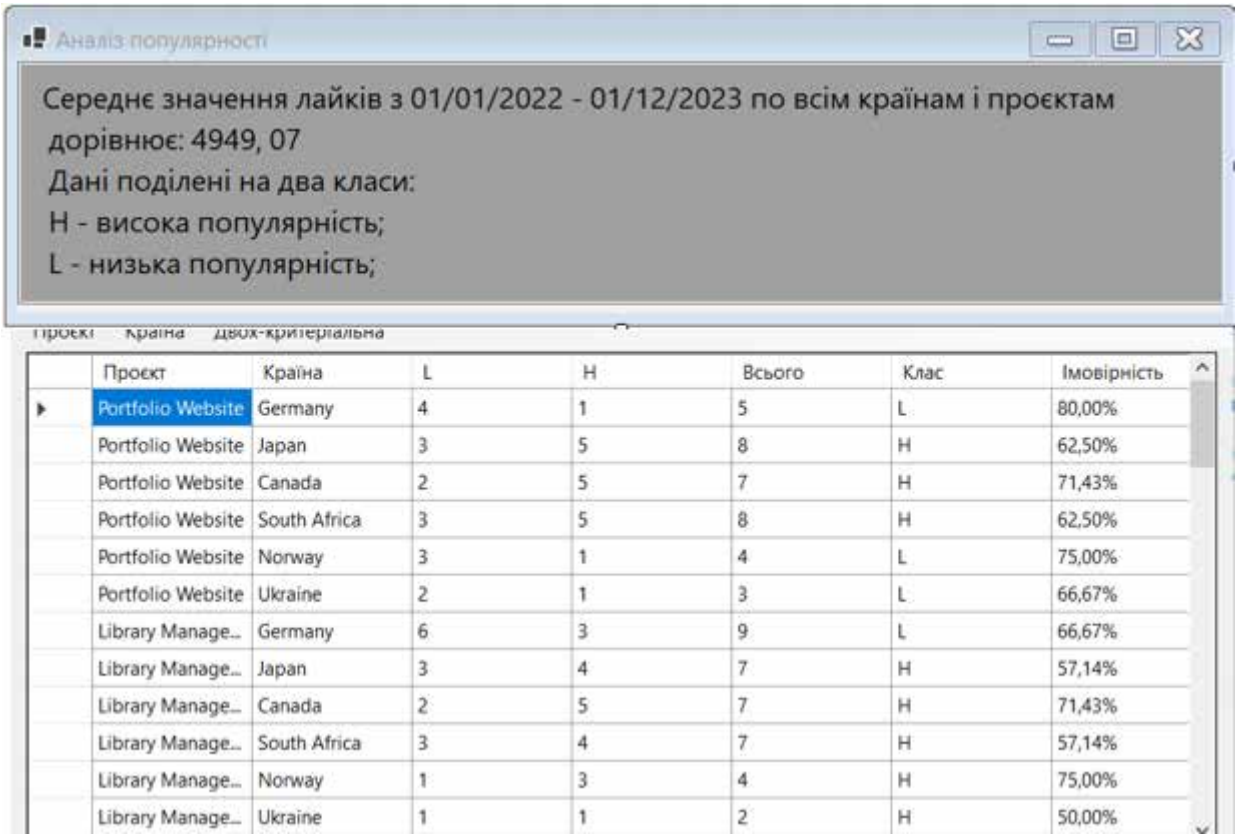
Алгоритм Naive Bayes був перевірений як у програмному забезпеченні, так і в середовищах SSAS (Mining Structure) для аналізу та

прогнозування переваг. Цей метод було програмно реалізовано в Microsoft Visual Studio на C#.

Результат роботи алгоритму, який показує зустріч двох незалежних змінних - Проект і Країна. Це показано на малюнку 10.

За результатами роботи алгоритму можна зробити висновок, що найпопулярнішими комбінаціями за кількістю лайків є такі: Online Learning Management System з України, Online Chat Application з України, Age Calculator Application з Норвегії та Blog Website з Японія/Канада/Південна

Африка. Це тому, що в цих випадках спостерігається високий рівень популярності, що досягає 100%.



Проєкт	Країна	L	H	Всього	Клас	Імовірність
Portfolio Website	Germany	4	1	5	L	80,00%
Portfolio Website	Japan	3	5	8	H	62,50%
Portfolio Website	Canada	2	5	7	H	71,43%
Portfolio Website	South Africa	3	5	8	H	62,50%
Portfolio Website	Norway	3	1	4	L	75,00%
Portfolio Website	Ukraine	2	1	3	L	66,67%
Library Manage...	Germany	6	3	9	L	66,67%
Library Manage...	Japan	3	4	7	H	57,14%
Library Manage...	Canada	2	5	7	H	71,43%
Library Manage...	South Africa	3	4	7	H	57,14%
Library Manage...	Norway	1	3	4	H	75,00%
Library Manage...	Ukraine	1	1	2	H	50,00%

Рис. 10 Результати зустрічі двох незалежних змінних – Проекту та Країни

Результати популярності Проекту та Країни наведено на рисунку 11.

Online Learnin...	Ukraine	0	2	2	H	100,00%
Online Chat Ap...	Ukraine	0	2	2	H	100,00%
Age Calculator ...	Norway	0	3	3	H	100,00%
Blog Website	Japan	0	1	1	H	100,00%
Blog Website	Canada	0	1	1	H	100,00%
Blog Website	South Africa	0	1	1	H	100,00%

Рис. 11 Результати популярності Проекту та Країни

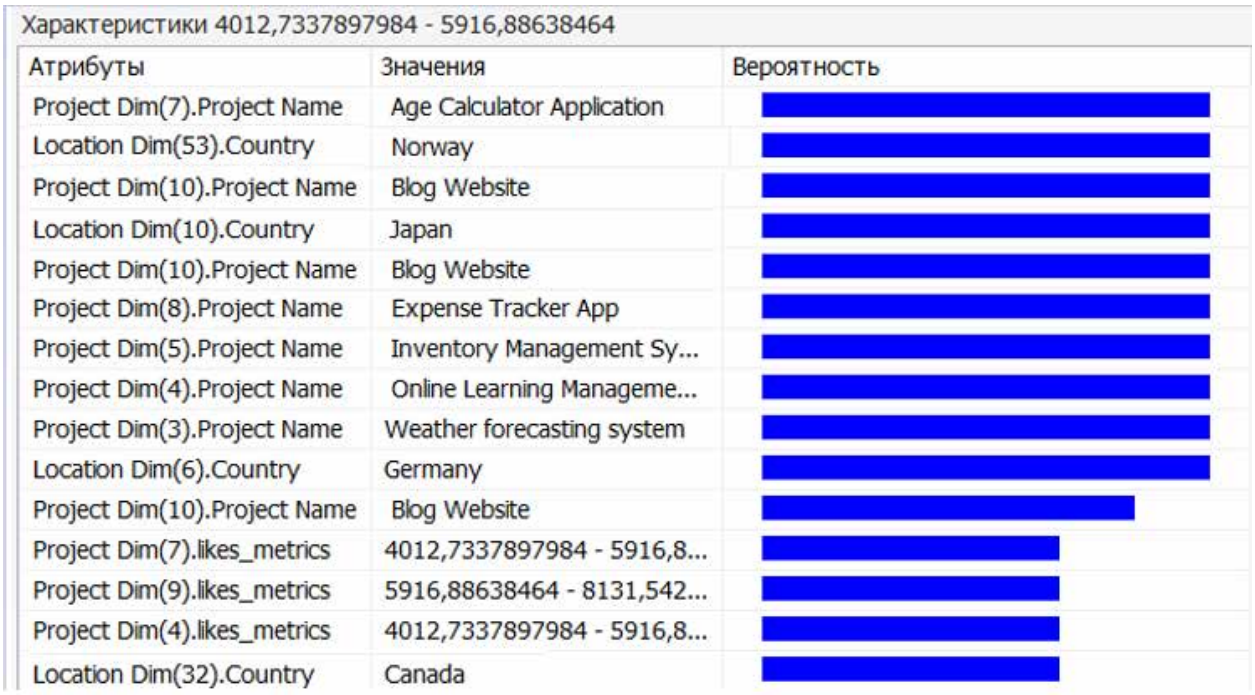
За результатами програмного алгоритму можна зробити висновок, що найменш популярними за кількістю лайків є наступні комбінації: Online Chat Application з Японії, Task Management System з Норвегії та Blog Website з Німеччини/Норвегії/України. Це пов'язано з тим, що в цих випадках спостерігається низький рівень популярності, який досягає 100%. Це показано на малюнку 12.

	Online Chat Ap...	Japan	2	0	2	L	100,00%
	Task Managem...	Norway	5	0	5	L	100,00%
	Blog Website	Germany	1	0	1	L	100,00%
	Blog Website	Norway	1	0	1	L	100,00%
	Blog Website	Ukraine	1	0	1	L	100,00%

Рис. 12 результати непопулярності проекту та країни

Алгоритм також реалізовано за допомогою засобів SSAS. За результатами наївного алгоритму Байєса очікувана кількість лайків коливається від 4012 до 5916. Якщо говорити про ймовірності на рівні 100%, то це збігається з реалізацією програми. Ці проекти – Age Calculator Application, Blog Website, Online Learning Management System та країни –

Норвегія, Японія, Німеччина – відповідають цій ймовірності. Це показано на малюнку 13.



Характеристики 4012,7337897984 - 5916,88638464

Атрибуты	Значения	Вероятность
Project Dim(7).Project Name	Age Calculator Application	
Location Dim(53).Country	Norway	
Project Dim(10).Project Name	Blog Website	
Location Dim(10).Country	Japan	
Project Dim(10).Project Name	Blog Website	
Project Dim(8).Project Name	Expense Tracker App	
Project Dim(5).Project Name	Inventory Management Sy...	
Project Dim(4).Project Name	Online Learning Manageme...	
Project Dim(3).Project Name	Weather forecasting system	
Location Dim(6).Country	Germany	
Project Dim(10).Project Name	Blog Website	
Project Dim(7).likes_metrics	4012,7337897984 - 5916,8...	
Project Dim(9).likes_metrics	5916,88638464 - 8131,542...	
Project Dim(4).likes_metrics	4012,7337897984 - 5916,8...	
Location Dim(32).Country	Canada	

Рис. 13 Результати за допомогою інструментів SSAS

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Вимоги до апаратного та програмного забезпечення

Забезпечення безпеки апаратного та програмного забезпечення має першорядне значення для підтримки цілісності, конфіденційності та доступності системи. Система оброблятиме конфіденційні дані, пов’язані з різними веб-проектами, включаючи інформацію про користувачів, специфікації проекту та фінансові деталі. Тому необхідно впроваджувати надійні заходи безпеки як на апаратному, так і на програмному рівнях.

Апаратна безпека системи буде захищена захищеними серверами, гарантуючи, що фізична інфраструктура, на якій розміщено програму, захищена від несанкціонованого доступу та втручання. Це включає використання захищених серверів із зашифрованим сховищем для конфіденційних даних, впровадження механізмів контролю доступу для обмеження фізичного доступу та використання апаратних модулів безпеки (HSM), де необхідно, для захисту криптографічних ключів.

Щоб запобігти потенційним атакам, таким як крадіжка даних або фізичні зломи, апаратні брандмауери та системи виявлення вторгнень (IDS) будуть налаштовані для моніторингу та блокування підозрілих дій. Будуть проводитися регулярні перевірки, щоб переконатися, що обладнання залишається в актуальному стані з останніми виправленнями безпеки та конфігураціями.

З боку програмного забезпечення система включатиме кілька рівнів заходів безпеки для запобігання несанкціонованому доступу, витоку даних та іншим кіберзагрозам. Одним із ключових елементів є шифрування даних — як під час передачі, так і в стані спокою — гарантуючи, що всі конфіденційні дані, такі як деталі проекту, облікові дані користувача та

фінансові транзакції, надійно зашифровані. Захищені протоколи, такі як HTTPS, використовуватимуться для зв'язку між клієнтом і сервером, щоб запобігти атакам типу «людина посередині».

Будуть реалізовані механізми контролю доступу, щоб забезпечити взаємодію з системою лише авторизованим користувачам. Контроль доступу на основі ролей (RBAC) використовуватиметься для призначення конкретних привілеїв користувачам на основі їхніх ролей у системі, обмежуючи доступ до конфіденційних даних лише тим, кому вони потрібні. Двофакторна автентифікація (2FA) використовуватиметься для входу користувачів, щоб додати додатковий рівень безпеки, зменшуючи ймовірність несанкціонованого доступу.

Система також проходитиме регулярне тестування безпеки, включаючи тестування на проникнення, для виявлення та усунення вразливостей у програмі. Під час процесу розробки дотримуватимуться методів безпечного кодування, щоб запобігти типовим уразливостям, таким як впровадження SQL, міжсайтовий сценарій (XSS) і підробка міжсайтового запиту (CSRF).

Крім того, оновлення програмного забезпечення та виправлення будуть регулярно застосовуватися для забезпечення захисту системи від останніх загроз безпеці. Використання брандмауерів, засобів захисту від зловмисного програмного забезпечення та систем запобігання вторгненням (IPS) стане невід'ємною частиною виявлення та пом'якшення будь-яких потенційних загроз у режимі реального часу.

Система має клієнт-серверну архітектуру. Клієнт-серверна архітектура — це модель розподіленої обчислювальної системи, де функції системи розділені між двома основними типами програмних компонентів:

клієнтами і серверами. Ця архітектурна модель забезпечує ефективний обмін інформацією та виконання задач між різними частинами системи.

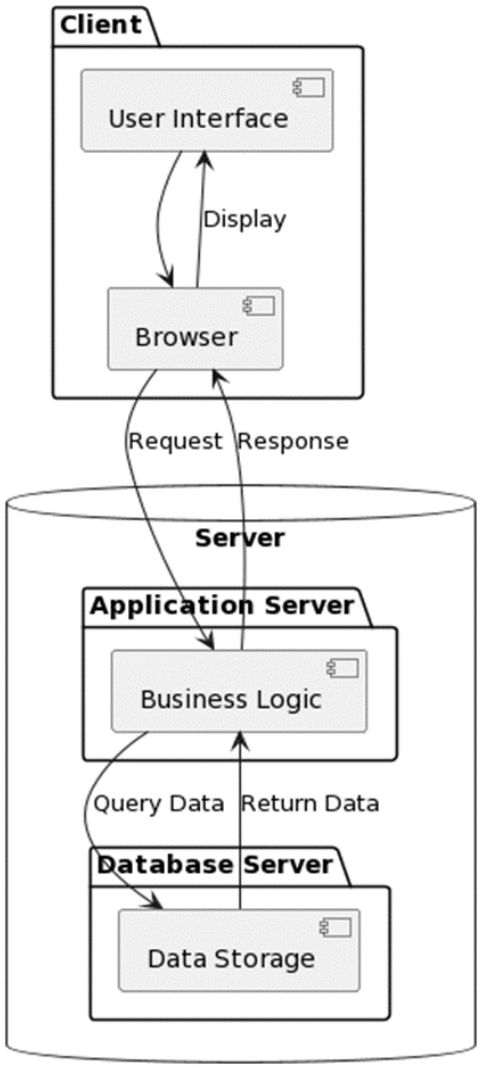


Рис. 14 Топологія системи

На рис. 14 представлена топологія системи. Вона показує взаємозв'язок між різними компонентами системи.

На вершині блок-схеми знаходиться користувач. Користувач взаємодіє з системою за допомогою браузера. Браузер відправляє запит на сервер.

Сервер відповідає на запит користувача. Відповідь сервера містить дані, які відображаються в браузері.

Сервер також відповідає за зберігання даних. Дані зберігаються в базі даних.

Топологію системи можна розділити на два основних компоненти:

- клієнт - це комп'ютер користувача, який взаємодіє з системою;
- сервер - це комп'ютер, який обробляє запити користувачів і надає їм доступ до даних.

Клієнтський компонент системи складається з двох частин:

- інтерфейс користувача (UI) - це те, що бачить і чує користувач. Це може включати в себе веб-сторінки, зображення, відео та інші елементи;
- браузер - це програма, яка використовується для відображення інтерфейсу користувача.

Серверний компонент системи складається з трьох частин:

- сервер застосунків (AS) - це програма, яка обробляє запити користувачів;
- бізнес-логіка - це правила і процедури, які використовуються для обробки запитів користувачів;
- база даних - це місце, де зберігаються дані, які використовуються системою.

4.2 Тестування системи

Запустивши систему, бачимо головну сторінку програмного забезпечення (рис. 15).

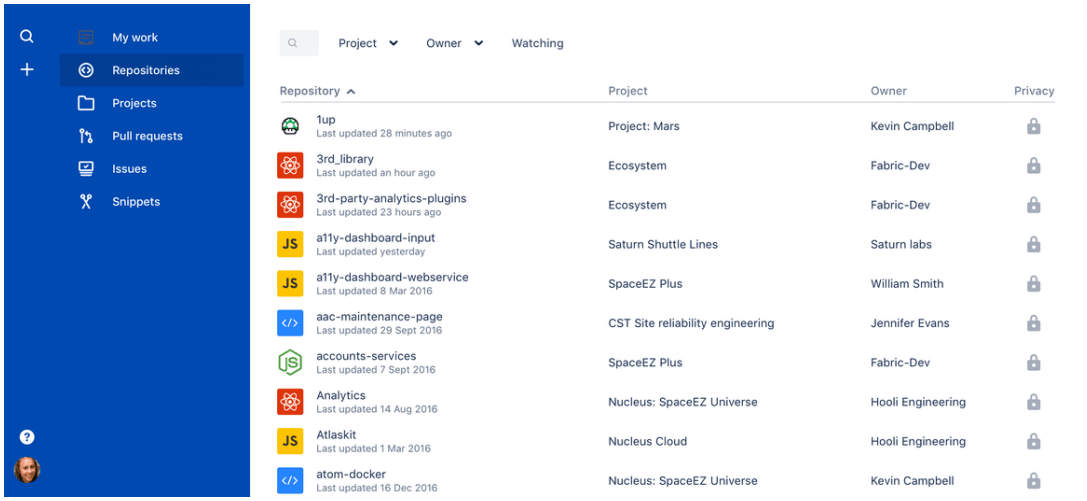


Рис. 15 Головна сторінка

На ній ми можемо переглянути всі знайдені проекти, їх назву, опис, автора, дату створення. Обравши проект, ми можемо відкрити його налаштування (Рис. 16)

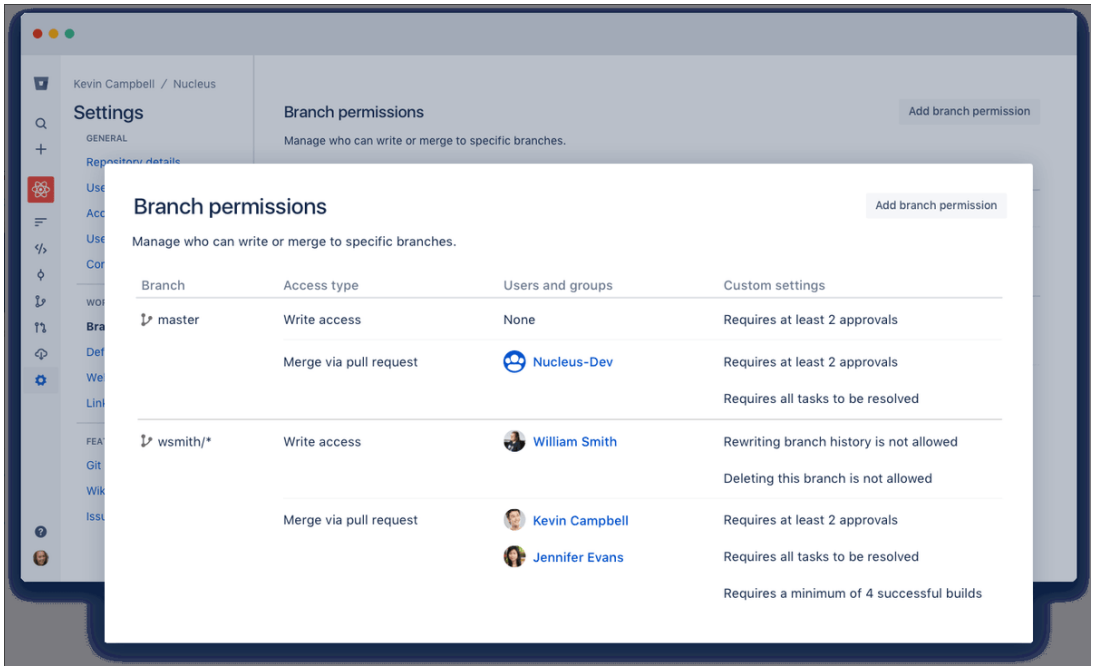


Рис. 16 Налаштування проектів

У налаштуваннях проекту ми бачимо всі допустимі гілки їхніх власників. Далі у кожній галці ми можемо перейти до перегляду коду (Рис. 17)

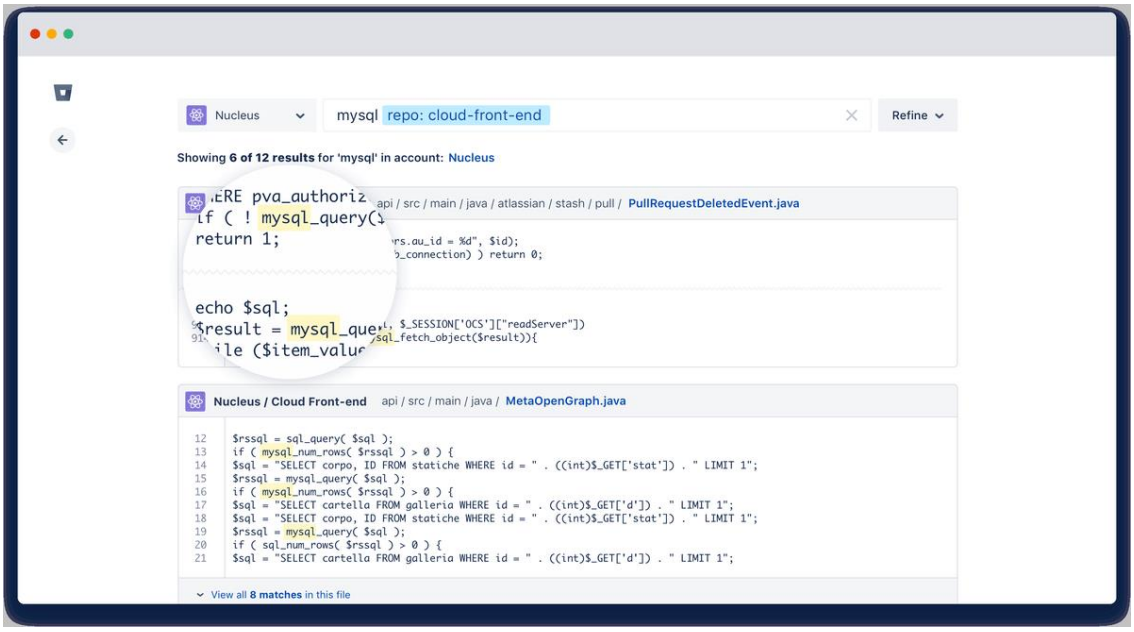


Рис. 17 Перегляд вмісту проектів

Тут ми можемо повністю переглядати код у вибраній гілці. Кожен рядок можна буде відредагувати або залишити повідомлення (Рис. 18)

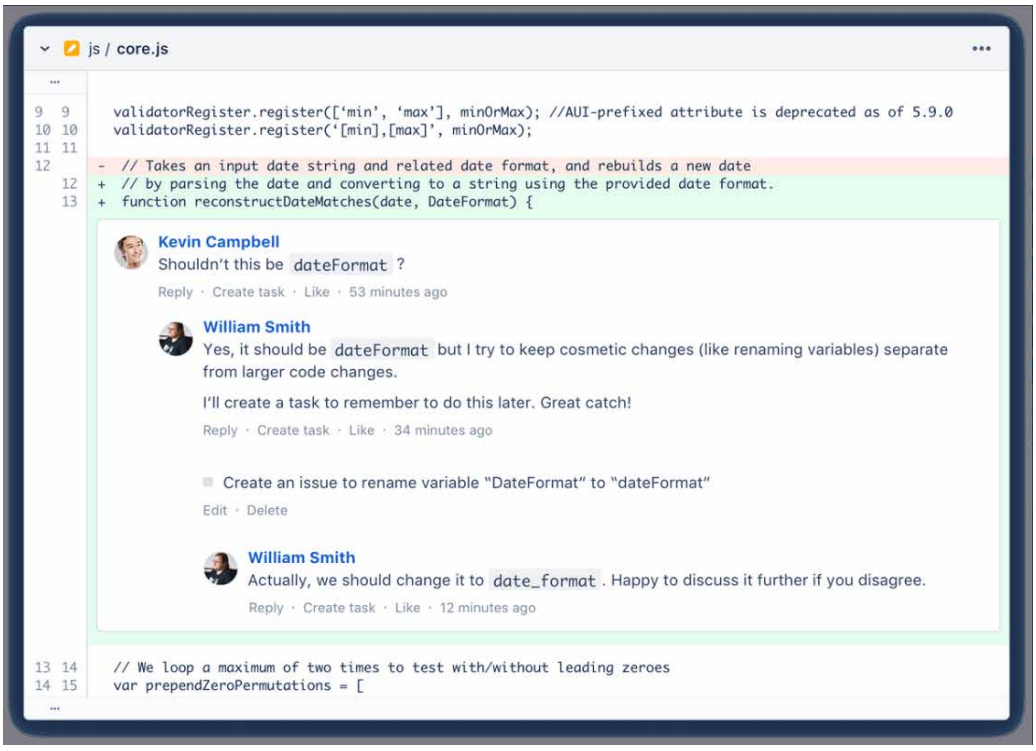


Рис. 18 Можливість залишати коментарі

Залишаючи повідомлення, ми можемо давати зрозуміти іншим, на що варто звернути увагу у гілці.

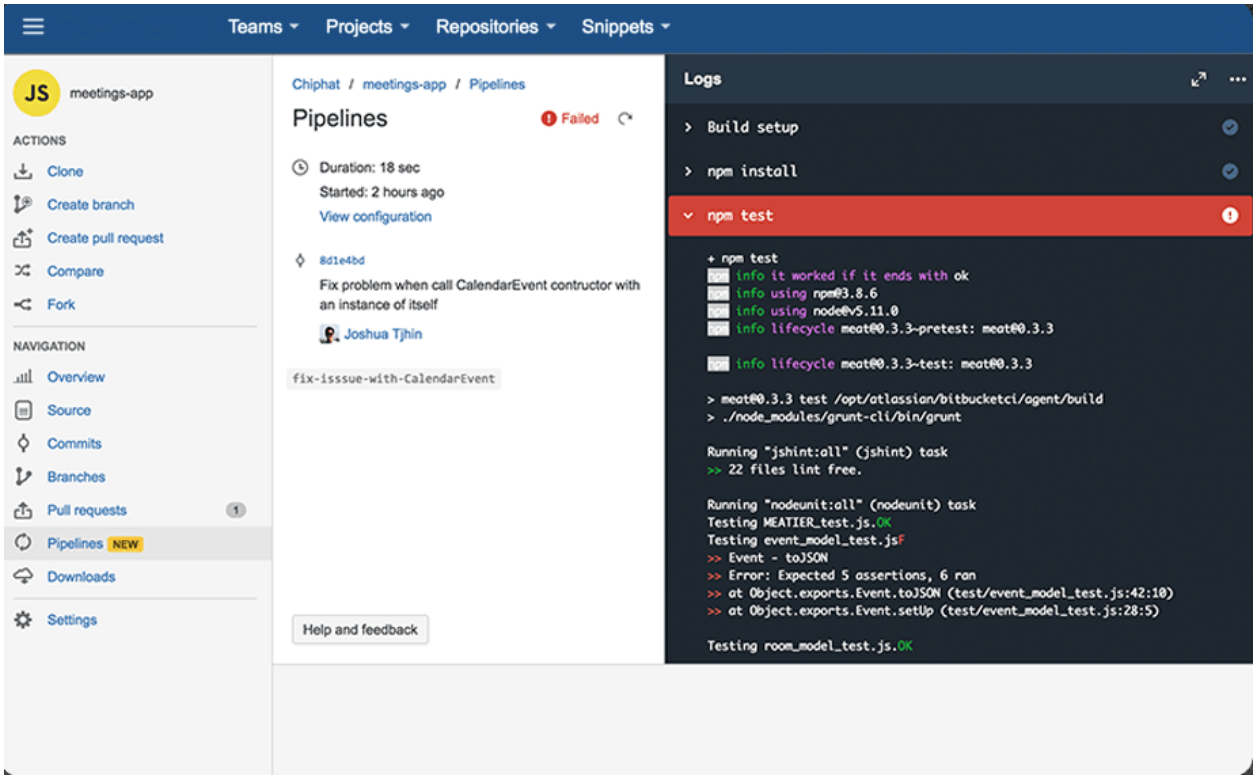


Рис. 19 Завантаження проекту локально

І, зрештою, ми можемо встановити проект собі локально та користуватися ним на своєму робочому пристрої.

4.3 Інструкція користувача

Інтелектуальна система обліку веб-проектів розроблена для оптимізації управління та відстеження веб-проектів, дозволяючи користувачам класифікувати, шукати та аналізувати проекти, зберігаючи детальні записи. Система зручна для користувача та пропонує ряд функцій, які допомагають користувачам швидко отримувати важливу інформацію, пов’язану з їхніми веб-проектами. У цьому посібнику користувача

викладено ключові функціональні можливості системи та надано чіткі інструкції щодо взаємодії з різними функціями.

Перед використанням системи переконайтеся, що ваш пристрій відповідає необхідним специфікаціям: він має працювати під керуванням сучасної операційної системи, як-от Windows, macOS або Linux, і підтримувати такі популярні веб-браузери, як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge. Для доступу до програми необхідне стабільне підключення до Інтернету, і користувачі повинні мати дійсні облікові дані для входу.

Після входу в систему користувача вітає інформаційна панель, яка є центральним центром системи. Звідси користувачі можуть переглядати огляд усіх веб-проектів, отримувати доступ до панелі пошуку для запитів проектів, застосовувати фільтри для категоризації проектів і отримувати важливі сповіщення. Інформаційна панель забезпечує легку навігацію до інших розділів, таких як керування проектами, категоризація та аналіз даних.

Функція пошуку системи дозволяє користувачам швидко знаходити певні веб-проекти, вводячи релевантні ключові слова або фрази. Просто введіть назву проекту, категорію чи будь-який інший ідентифікаційний термін у рядок пошуку та натисніть кнопку пошуку. З’явиться список відповідних результатів, і користувачі зможуть натиснути будь-який проект, щоб переглянути докладнішу інформацію.

Категоризація проектів є ще однією важливою функцією, яка допомагає користувачам залишатися організованими. Проекти можна класифікувати за різними категоріями, такими як електронна комерція, медицина чи криптовалюта. Щоб класифікувати проект, користувачі можуть отримати доступ до розділу керування проектом, заповнити

необхідні деталі, такі як назва та опис проекту, вибрати відповідну категорію зі спадного меню та зберегти запис. За потреби також можна створити нові категорії.

Управління та редагування деталей проекту просте. У розділі огляду проекту користувачі можуть переглянути всі зареєстровані проекти. За допомогою пошуку або перегляду списку користувачі можуть знайти певний проект і натиснути на нього, щоб переглянути його деталі. Якщо потрібні будь-які оновлення, натиснувши кнопку «Редагувати», користувачі зможуть змінити інформацію про проект. Після внесення змін користувачі можуть зберегти оновлені дані проекту.

Система також містить опцію видалення проекту. Щоб видалити проект, користувачі можуть знайти проект у розділі огляду, натиснути кнопку «Видалити» поруч із проектом і підтвердити видалення. Видалені проекти не видаляються негайно, а архівуються, що дозволяє відновити їх у разі необхідності.

Що стосується аналізу даних, система пропонує інструменти для аналізу різних аспектів проектів, таких як використання ресурсів, статус проекту та часові рамки. Користувачі можуть отримати доступ до розділу аналізу даних, вказати фільтри, наприклад діапазони дат, і створити звіти. Потім ці звіти можна експортувати в різні формати, включаючи CSV і PDF, для подальшого використання.

Ще одна ключова функція — можливість експортувати дані проекту. Користувачі можуть вибирати проекти на основі своїх конкретних критеріїв і експортувати їх у бажаному форматі. Після вибору проектів і

формату користувачі можуть завантажити файл на свій пристрій для використання в автономному режимі або ведення записів.

Для керування обліковими записами користувачів система дозволяє користувачам змінювати свої паролі та виходити, коли вони закінчать. У розділі налаштувань користувача користувачі можуть оновити свій пароль, ввівши поточний пароль, а потім новий пароль. Після збереження змін буде застосовано новий пароль. Щоб вийти, просто натисніть значок профілю у верхньому правому куті та виберіть «Вийти».

У разі будь-яких проблем у посібнику наведено кроки щодо усунення несправностей. Якщо користувач не може ввійти, важливо перевірити облікові дані для входу та скинути пароль, якщо потрібно. Відсутні проекти зазвичай можна знайти, перевіряючи фільтри або шукаючи безпосередньо за назвою проекту. Якщо система працює повільно, оновіть сторінку або очистіть кеш браузера, щоб вирішити проблему.

Ця система є комплексним рішенням для управління та аналізу веб-проектів. Його інтуїтивно зрозумілий інтерфейс і потужні функції допомагають користувачам вести впорядковані записи та отримувати цінну інформацію про продуктивність проекту. Якщо виникнуть будь-які труднощі, розділ усунення несправностей або підтримка системного адміністратора завжди доступна для допомоги користувачам.

ВИСНОВКИ

Підсумовуючи, розробка інтелектуальної системи обліку веб-проектів представляє значний прогрес у способі керування, відстеження та аналізу веб-проектів. Система задовольняє зростаючу потребу в ефективній категоризації проектів, пошуку даних і комплексному аналізі, зберігаючи при цьому зручний інтерфейс. Завдяки використанню сучасних технологій, таких як C#, ASP.NET, MySQL і EF Core, система забезпечує безперебійну продуктивність, масштабованість і надійні можливості керування даними.

Основні функції системи, включаючи пошук проектів, категоризацію, керування та аналіз даних, надають користувачам потужний інструмент для вдосконалення процесів відстеження проектів. Це дозволяє оновлювати в режимі реального часу, класифікувати проекти за різними темами та можливість легко експортувати дані для подальшого аналізу. Інтелектуальні алгоритми, вбудовані в систему, дозволяють краще приймати рішення та ефективніше керувати веб-проектами.

Протягом усього процесу розробки використовувалися різні принципи розробки програмного забезпечення, включаючи об’єктно-орієнтоване проектування та методи гнучкої розробки, щоб забезпечити ефективність і надійність системи. Система розроблена, щоб задовольнити потреби користувачів, починаючи від керівників проектів і аналітиків до інвесторів, які шукають розуміння веб-проектів.

Дизайн і впровадження проекту демонструють потенціал інтеграції інтелектуальних функцій в інструменти управління проектами, що робить його незамінним рішенням для керування веб-проектами. Крім того, адаптивність системи до різних категорій веб-проектів разом із простим у

використанні інтерфейсом позиціонує її як цінний ресурс для окремих осіб і організацій, які займаються веб-розробкою та управлінням проектами.

У майбутньому систему можна буде вдосконалити, включивши методи машинного навчання для забезпечення прогнозної аналітики або розширивши її базу даних, включивши ще більше різноманітних категорій проектів. Крім того, подальша оптимізація продуктивності системи та впровадження додаткових функцій, таких як співпраця в режимі реального часу, може зробити її ще більш ефективною для задоволення потреб користувачів.

Загалом, інтелектуальна система обліку веб-проектів пропонує комплексне та ефективне рішення для керування й аналізу веб-проектів, а її розробка закладає міцну основу для майбутніх удосконалень і розширення в цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Use Cases. – [Електронний ресурс]. – Режим доступу: https://www.usability.gov/how-to-and-tools/methods/use-cases.html
2. Overview of Online Analytical Processing (OLAP). – [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/enus/office/overview-of-online-analytical-processing-olap-15d2cddef70b-4277-b009-ed732b75fdd6
3. Що таке OLAP? – [Електронний ресурс]. – Режим доступу: https://uk.education-wiki.com/5528785-what-is-olap
4. What is MOLAP (Multidimensional OLAP) in Data Warehouse? –
5. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/multidimensional-online-analyticalprocessing.html
6. What Is a Data Warehouse? – [Електронний ресурс]. – Режим доступу: https://www.oracle.com/database/what-is-a-data-warehouse/
7. SSAS Tutorial: What is SSAS Cube, Architecture & Types . –
8. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/ssas-tutorial.html
9. Data Source Views in Multidimensional Models. – [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/analysisservices/multidimensional-models/data-source-views-inmultidimensional-models?view=asallproducts-allversions
10. SQL Server Integration Services (SSIS) для начинающих – часть 1. – [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/330618/
11. Які основні поняття інтелектуального аналізу даних? – [Електронний ресурс]. – Режим доступу:

https://www.tutorialspoint.com/what-are-the-basic-concepts-of-data-mining

12. Які 7 найкращих інструментів інтелектуального аналізу даних? – [Електронний ресурс]. – Режим доступу: https://careerfoundry.com/en/blog/data-analytics/best-data-mining-tools/
13. «Машинне навчання: імовірнісна перспектива» Кевіна П. Мерфі.
14. Педрегоза, Ф., Варокво, Г., Грамфор, А., Мішель, В., Тіріон, Б., Грізель, О., ... і Вандерплас, Дж. (2011). Scikit-learn: машинне навчання на Python. Журнал досліджень машинного навчання, 12 (жовтень), 2825-2830
15. Association Rule Learning (ARL) – [Електронний ресурс]. –Режим доступу: https://medium.com/@kaanerdenn/association-rule-learning-arl-f2e6bf35bd98
16. Introduction to Clustering Algorithms in Data Mining- [Електронний ресурс]. –Режим доступу: https://datatrained.com/post/best-clustering-algorithms-in-data-mining/#:~:text=Clustering%20algorithms%20in%20data%20mining%20are%20an%20unsupervised%20Machine%20Learning,has%20дані%20як%20один%20інший
17. MySQL – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
18. Microsoft, C#.NET Documentation – [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
19. Соммервіль І. (2011). Інженерія програмного забезпечення (9-е вид.). Бостон: Addison-Wesley.
20. ASP.NET Core Documentation – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/
21. Хокінс, Дж. (2015). Управління веб-проектами: повний посібник. Нью-Йорк: Pearson Education.

22. Фаулер, М. (2003). Шаблони архітектури корпоративних додатків. Аддісон-Уеслі.
23. O'Рейлі, Т. (2005). Web 2.0: компактне визначення. O'Reilly Media. Доступно за адресою: https://www.oreilly.com/pub/a/web2.0/
24. Database Design for Mere Mortals, 3rd Edition – [Електронний ресурс] – Режим доступу: https://www.pearson.com/store/p/database-design-for-mere-mortals/P100000715011
25. Дописувачі Вікіпедії. (2023). Інтелектуальна система. Вікіпедія. Доступно за адресою: https://en.wikipedia.org/wiki/Intelligent_system
26. Готфрід, М. (2017). Вивчення SQL: основні основи SQL (3-є видання). O'Reilly Media.
27. Йоргенсен, М. (2018). Управління проектами інформаційних систем (4-е вид.). Лондон: Пірсон.
28. Коллер, Д., Крентел, М. (2019). Повний посібник із ядра EF і керування даними (1-е видання). Нью-Йорк: Packt Publishing.
29. Лю К. (2019). Розуміння методів аналізу даних для веб-додатків. Спрингер.
30. GitHub – [Електронний ресурс] – Режим доступу: https://github.com/
31. Microsoft. (2022). Основна документація Entity Framework. Доступно за адресою: https://docs.microsoft.com/en-us/ef/core/
32. Вірт, Н. (2009). Алгоритми та структури даних: об’єктно-орієнтований підхід. Спрингер.
33. Малік, Д. (2021). Освоєння управління веб-проектами в цифрову еру. Лондон: Routledge.
34. Веб-документи Mozilla Developer Network (MDN) – [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web

ДОДАТОК А

Фрагменти програмного коду. Запити створення об’єктів сховища даних

```
CREATE TABLE dim_project (
    id INT PRIMARY KEY,
    project_name VARCHAR(255),
    description TEXT,
    creation_date DATE
);

CREATE TABLE dim_time (
    id INT PRIMARY KEY,
    year INT,
    month INT,
    day INT,
);


CREATE TABLE dim_location (
    id INT PRIMARY KEY,
    country VARCHAR(255)
);

CREATE TABLE dim_category (
    id INT PRIMARY KEY,
    name VARCHAR(255)
);

CREATE TABLE fact_project_metrics (
    id INT PRIMARY KEY IDENTITY(1,1),
    clicks INT,
    views INT,
    likes INT,
    project_id INT,
    category_id INT,
    time_dimension_id INT,
    location_dimension_id INT,
    FOREIGN KEY (project_id) REFERENCES dim_project(id),
    FOREIGN KEY (time_dimension_id) REFERENCES dim_time(id),
    FOREIGN KEY (location_dimension_id) REFERENCES dim_location(id),
    FOREIGN KEY (category_id) REFERENCES dim_category(id)
);
```

ДОДАТОК Б

Фрагменти програмного коду. Функція перегляду до завантаження проектів

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

[Route("api/[controller]")]
[ApiController]
public class ProjectsController : ControllerBase
{
    private readonly ApplicationDbContext _context;

    public ProjectsController(ApplicationDbContext context)
    {
        _context = context;
    }

    [HttpPost]
    public async Task<IActionResult> AddProject([FromBody] Project project)
    {
        if (project == null)
        {
            return BadRequest("Invalid project data.");
        }

        project.DateCreated = DateTime.Now; // Устанавливаем дату создания проекта
        _context.Projects.Add(project);
        await _context.SaveChangesAsync();

        return CreatedAtAction(nameof(GetProject), new { id = project.Id }, project);
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<Project>>> GetProjects()
    {
        var projects = await _context.Projects.ToListAsync();
        return Ok(projects);
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<Project>> GetProject(int id)
    {
        var project = await _context.Projects.FindAsync(id);

        if (project == null)
        {
            return NotFound();
        }

        return Ok(project);
    }
}

public class Startup
{
```

```
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

        services.AddControllers();
    }

    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseRouting();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```