МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 2002–“С” 2023.01.11. 023 ПЗ

ГУМЕННОГО ІВАНА ОЛЕКСАНДРОВИЧА

2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.777

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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Інтелектуальна система створення HTML-розмітки інтерфейсів веб-сторінок

Спеціальність 121 - Інженерія програмного забезпечення
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

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

Гарант освітньої програми

д.т.н., професор _____ Семко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

д.т.н., професор _____ Семко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Гуменний І.О.
(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)
"1" листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Гуменний Іван Олександрович
(прізвище, ім'я, по батькові)

Спеціальність 121 - Інженерія програмного забезпечення
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

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

Тема магістерської кваліфікаційної роботи Інтелектуальна система створення HTML-розмітки інтерфейсів веб-сторінок
затверджена наказом ректора НУБіП України від "1" листопада 2024р. № 2002 С

Термін подання завершеної роботи на кафедру 29.11.2024
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи отримання рекомендацій щодо покращення якості зовнішнього вигляду та функціоналу програмних веб-додатків та веб-сервісів

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 "1" листопада 2024 р.

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

Завдання прийняв до виконання _____ Гуменний І.О.
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Зростаючий попит на ефективні, орієнтовані на користувача та чутливі веб-інтерфейси зробив автоматизацію створення розмітки HTML гострою потребою у веб-розробці. Створення HTML-розмітки вручну може бути трудомістким, чутливим до помилок і займає багато часу, особливо для складних інтерфейсів. Розробка інтелектуальної системи для автоматизації генерації розмітки HTML спростить процес розробки, мінімізує людські помилки та сприятиме узгодженості веб-проектів. Таким чином, це дослідження узгоджується з ширшим галузевим рухом до інтелектуальної автоматизації розробки програмного забезпечення, підкреслюючи його **актуальність** у сучасному веб-дизайні.

Дослідження зосереджено на створенні рішення, яке може інтерпретувати специфікації веб-дизайну та автоматично перетворювати їх у функціональну розмітку HTML. Інтелектуальна система, запропонована в цьому дослідженні, слугуватиме як основним **предметом**, так і **об'єктом** дослідження, оскільки вона безпосередньо пов'язана з процесами та методами, які використовуються для розробки автоматизованих рішень для створення розмітки HTML.

Мета дослідження спрямована на покращення існуючих систем, здатних точно генерувати розмітку HTML для веб-сторінок з мінімальним ручним втручанням. Щоб досягти цього, окреслено кілька завдань, починаючи з вичерпного огляду поточних методів генерації HTML. Після цього аналізу буде створено модель системи, що відображатиме ключові вимоги до автоматизації HTML. Потім дослідження переходить до розробки інтелектуальної системи з використанням відповідних алгоритмів і фреймворків і завершується оцінкою продуктивності, зосередженою на зручності використання, точності та швидкості.

Цей підхід включає різні **методи дослідження**, включаючи аналіз системи для оцінки існуючих процесів розмітки та моделювання системи для створення плану інтелектуальної системи. Етап розробки включатиме використання методів штучного інтелекту, ймовірно, машинного навчання або алгоритмів на основі правил, щоб полегшити автоматизовану генерацію HTML. Тестування та оцінка вимірюють якість результату роботи системи та оцінять її практичну ефективність у створенні надійного, семантично точного та доступного HTML-коду.

Наукова новизна цього дослідження полягає в застосуванні інтелектуальних алгоритмів для створення розмітки HTML, що пропонує унікальний підхід, який зменшує потребу в створенні коду вручну. Впроваджуючи та тестуючи цю автоматизовану систему, дослідження сприяє інноваційному рішенню у сфері веб-розробки, усуваючи розрив між специфікаціями дизайну та виконуваним кодом HTML.

Апробація результатів дослідження - дана магістерська дипломна робота побудована таким чином, щоб охопити всі аспекти концепції, розробки та тестування системи. По-перше, аналіз предметної області досліджує обмеження ручної розмітки та переглядає існуючі засоби автоматизації. Далі представлена детальна модель запропонованої системи, включаючи її архітектуру, потік даних і обрані алгоритми. Розділ розробки документує фактичне створення системи, окреслюючи середовище, інструменти та використовувані рамки. Дослідження завершується обговоренням результатів тестування, що демонструє здатність системи досягати цілей точного й ефективного створення HTML.

Магістерська робота містить 81 сторінки, в тому числі 17 рисунків, 1 таблицю, 2 додатки. Вона посилається на 30 джерел, включаючи наукові статті, книги та електронні ресурси.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Процес створення HTML-розмітки для інтерфейсів веб-сторінок є основною діяльністю веб-розробки, яка має вирішальне значення для структури, доступності та загальної функціональності веб-сайтів. HTML служить скелетом кожної веб-сторінки, забезпечуючи не лише основу для макета та стилю, але й важливі зв’язки для взаємодії, швидкості реагування та доступності. Історично розмітка HTML писалася розробниками вручну, причому кожен елемент дизайну ретельно перекладався в код. Хоча ручне кодування забезпечує точний контроль, воно також може зайняти багато часу, бути схильним до помилок і сильно залежати від кваліфікації розробника. Оскільки веб-сайти стають дедалі складнішими, а вимоги до взаємодії з користувачами зростають, обмеження ручного створення HTML стають очевидними, що спонукає до пошуку більш ефективних та інтелектуальних рішень.

Однією з головних проблем під час створення розмітки HTML вручну є значні витрати часу та зусиль, особливо для сучасних веб-сайтів, які часто складаються з кількох сторінок з інтерактивними елементами та мультимедійним вмістом. Розробники витрачають багато часу на кодування кожного компонента інтерфейсу та підтримання узгодженості подібних елементів. З кожною зміною інтерфейсу розробники повинні переглядати, оновлювати та часто переписувати HTML, щоб переконатися, що він відображає останній дизайн. Ця повторювана ручна робота, особливо в проектах, які вимагають частих оновлень, може призвести до неефективності, забираючи час на виконання інших критичних завдань, таких як реалізація складних функціональних можливостей або вдосконалення взаємодії з користувачем. Для розробників, які працюють у стислі терміни, цей процес

може сповільнити загальні терміни проекту та збільшити витрати на розробку[1].

Створення розмітки вручну також створює значний ризик людської помилки, особливо під час обробки великих складних інтерфейсів. Навіть невеликі помилки в структурі HTML, як-от відсутність закриваючих тегів або неправильно вкладені елементи, можуть призвести до візуальних неузгодженостей або функціональних проблем у різних браузерах і пристроях. Ці помилки не тільки впливають на зовнішній вигляд інтерфейсу, але також можуть порушити взаємодію користувача, особливо якщо вони впливають на адаптивні макети на мобільних пристроях. Враховуючи широкий спектр пристроїв і розмірів екранів, які сьогодні мають підтримувати веб-сайти, забезпечення правильного та адаптованого HTML стає дедалі складнішим лише за допомогою ручних засобів.

Крім того, ручному створенню HTML часто не вистачає вбудованих механізмів для забезпечення дотримання стандартів доступності, залишаючи розробникам завдання гарантувати відповідність інструкціям щодо доступності веб-сайтів, таким як ті, що визначені Інструкціями щодо доступності веб-контенту (WCAG). Важливі аспекти доступності, зокрема ролі ARIA (Accessible Rich Internet Applications), альтернативний текст для зображень і навігація за допомогою клавіатури, іноді не помічаються або застосовуються непослідовно. Цей недогляд може призвести до того, що веб-інтерфейси стануть менш доступними для користувачів з обмеженими можливостями, зменшуючи інклюзивність і зручність використання веб-сайту. Оскільки цифрова доступність набуває важливості в веб-розробці, забезпечення того, щоб розмітка HTML за своєю суттю підтримувала функції доступності, стало пріоритетом, але ручне кодування HTML часто не вдається в цьому відношенні.

Розвиток робочих процесів спільної та гнучкої розробки також підкреслив обмеження ручної розмітки HTML. Сучасна розробка часто

передбачає співпрацю кількох команд — дизайнерів, розробників, творців вмісту, а іноді навіть кінцевих користувачів — для втілення інтерфейсу в життя. Дизайнери створюють детальні макети та прототипи в таких інструментах, як Figma, Sketch або Adobe XD, а розробники вручну інтерпретують ці проекти для кодування остаточного інтерфейсу. У цьому процесі передачі критичні нюанси дизайну можуть бути втрачені, що призведе до розбіжностей між запланованим дизайном і кінцевим продуктом. Такі розбіжності можуть вимагати додаткового часу для перегляду, порушуючи робочий процес і іноді призводячи до розчарування серед членів команди. Крім того, у міру розвитку інструментів проектування зростає потреба в бездоганній інтеграції між фазами проектування та розробки, вимога, яку часто важко виконати при ручному створенні HTML[1].

Поява інтелектуальних систем для створення розмітки HTML містить потенціал для вирішення цих проблем шляхом автоматизації значної частини ручної роботи, пов’язаної з перекладом специфікацій дизайну в код HTML. Така система використовуватиме методи штучного інтелекту (AI) і машинного навчання (ML) для аналізу елементів дизайну, таких як макети, колірні схеми, шрифти та інтерактивні компоненти, і автоматичного створення відповідного HTML-коду. Це значно зменшить навантаження на розробників, дозволяючи їм зосередитися на завданнях вищого рівня та гарантувати, що згенерована розмітка узгоджено відповідає специфікаціям дизайну. Система також може бути розроблена для застосування найкращих практик у структурі HTML, гарантуючи, що згенерований код є семантично правильним, організованим і простим у обслуговуванні, перевага, яку ручне кодування не завжди гарантує.

Інтелектуальна система розмітки HTML також матиме можливість вбудовувати функції доступності безпосередньо в згенерований код, покращуючи інклюзивність, не вимагаючи додаткових зусиль від розробників. Завдяки автоматичному додаванню ролей ARIA, альтернативного тексту та доступних параметрів навігації система допоможе забезпечити відповідність

стандартам доступності, зробивши веб-інтерфейси більш зручними для людей з обмеженими можливостями. Ця автоматизація міркувань доступності принесе значну цінність організаціям, які віддають перевагу інклюзивності, але можуть не мати ресурсів для ручної перевірки кожного аспекту свого HTML на доступність.

З точки зору адаптивності, інтелектуальну систему можна навчити створювати адаптивні макети HTML, які адаптуються до різних розмірів екрану, оптимізуючи інтерфейс як для мобільних пристроїв, так і для робочих столів. Ця функція стає все більш важливою, оскільки мобільний веб-перегляд продовжує зростати, вимагаючи веб-інтерфейсів для бездоганної роботи на різних пристроях. Завдяки автоматизованій системі розробники могли б легше переконатися, що згенерований HTML відповідає вимогам оперативності, усуваючи потребу в трудомістких налаштуваннях і тестуванні на різних пристроях.

Предметна область «Інтелектуальної системи для створення розмітки HTML для інтерфейсів веб-сторінок» знаходиться на стику веб-розробки, штучного інтелекту та автоматизованої інтерпретації дизайну. Інтерфейси веб-сторінок є важливими компонентами Інтернету та служать основним засобом взаємодії користувачів із онлайн-вмістом. Ці інтерфейси створюються за допомогою HTML (HyperText Markup Language), основної мови розмітки для структурування веб-сторінок. HTML визначає, як елементи на сторінці, такі як текст, зображення, кнопки та посилання, відображаються та впорядковуються, тим самим формуючи взаємодію з користувачем і забезпечуючи безперебійну взаємодію з веб-сайтами.

Традиційно створення HTML-розмітки — це ручний процес, який виконують інтерфейсні розробники, які перетворюють специфікації дизайну в структурований HTML-код. Цей процес займає багато часу, тому розробники повинні ретельно створювати кожен елемент сторінки та забезпечувати дотримання стандартів дизайну, швидкості реагування та доступності.

Незважаючи на те, що ручне створення HTML є ефективним, воно має обмеження щодо ефективності та масштабованості. Оскільки вимоги до веб-розробки зростають, особливо із збільшенням складності сучасних інтерфейсів, стала очевидною потреба у швидшому та точнішому способі створення розмітки HTML. Предмет цієї дипломної роботи зосереджений на подоланні цих обмежень шляхом розробки інтелектуальної системи, яка автоматизує процес створення HTML-коду, таким чином зменшуючи залежність від ручного кодування та покращуючи узгодженість веб-інтерфейсів[2].

За останні роки технології штучного інтелекту (AI) і машинного навчання (ML) розвинулися настільки, що їх можна використовувати для автоматизації повторюваних або логічних завдань у різних сферах. У контексті веб-розробки інтелектуальна система для створення HTML аналізувала б специфікації дизайну, такі як макети, колірні схеми, шрифти та інтерактивні елементи, і створювала відповідний HTML-код, який точно представляв би дизайн. Ця система могла б значно оптимізувати робочий процес розробки, усунувши багато ручних кроків, які зараз необхідні для втілення дизайну в життя в Інтернеті. Шляхом безпосередньої інтерпретації файлів дизайну інтелектуальна система генерації HTML подолає розрив між дизайном і кодом, мінімізуючи розбіжності та забезпечуючи точне відображення веб-інтерфейсів баченням дизайнера.

Інтелект системи, швидше за все, покладатиметься на комбінацію розпізнавання шаблонів і алгоритмів на основі правил, що дозволить їй розуміти загальні структури дизайну та застосовувати відповідні теги та структури HTML. Наприклад, він може розпізнавати макети сітки, заповнювачі зображень, текстові блоки та кнопки та генерувати HTML-код, який узгоджується з цими шаблонами. Крім того, ця система не тільки автоматизує створення макета, але також може інтегрувати семантичні та доступні практики HTML, створюючи код, який відповідає веб-стандартам і

покращує зручність використання для всіх користувачів, у тому числі для людей з обмеженими можливостями.

Чуйність є ще одним важливим аспектом розмітки HTML у контексті сучасного веб-дизайну. Сьогодні користувачі отримують доступ до веб-сайтів з різних пристроїв, включаючи настільні комп'ютери, планшети та смартфони, кожен з яких має різні розміри екрана та роздільну здатність. Чуйний дизайн гарантує бездоганну адаптацію веб-інтерфейсу до цих різних екранів, забезпечуючи оптимізовану роботу незалежно від пристрою. Ключовим аспектом цієї предметної області є можливість інтелектуальній системі генерувати HTML-розмітку, яка за своєю суттю є чуйною, зменшуючи потребу в ручному коригуванні коду та забезпечуючи узгоджену взаємодію з користувачем на різних пристроях. Використовуючи адаптивні фреймворки та методології, система може автоматизувати створення макетів, які динамічно адаптуються до різних екранів, що ще більше підвищить її практичну корисність у сучасній веб-розробці.

Іншим важливим елементом у цій тематичній області є доступність, наріжний камінь сучасного веб-дизайну, спрямованого на те, щоб зробити онлайн-вміст доступним для людей з обмеженими можливостями. Спеціальні можливості включають такі практики, як додавання ролей ARIA (Accessible Rich Internet Applications), підтримку навігації з клавіатури та текстові альтернативи для зображень. Як правило, розробники додають ці функції вручну, що вимагає досвіду та збільшує час, витрачений на кодування. Інтелектуальна система може автоматизувати функції доступності, вбудовуючи їх у згенерований HTML, сприяючи інклюзивності без додаткової складності для розробників. Це полегшило б організаціям виконання юридичних та етичних зобов'язань, одночасно покращуючи взаємодію з користувачами для різноманітної аудиторії.

1.2 Автоматизація розробки веб-сторінок

Предметною областю дослідження є автоматизація процесу розробки вебсторінок. Для автоматизованого створення HTML-інтерфейсів часто використовують спеціальні конструктори, проте вони нерідко мають певні недоліки, що ускладнюють процес розробки для недосвідчених користувачів. В той же час автоматизація розробки є важливою в першу чергу саме для користувачів, що не мають досвіду розробки програмного забезпечення.

Розробка веб-сайтів або веб-додатків (наприклад, з використанням технології WebView) стала невід'ємною частиною сучасного програмного забезпечення. Проте процес створення веб-сторінок традиційно вимагає від розробників значних технічних знань, особливо це стосується знань таких мов як HTML, CSS та JavaScript. Тому для значної частини користувачів, які не мають досвіду в програмуванні, створення веб-сторінок залишається складним і часозатратним завданням.

З появою різноманітних інструментів для автоматизації розробки з'явилася можливість значно спростити цей процес. Серед них варто відзначити візуальні редактори та конструктори сайтів, які дозволяють створювати веб-сторінки без написання коду. Проте, навіть з усіма перевагами таких інструментів, більшість з них часто обмежують можливості користувача в налаштуванні складних елементів дизайну, таких як адаптивні розмітки або кастомізація стилів. Крім того, багато з них потребують значних витрат часу на освоєння інтерфейсу та налаштування проекту.

Один із найбільш важливих факторів, який необхідно враховувати під час створення сучасних веб-сторінок – це адаптивність. Будь-яка веб-сторінка повинна коректно відображатись на різних пристроях, таких як настільні комп'ютери, планшети, мобільні телефони, що можуть мати різна розміри екранів. Це завдання стає значно складнішим без використання спеціальних фреймворків і бібліотек. Сьогодні одним із найбільш поширених і популярних

інструментів для створення адаптивних веб-сторінок є Bootstrap – фреймворк, що дозволяє розробникам швидко створювати веб-сайти з адаптивним дизайном, використовуючи готові компоненти і стилі.

Такі фреймворки та бібліотеки забезпечують зручні можливості для автоматизації розмітки, надаючи користувачам вже готові блоки для створення окремих елементів сторінки. Проте, для того щоб система могла стати зручним інструментом для людей без досвіду розробки, необхідно ще більше спростити процес. Для цього використовується графічний інтерфейс, який дозволяє користувачу без необхідності писати код вибирати та додавати елементи веб-сторінки, редагувати їх стилі і налаштовувати адаптивність[2].

Використання різноманітних фреймворків у поєднанні з інтуїтивно зрозумілими інтерфейсами дозволяє створювати потужні інструменти для автоматизації розробки веб-сторінок – конструктори сайтів. Однак для досягнення повної автоматизації важливо також враховувати можливості інтелектуальних систем, які б дозволяли користувачам автоматично генерувати веб-сторінки на основі простих вказівок і без необхідності ручного налаштування всіх параметрів.

Автоматизація розробки веб-сторінок значно знижує поріг входу для користувачів, які хочуть створити власний сайт, але не мають знань програмування. Це дозволяє створювати веб-сторінки без потреби в глибокому розумінні HTML, CSS чи JavaScript. Крім того, автоматизація дає змогу значно зекономити час на розробку проекту, оскільки користувач може просто вибрати готові шаблони, налаштувати їх під свої вимоги і отримати функціональну веб-сторінку в найкоротші терміни.

Іншими важливими перевагами автоматизації є:

- використання готових шаблонів і компонентів дає змогу користувачам швидко створювати веб-сторінки, зменшуючи час на розробку;

- фреймворки типу Bootstrap автоматично генерують адаптивні сторінки, що дозволяє не витрачати час на налаштування відображення сторінки на різних пристроях;
- інтерфейси, що надають можливість налаштування без програмування, дозволяють користувачам зосередитись на контенті та дизайні, а не на коді;
- користувачі можуть додавати або видаляти елементи сторінки відповідно до своїх вимог і змінювати стилі та теми без складних налаштувань.

Для підвищення рівня автоматизації створення веб-сторінок важливо впроваджувати інтелектуальні системи, що надають можливість адаптації сторінки під вибір користувача. Такі системи здатні автоматично обирати найбільш підходящий стиль, кольорову схему або компоненти, виходячи з введених даних чи вибору користувача. Використання елементів експертних систем або алгоритмів машинного навчання для аналізу вимог користувача може значно спростити процес створення сторінки та зробити його ще більш інтуїтивно зрозумілим[3].

Зокрема, інтелектуальні системи можуть допомогти користувачу обрати стиль для веб-сторінки, запропонувавши кілька варіантів залежно від контексту або вибору користувача, а потім автоматично застосувати цей стиль до елементів сторінки. Це дозволяє не тільки заощадити час, але й забезпечити високу якість кінцевого продукту, оскільки система може застосувати кращі практики в дизайні та зручності користування.

Описуючи предметну область дослідження, варто також визначити загальні вимоги до систем автоматизації веб-розробки. Ці вимоги залежать від потреб користувачів і в подальшому мають бути використані для постановки задачі дослідження та проєктування системи. Важливо враховувати, що

користувачами системи в першу чергу є люди, що не мають достатнього досвіду веб-розробки для самостійного створення сторінок.

Системи автоматизації веб-розробки повинні забезпечувати зручний, інтуїтивно зрозумілий інтерфейс для користувачів різного рівня підготовки. Однією з основних вимог є високий рівень інтеграції з сучасними технологіями, що дозволяє автоматично генерувати оптимізовану кодову базу, знижуючи ризик виникнення помилок, пов’язаних з ручним кодуванням. Система повинна мати здатність адаптуватися до різних сценаріїв розробки, включаючи створення як простих веб-сторінок, так і більш складних інтерфейсів, використовуючи для цього сучасні фреймворки та бібліотеки. Зважаючи на динамічний розвиток веб-технологій, система також повинна надавати можливість оновлення або інтеграції нових інструментів і стандартів без необхідності значної зміни основної архітектури.

Інтерфейс користувача є важливим аспектом, оскільки він має спростити процес розробки для недосвідчених користувачів. Важливо, щоб система забезпечувала можливість швидкої візуалізації результату роботи, дозволяючи користувачу бачити зміни в реальному часі. Таким чином, автоматизація не лише оптимізує процес створення веб-сторінок, а й забезпечує стабільність і надійність системи в процесі роботи.

Також необхідно приділити увагу питанню кібербезпеки, оскільки зростання рівня автоматизації може призвести до збільшення кількості вразливостей, які можуть бути використані для атак на систему. При цьому, веб-додатки, до яких відноситься і розроблювана система, є потенційними цілями для зловмисників. Для забезпечення кібербезпеки в таких системах важливо впроваджувати вбудовані механізми перевірки та валідації всіх введених даних, щоб уникнути різноманітних атак. Крім того, важливо використовувати механізми автентифікації і авторизації для захисту доступу до ресурсів і конфіденційної інформації користувачів.

Детальніший опис класів та атрибутів предметної області подано в таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас	**Атрибут**	**Опис**
Сторінка	URL	Посилання на веб-сторінку, що генерується.
	Заголовок	Текст заголовку сторінки, який відображається у браузері.
	Контент	Основний вміст сторінки, що містить текстові або графічні дані.
Компонент	Тип	Тип компонента, наприклад, "заголовок", "кнопка", "параграф".
	Стилі	CSS-стилі, застосовані до компонента.
	Позиція	Місце розташування компонента на сторінці (наприклад, "верхня частина", "основний блок").
Система стилів	Шаблон	Набір стандартних стилів для зручного оформлення сторінки.
	Колірна схема	Основна кольорова палітра для компонентів.
	Адаптивність	Параметри для відображення на різних пристроях (наприклад, ПК, планшет, телефон).

Ця таблиця представляє основні класи в моделі предметної області для даного проекту. Кожен клас відповідає основній концепції в системі, наприклад «Сторінка», «Компонент» або «Система стилів», і включає в себе специфічні атрибути, які визначають властивості та функціональні можливості кожного класу.

Опис таблиці:

1. **Сторінка:** цей клас представляє веб-сторінку, яку згенерує система. Він має такі атрибути, як URL, який визначає посилання на згенеровану сторінку; Заголовок, який містить назву сторінки, що відображається в браузері; і Вміст, який містить основний вміст, наприклад текст або зображення.
2. **Компонент:** клас компонент включає окремі елементи інтерфейсу користувача, як-от заголовки, кнопки або абзаци, які складають сторінку. Такі атрибути, як «Тип», визначають роль компонента (наприклад, «кнопка» або «заголовок»), «Стилі» визначають застосовані стилі CSS, а «Позиція» вказує на розташування компонента на сторінці.
3. **Система стилів:** цей клас надає шаблони стилів для стандартизації візуального вигляду сторінок. Шаблон зберігає попередньо визначені колекції стилів, колірна схема визначає колірні палітри для компонентів, а адаптивність включає параметри для адаптації дизайну до різних пристроїв.

Кожен клас і атрибут працюють разом, щоб дозволити інтелектуальній системі динамічно створювати та стилізувати макети HTML, адаптовані до вподобань і вимог користувача, створюючи структурований і візуально згуртований веб-інтерфейс.

1.3 Огляд існуючих рішень

Розглянемо існуючі рішення предметної області.

Webflow — це універсальна платформа для веб-дизайну, яка дає змогу користувачам створювати візуально приголомшливі адаптивні веб-сайти без необхідності писати код. Він поєднує в собі легкість інструментів дизайну за допомогою перетягування та гнучкість детального налаштування, що робить

його ідеальним як для початківців, так і для досвідчених дизайнерів. Ця платформа дозволяє користувачам контролювати кожен аспект макета, стилю та функціональності свого сайту в зручному середовищі, подібному до таких інструментів, як Figma або Adobe Photoshop, але з додатковою перевагою створення фактичного коду HTML і CSS у фоновому режимі.

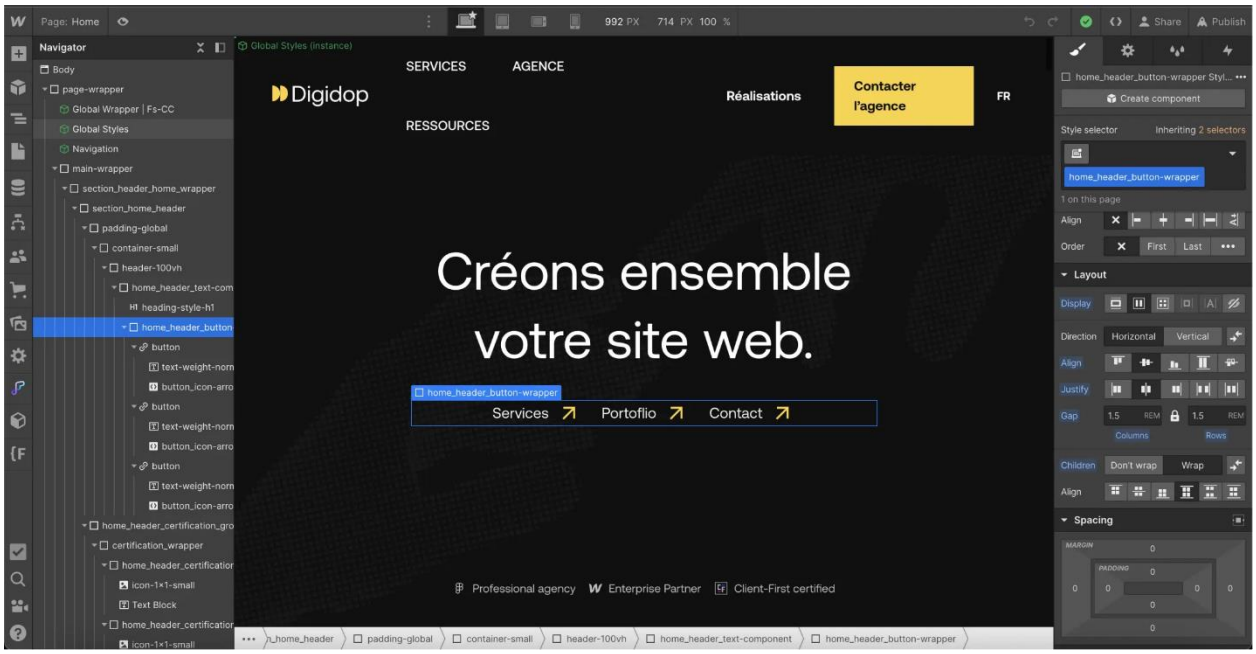


Рис. 1 Програмне забезпечення «Webflow»

Однією з видатних особливостей Webflow є його можливості адаптивного дизайну, які дозволяють користувачам автоматично налаштовувати свої веб-сайти для різних розмірів екрана, забезпечуючи безперебійну роботу на таких пристроях, як настільні ПК, планшети та смартфони. Ця функція усуває необхідність ручного написання медіа-запитів і полегшує створення веб-сайтів, зручних для мобільних пристроїв.

Платформа також містить вбудовану CMS (систему керування вмістом), яка дозволяє користувачам створювати власні типи вмісту та керувати динамічним вмістом безпосередньо на платформі. Завдяки цьому Webflow особливо добре підходить для веб-сайтів із великим вмістом, таких як блоги чи портфоліо, де важливі часті оновлення та керування вмістом. Крім того, Webflow надає інтуїтивно зрозумілий інтерфейс для інтеграції складної

анімації та взаємодії на веб-сайтах, що дозволяє користувачам легко створювати привабливі інтерактивні дизайни. Ці анімації можуть бути викликані такими діями користувача, як прокручування, наведення курсора або клацання, покращуючи загальну взаємодію з користувачем сайту.

Для користувачів, які хочуть швидко запускати свої веб-сайти, Webflow пропонує інтегрований хостинг і керування доменом. Після розробки сайту користувачі можуть опублікувати його безпосередньо на платформі Webflow, яка обслуговує всі аспекти хостингу, безпеки та оптимізації продуктивності. Ця функція спрощує процес розміщення веб-сайту в Інтернеті, заощаджуючи час і зусилля користувачів[4].

Для тих, хто цікавиться електронною комерцією, Webflow також надає інструменти для створення та керування онлайн-магазинами. Від настроюваних сторінок продуктів до інтегрованих кошиків для покупок і безпечної обробки платежів Webflow пропонує комплексне рішення для електронної комерції. Хоча він може бути не таким просунутим, як платформи на зразок Shopify, це чудовий варіант для малих і середніх онлайн-магазинів.

Webflow також містить потужні інструменти SEO, які дозволяють користувачам оптимізувати свої веб-сайти для пошукових систем. Такі функції, як настроювані мета-заголовки, описи та альтернативний текст для зображень, а також можливість керувати картами сайту та перенаправленнями, допомагають покращити видимість сайту в результатах пошукової системи.

Хоча Webflow пропонує безліч функцій і параметрів налаштування, він не позбавлений своїх обмежень. Ціни на платформі можуть бути вищими, ніж на інших конструкторах веб-сайтів, особливо для більш розширених функцій, таких як електронна комерція або сайти з високим трафіком. Крім того, незважаючи на те, що Webflow є зручним, крива навчання може бути крутішою, ніж простіші конструктори веб-сайтів, такі як Wix або Squarespace.

Він пропонує великий контроль над дизайном, але користувачам все одно потрібно інвестувати час у вивчення платформи, щоб повністю розкрити її потенціал.

Іншим обмеженням є відсутність у Webflow можливостей розробки серверної частини. Незважаючи на те, що він чудово підходить для дизайну інтерфейсу та керування вмістом, він не підтримує логіку на стороні сервера чи спеціальні бази даних, окрім своєї CMS, що може обмежити його використання для складніших програм. Так само, хоча він пропонує функції електронної комерції, він не такий надійний, як спеціалізовані платформи, такі як Shopify, які надають розширену функціональність для великих магазинів.

Незважаючи на ці проблеми, Webflow є чудовим рішенням для дизайнерів і компаній, які шукають платформу, яка забезпечує високий рівень контролю над дизайном, не вимагаючи навичок програмування. Він особливо добре підходить для створення портфоліо, цільових сторінок, невеликих сайтів електронної комерції або веб-сайтів, орієнтованих на вміст. Гнучкість платформи, простота використання та інтеграція з хостингом роблять її привабливим вибором для користувачів, які хочуть створити професійний веб-сайт, не покладаючись на розробників. Однак для великих, складніших проектів або налаштованих серверних рішень Webflow може не найкраще підійти.

Bootstrap Studio — це потужна настільна програма, розроблена, щоб допомогти користувачам створювати адаптивні веб-сайти за допомогою фреймворку Bootstrap. Він пропонує інтерфейс перетягування, що робить його доступним як для початківців, так і для досвідчених веб-дизайнерів. Завдяки своїм інтуїтивно зрозумілим інструментам дизайну Bootstrap Studio дозволяє користувачам створювати складні адаптивні веб-сторінки без необхідності писати великий код, хоча він все ще надає можливості для розробників, які бажають додатково налаштувати свій дизайн за допомогою HTML, CSS і JavaScript[5].

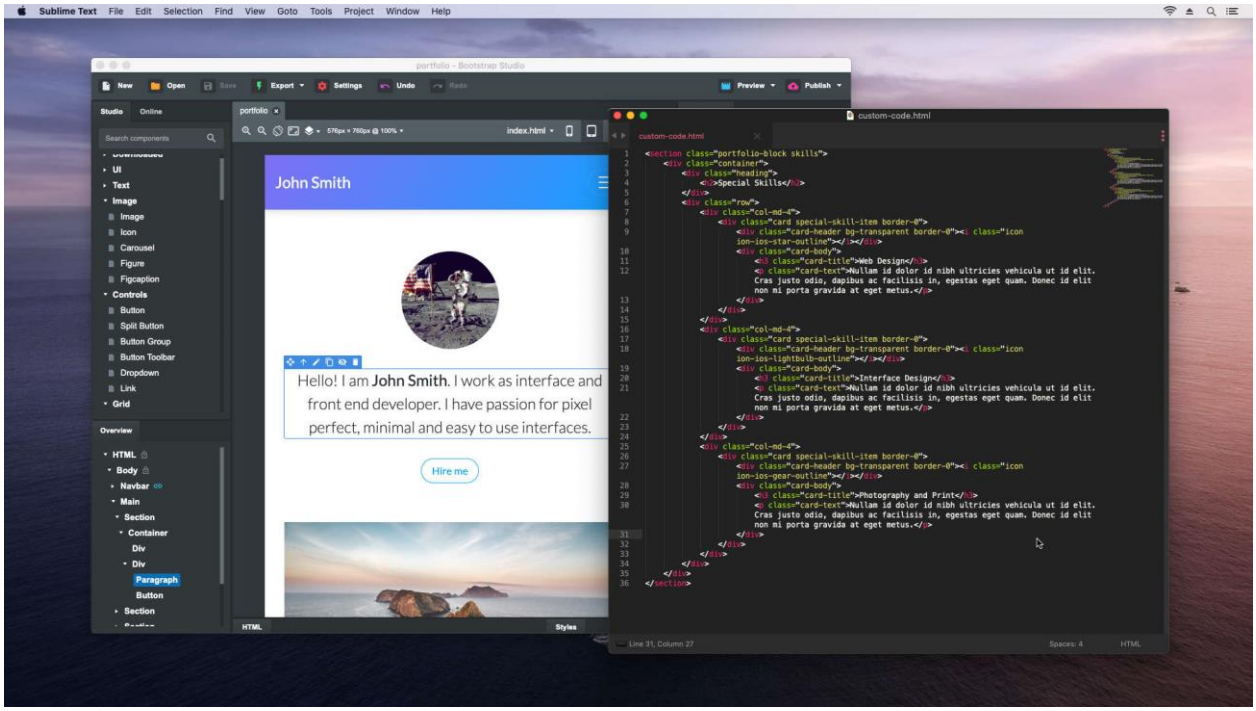


Рис. 2 Програмне забезпечення «Bootstrap Studio»

Однією з ключових сильних сторін Bootstrap Studio є її повна інтеграція з системою сітки Bootstrap. Платформа надає багатий набір готових компонентів і макетів, які базуються на фреймворку Bootstrap, що гарантує, що веб-сайти, створені за допомогою Bootstrap Studio, зручні для мобільних пристроїв і оптимізовані для різних розмірів екранів. Ця функція усуває потребу в ручних налаштуваннях, роблячи адаптивний дизайн простим і швидким.

Функція перетягування Bootstrap Studio дозволяє легко розміщувати такі елементи, як верхні та нижні колонтитули, панелі навігації, форми та кнопки на веб-сторінці. Користувачі також можуть точно налаштувати властивості цих компонентів, такі як інтервали, колір, типографіка та вирівнювання, за допомогою візуального редактора. Додаток також містить функцію попереднього перегляду в реальному часі, що дозволяє користувачам бачити, як виглядають їхні проекти в режимі реального часу, коли вони вносять зміни, що прискорює процес розробки.

Програмне забезпечення включає надійний редактор коду для користувачів, які віддають перевагу написанню власних HTML, CSS і JavaScript. Це забезпечує більшу гнучкість і можливість тонкого налаштування дизайну за межами інтерфейсу перетягування. Крім того, Bootstrap Studio надає розширений редактор CSS із такими функціями, як живе редагування CSS, що дозволяє користувачам бачити миттєві зміни в попередньому перегляді під час редагування стилів.

Ще одна примітна особливість Bootstrap Studio — це вбудована підтримка створення динамічних інтерактивних сторінок. Користувачі можуть додавати власні анімації та переходи до елементів, покращуючи взаємодію з користувачем. Додаток також підтримує JavaScript, що дозволяє розробникам реалізувати більш складні функції та взаємодії[5].

Для організації проекту Bootstrap Studio містить зручний робочий простір, де користувачі можуть керувати кількома сторінками, ресурсами та компонентами в одному проекті. Він також підтримує експорт дизайнів у чистий, багаторазово використовуваний код, який повністю сумісний із будь-якою службою веб-хостингу, що робить перехід від дизайну до розгортання плавним.

Хоча Bootstrap Studio є настільною програмою, вона надає такі функції, як хмарна синхронізація, що дозволяє користувачам отримувати доступ до проектів і працювати над ними з різних пристроїв. Програма також інтегрується зі сторонніми ресурсами дизайну, що полегшує імпорт піктограм, зображень та інших ресурсів безпосередньо у ваш проект.

Незважаючи на широкі можливості, Bootstrap Studio має кілька обмежень. Наприклад, хоча він чудово підходить для створення статичних веб-сайтів і прототипів, йому не вистачає вбудованих інструментів для більш складної веб-розробки, наприклад програмування на стороні сервера або інтеграції бази даних. Крім того, незважаючи на те, що він заснований на

Bootstrap, він може бути не таким гнучким, як спеціальні фреймворки або інші конструктори веб-сайтів, які пропонують більше свободи в дизайні та функціональності.

Загалом, Bootstrap Studio є чудовим інструментом для тих, хто хоче швидко створювати красиві, адаптивні веб-сайти за допомогою фреймворку Bootstrap. Його інтерфейс із можливістю перетягування та потужні інструменти дизайну роблять його сильним вибором для дизайнерів, які хочуть оптимізувати процес веб-розробки, а його можливості редагування коду забезпечують гнучкість для розробників. Однак для більш складних динамічних веб-додатків користувачам може знадобитися інтегрувати інші технології або фреймворки разом із Bootstrap Studio.

1.4 Постановка завдання

Основною проблемою при розробці інтелектуальної системи для створення розмітки HTML для інтерфейсів веб-сторінок є складність, пов’язана з автоматизацією процесу проектування, одночасно забезпечуючи гнучкість, узгодженість і швидкість реагування на різних платформах і розмірах екрана. Сучасні системи часто вимагають ручного втручання або покладаються на жорсткі шаблони, що може обмежити творчий контроль і призвести до неузгодженості взаємодії з користувачем.

Дизайн веб-сторінки за своєю суттю є складним через величезну кількість змінних, які необхідно брати до уваги — наприклад, розмір екрана, орієнтація пристрою, уподобання користувача та умови навколишнього середовища тощо. Досягнення оптимальних інтерфейсів користувача вимагає не лише технічних можливостей; це вимагає здатності адаптуватися та реагувати на ці різноманітні фактори в реальному часі.

Основна проблема, з якою сьогодні стикаються веб-розробники, — це нездатність швидко створити динамічну, адаптовану та естетично привабливу

розмітку HTML, узгоджену в різних середовищах. Наприклад, під час проектування для мобільних пристроїв, планшетів і настільних пристроїв може бути важко забезпечити правильну адаптацію макета без втрати функціональності чи зручності для користувача. Крім того, у сучасних інструментах проектування часто відсутні інтелектуальні алгоритми, які можуть аналізувати та коригувати макет автоматично на основі введення користувача або контексту.

Крім того, зростання складності веб-додатків з їх інтерактивними елементами та різноманітним вмістом робить традиційні методи ручного створення розмітки HTML недостатніми. Автоматизовані інструменти мають виходити за рамки простого створення макета та включати розширені функції, такі як адаптація даних у реальному часі, оптимізація для доступності та інтеграція з серверними системами. Це вимагатиме від системи виконання таких завдань, як автоматичний стиль, адаптивне коригування макета та динамічне оновлення вмісту, зберігаючи при цьому високу продуктивність і надійність.

Щоб вирішити ці проблеми, потрібна інтелектуальна система, яка не тільки автоматично генерує HTML-розмітку, але й оптимізує її на основі потреб користувачів, специфікацій платформи та відгуків у реальному часі. Ця система оптимізує процес проектування, зробивши його більш ефективним і доступним, особливо для нетехнічних користувачів, і гарантує, що кінцеві веб-сторінки будуть не тільки функціональними, але й привабливими та інтуїтивно зрозумілими для користувачів на всіх пристроях.

Щоб подолати описані вище проблеми, інтелектуальна система створення HTML-розмітки повинна містити наступні ключові функції:

- система повинна автоматично генерувати чистий, добре структурований HTML-код на основі введених користувачем даних, специфікацій дизайну та вибраних шаблонів;

- система має гарантувати, що згенеровані веб-сторінки повністю реагують, плавно налаштовуючи їх макет і елементи на різних розмірах екрана та на різних пристроях;
- система повинна мати можливість інтегрувати динамічні дані в розмітку HTML;
- система повинна генерувати HTML, який є не тільки функціональним, але й доступним, забезпечуючи дотримання стандартів веб-доступності.

1.5 Функціональні та нефункціональні вимоги

Функціональні та нефункціональні вимоги є важливими аспектами розробки програмного забезпечення, які визначають, що повинна робити система і як вона повинна виконувати ці функції[14].

Функціональні вимоги

1. Система повинна бути здатна автоматично генерувати чистий, дійсний і семантичний HTML-код на основі введених користувачем або попередньо визначених шаблонів;
2. Система повинна підтримувати створення основних структур HTML, таких як заголовки, абзаци, форми, списки, таблиці та посилання;
3. Система повинна автоматично генерувати адаптивні макети, які налаштовуються залежно від розміру екрана або типу пристрою (наприклад, мобільний, планшет, настільний комп’ютер);
4. Користувач повинен мати можливість попередньо переглянути, як згенерований HTML виглядатиме на різних пристроях і в різних орієнтаціях;

5. Користувачі повинні мати можливість налаштовувати згенерований HTML, наприклад вибирати колірні схеми, типографіку, інтервали та інші властивості, пов’язані зі стилем;
6. Система має запропонувати інтуїтивно зрозумілий інтерфейс користувача (UI) для цих налаштувань, не вимагаючи від користувачів писати код CSS або HTML вручну;
7. Система повинна мати можливість інтегрувати динамічний вміст у створений HTML. Це включає зв’язування даних із зовнішніх джерел, таких як API або бази даних;
8. Система має гарантувати, що весь створений HTML є доступним відповідно до стандартів веб-доступності (WCAG);
9. Система повинна дозволяти користувачам переглядати згенерований HTML в інтерфейсі, схожому на браузер, перед експортом;
10. Користувачі повинні мати можливість експортувати згенеровану розмітку як окремі файли HTML або інтегрувати її з іншими інструментами, такими як системи керування вмістом (CMS) або зовнішні фреймворки;
11. Система повинна підтримувати попередньо визначені шаблони для різних типів веб-сторінок (наприклад, домашніх сторінок, сторінок продуктів, цільових сторінок);
12. Користувачі повинні мати можливість вибрати шаблон і змінити його або створити новий шаблон з нуля;
13. Система повинна забезпечувати зворотний зв’язок з користувачами, пропонуючи пропозиції щодо покращення HTML-коду, наприклад, виправлення проблем із доступністю чи оптимізації продуктивності.

Нефункціональні вимоги

1. Система має бути швидкою та чуйною, здатною генерувати складні HTML-сторінки в реальному часі без помітних затримок.
2. Система повинна бути масштабованою, щоб відповідати різним рівням складності веб-сторінок, від простих статичних сторінок до динамічних додатків, керованих даними;
3. Система повинна мати інтуїтивно зрозумілий і зручний інтерфейс, який вимагає мінімальних технічних знань для використання;
4. Система повинна гарантувати, що згенерований HTML-код відповідає найкращим практикам безпеки, щоб запобігти типовим уразливостям, таким як XSS (міжсайтовий сценарій) і впровадження SQL, під час інтеграції із зовнішніми даними;
5. Кодова база системи має бути модульною та добре структурованою, що дозволяє легко оновлювати, виправляти помилки та додавати функції;
6. Згенерований HTML-код має бути сумісним з усіма сучасними веб-браузерами (наприклад, Chrome, Firefox, Safari, Edge) і мати зворотну сумісність зі старими версіями цих браузерів, де це можливо;
7. Система також повинна підтримувати сумісність із популярними системами керування контентом (CMS) і фреймворками (наприклад, WordPress, Drupal, React, Vue.js);
8. Система повинна підтримувати кілька мов і регіональних налаштувань для глобальної бази користувачів, що дозволяє користувачам працювати мовою, яку вони вибирають.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Загальні відомості про UML та інтелектуальний аналіз даних

Уніфікована мова моделювання (UML) — це стандартизована візуальна мова, яка відіграє вирішальну роль у розробці програмного забезпечення. Це дозволяє специфікувати, візуалізувати, конструювати та документувати артефакти програмної системи. UML надає широкий спектр діаграм, кожна з яких пропонує різний погляд на систему, будь то структурна, поведінкова або взаємодіюча. Це дає змогу розробникам, дизайнерам і зацікавленим сторонам розуміти, спілкуватися та співпрацювати над проектами складних систем.

Як незалежну від платформи мову, UML можна застосовувати до програмних систем, розроблених за допомогою будь-якої мови програмування або стеку технологій, що робить її універсальною та гнучкою. Він допомагає як на етапі проектування, так і на етапі реалізації проекту, пропонуючи повний набір інструментів для моделювання різних аспектів архітектури системи.

UML поділяється на дві основні категорії діаграм: структурні та поведінкові. Структурні діаграми зосереджуються на статичних аспектах системи, таких як її компоненти, класи та об’єкти. Ці діаграми моделюють структуру системи та зв’язки між її елементами. Наприклад, діаграми класів використовуються для представлення зв’язків між класами та їхніми атрибутами, тоді як діаграми об’єктів фіксують екземпляри цих об’єктів у певний момент часу, показуючи їхні зв’язки та атрибути. Інші структурні діаграми, такі як діаграми компонентів і розгортання, описують фізичні компоненти та апаратну конфігурацію системи відповідно.

З іншого боку, діаграми поведінки зосереджені на динамічних аспектах системи. Вони ілюструють, як взаємодіють об’єкти, як перетікають дані та як система поводиться з часом. Діаграми варіантів використання представляють функціональні вимоги з точки зору користувачів або «акторів», тоді як діаграми послідовності показують взаємодію між об’єктами в часовій послідовності. Діаграми діяльності представляють робочі процеси, подібні до блок-схем, тоді як діаграми станів моделюють переходи між різними станами об’єкта. Діаграми зв’язку, як і діаграми послідовності, показують взаємодії, але підкреслюють зв’язки між об’єктами в системі[6].

На додаток до цих основних типів діаграм, UML також включає діаграми взаємодії, діаграми складеної структури та часові діаграми. Діаграми взаємодії далі розбивають зв’язок між об’єктами, виділяючи послідовність обміну повідомленнями. Складені структурні діаграми показують внутрішню структуру класу, показуючи, як взаємодіють його компоненти, тоді як часові діаграми моделюють поведінку об’єктів у реальному часі, що корисно для систем, де час і зміни стану є вирішальними.

UML має численні практичні застосування в процесі розробки програмного забезпечення. Він служить цінним інструментом для проектування системи, допомагаючи визначити архітектуру системи від моделей високого рівня до взаємодії конкретних компонентів. Це також допомагає в аналізі вимог, уточнюючи функціональні та нефункціональні потреби. Крім того, UML ефективний для документування, дозволяючи створювати візуальні представлення, які спрощують спілкування між розробниками, аналітиками та зацікавленими сторонами.

Крім того, деякі інструменти UML можуть генерувати код із діаграм, прискорюючи перехід від проектування до розробки. UML забезпечує спільну мову, яка покращує спілкування між командами, сприяючи спільному розумінню дизайну та поведінки системи. Це особливо важливо у великих складних проектах, де задіяно кілька команд.

Переваги UML значні. Він пропонує стандартизований підхід до моделювання систем, забезпечуючи узгодженість і ясність документації та спілкування. Його візуальний характер полегшує розуміння структури та поведінки системи навіть тим, хто не залучений до технічних аспектів розробки. Крім того, UML можна адаптувати до різних методологій, що робить його придатним для широкого кола проектів. Наявність численних інструментів для моделювання UML, включаючи комерційні та з відкритим вихідним кодом, забезпечує гнучкість застосування та підтримки UML.

UML є незамінним інструментом у життєвому циклі розробки програмного забезпечення. Пропонуючи різноманітні діаграми, які відображають як статичний, так і динамічний вигляд системи, він допомагає прояснити дизайн, структуру та поведінку складного програмного забезпечення. Його здатність підтримувати кращий зв’язок і співпрацю між розробниками, зацікавленими сторонами та командами має вирішальне значення для успіху проектів програмного забезпечення.

Інтелектуальний аналіз даних – це процес виявлення закономірностей, кореляцій, тенденцій і корисної інформації з великих наборів даних за допомогою різних методів, таких як статистичний аналіз, машинне навчання та системи баз даних. Це важлива галузь у науці про дані та відіграє вирішальну роль у перетворенні необроблених даних у цінні ідеї, допомагаючи в прийнятті рішень у багатьох галузях, включаючи фінанси, охорону здоров’я, маркетинг та електронну комерцію.

За своєю суттю інтелектуальний аналіз даних включає вилучення прихованих знань із моря даних, осмислення їх і перетворення цієї інформації в практичні ідеї. Процес інтелектуального аналізу даних зазвичай складається з кількох етапів, включаючи збір даних, попередню обробку даних, побудову моделі, оцінку шаблону та інтерпретацію результатів. Якість отриманої інформації значною мірою залежить від того, наскільки ефективно виконуються ці кроки.

Одним із ключових аспектів інтелектуального аналізу даних є його здатність визначати закономірності та зв’язки у великих наборах даних, які було б важко або неможливо розкрити за допомогою традиційних методів аналізу. Це може включати пошук кореляції між різними змінними, виявлення тенденцій у часі, сегментацію наборів даних у значущі кластери та навіть прогнозування майбутніх результатів на основі історичних даних.

Методи інтелектуального аналізу даних різноманітні та включають підходи до навчання під наглядом і без нагляду. Контрольоване навчання передбачає навчання моделі на позначеному наборі даних, де результат відомий, щоб передбачити майбутні результати або класифікації. Наприклад, під час виявлення шахрайства з кредитними картками моделі навчання під наглядом можна навчити на історичних даних транзакцій, щоб ідентифікувати моделі, пов’язані з шахрайською діяльністю.

Навпаки, методи неконтрольованого навчання працюють з немаркованими даними, шукаючи властиві їм структури або групи. Наприклад, такі методи, як кластеризація, групують схожі точки даних, що може бути корисним для сегментації клієнтів на основі купівельної поведінки або виявлення ненормальних моделей для виявлення аномалій.

Деякі поширені методи, які використовуються в інтелектуальному аналізі даних, включають класифікацію, регресію, кластеризацію, аналіз правил асоціації та виявлення аномалій. Класифікація — це процес класифікації даних у заздалегідь визначені класи, тоді як регресія зосереджена на прогнозуванні числових результатів. Кластеризація, як згадувалося раніше, групує подібні точки даних, тоді як аналіз правил асоціації розкриває зв’язки між змінними у великих наборах даних, наприклад «клієнти, які купили X, також купили Y». Виявлення аномалій визначає викиди або незвичайні точки даних, які суттєво відрізняються від більшості набору даних.

Застосування методів аналізу даних різниться залежно від цілей аналізу та типу даних, що обробляються. У сфері бізнесу інтелектуальний аналіз даних часто використовується для сегментації клієнтів, прогнозування продажів, аналізу ринкового кошика та прогнозування відтоку. В охороні здоров’я інтелектуальний аналіз даних застосовується до записів пацієнтів, щоб визначити моделі спалахів захворювань, ефективність лікування та взаємодію ліків. У фінансах це допомагає виявляти шахрайські операції, оцінювати кредитний ризик і оптимізувати інвестиційні стратегії.

Незважаючи на численні переваги, інтелектуальний аналіз даних також створює труднощі, особливо при роботі з великими складними наборами даних. Якість даних є критичним фактором; шумні, неповні або упереджені дані можуть призвести до оманливих результатів. Важливо також забезпечити використання правильних моделей і алгоритмів для інтерпретації даних. Крім того, під час роботи з конфіденційними даними, особливо в таких секторах, як охорона здоров’я та фінанси, важливі міркування щодо конфіденційності та етики.

Загалом інтелектуальний аналіз даних є потужним інструментом для виявлення цінної інформації зі складних наборів даних, сприяння прийняттю кращих рішень і стимулювання інновацій у галузях. Використовуючи передові алгоритми та методи, організації можуть отримати конкурентну перевагу, підвищити операційну ефективність і пропонувати персоналізовані послуги своїм клієнтам.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це візуальне представлення взаємодії між користувачами (або «акторами») і системою, яке показує, як система використовуватиметься для досягнення конкретних цілей або функцій. Він є частиною Уніфікованої мови

моделювання (UML) і служить важливим інструментом у проектуванні системи та аналізі вимог. Діаграми варіантів використання допомагають зацікавленим сторонам, таким як розробники, клієнти та кінцеві користувачі, зрозуміти, як система поводитиметься в різних сценаріях, і надають чітке уявлення про функціональні можливості системи з точки зору користувача[11].

На діаграмі варіантів використання актори представляють зовнішні сутності, які взаємодіють із системою. Це можуть бути окремі особи (наприклад, користувачі, адміністратори або клієнти), інші системи або апаратні пристрої. Варіанти використання — це конкретні завдання або операції, які система виконує у відповідь на введення актора. Кожен варіант використання представляє одну функцію, ціль або частину функціональності в системі.

Діаграми варіантів використання зазвичай структуровані таким чином:

- **актори:** вони показані як фігурки за межами системи. Вони представляють будь-кого або щось, що взаємодіє з системою. Наприклад, у системі онлайн-магазину учасниками можуть бути «Клієнт», «Адміністратор» або «Платіжний шлюз»;
- **випадки використання:** це овальні форми всередині межі системи. Кожен варіант використання представляє функцію,

яку виконує система. Наприклад, «Увійти», «Розмістити замовлення» або «Переглянути профіль».

Спроєктована діаграма прецедентів представлена на рис.3.

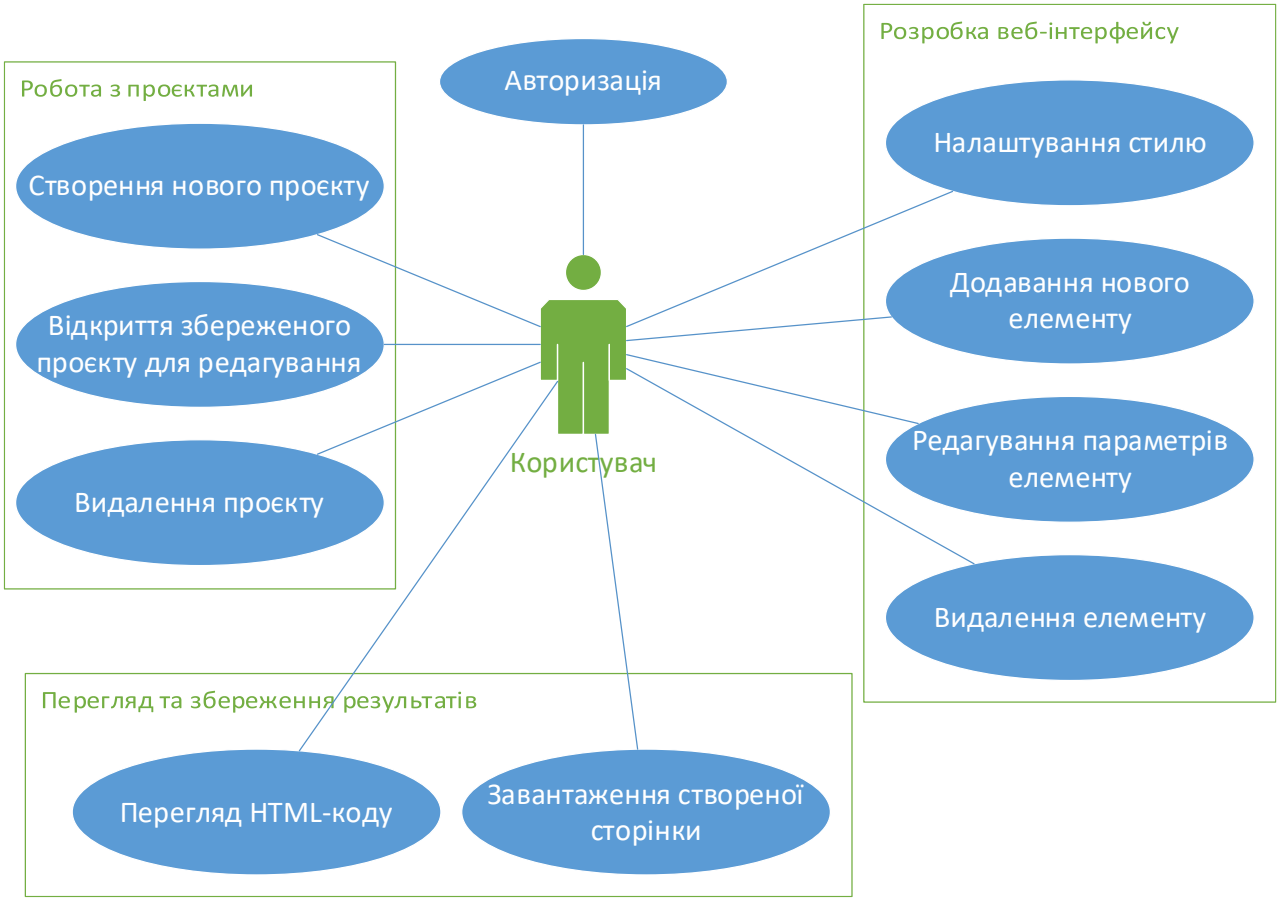


Рис. 3 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Користувач”.

Актор «Користувач» включає такі прецеденти:

- авторизація;
- створення нового проекту;
- відкриття збереженого проекту для редагування;
- видалення проекту;
- перегляд HTML-коду;
- завантаження сторінки;
- налаштування стилю;

- додання нового елементу;
- редагування параметрів елементу;
- видалення елементу.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Випадок використання: створення нового проекту.

Актор: користувач.

Опис: користувач ініціює процес створення нового проекту в інтелектуальній системі, призначеній для створення HTML-розмітки для інтерфейсів веб-сторінок. Система допомагає користувачеві налаштувати новий проект, визначити основні параметри проекту та підготувати необхідне середовище для створення HTML-контенту. Цей варіант використання включає в себе керівництво користувача послідовністю кроків для налаштування нового проекту та початку розробки інтерфейсу.

Передумови:

Користувач авторизується в системі. Система працює і готова до створення нового проекту.

Основний потік:

1. Користувач натискає кнопку «Створити новий проект» або пункт меню на головній інформаційній панелі системи або в інтерфейсі керування проектом;
2. Система запропонує користувачеві за допомогою діалогового вікна або набору форм заповнити важливі деталі для нового проекту;
3. Користувач надає унікальну назву для нового проекту;
4. Користувач вибирає тип проекту (наприклад, веб-сторінка, цільова сторінка, блог тощо);
5. Користувач може вибрати технології або фреймворки для використання, наприклад HTML5, CSS3, JavaScript, Bootstrap або спеціальні фреймворки;

6. Якщо застосовно, користувач може вибрати шаблон або базовий макет для проекту;
7. Користувач визначає, чи має веб-сторінка бути адаптивною (мобільний, планшет, комп’ютер);
8. Користувач може визначити параметри макета;
9. Користувач може вибрати стиль навігації (наприклад, верхня панель, бічне меню, меню гамбургерів);
10. Система може запропонувати користувачеві вибрати додаткові функції;
11. Після того, як користувач заповнив необхідну інформацію, система надає попередній перегляд конфігурації проекту. Користувач може підтвердити правильність налаштувань або внести корективи;
12. Після підтвердження налаштувань користувач натискає на кнопку «Створити проект»;
13. Створює новий запис про проект у системі;
14. Налаштування відповідної структури каталогів (наприклад, створення файлів HTML, CSS, JS);
15. Зберігає вибрані конфігурації в базі даних системи;
16. Система сповіщає користувача про успішне створення проекту. Потім користувач перенаправляється до робочої області проекту, де він може почати створювати та редагувати розмітку HTML для інтерфейсу своєї веб-сторінки.

Альтернативний потік:

1. Якщо користувач надає неповні або недійсні дані (наприклад, повторювана назва проекту або недійсний тип проекту), система відображає повідомлення про помилку та просить користувача виправити інформацію;

2. На будь-якому кроці користувач може вирішити скасувати процес створення проекту. У цьому випадку система повертає користувача на попередній екран без збереження даних.

Цей варіант використання описує процес створення нового проекту в інтелектуальній системі. Він наголошує на взаємодії з користувачем, направляючи користувача через основні етапи налаштування та гарантуючи, що всі відповідні конфігурації виконано перед створенням HTML-розмітки для інтерфейсу веб-сторінки.

Випадок використання: налаштування стилю.

Актор: користувач.

Опис: користувач налаштовує візуальний стиль і аспекти дизайну веб-сторінки, встановлюючи стилі для таких елементів, як шрифти, кольори, поля, інтервали та макет. Інтелектуальна система забезпечує інтуїтивно зрозумілий інтерфейс для зміни властивостей CSS, що дозволяє користувачеві застосовувати послідовний і візуально привабливий стиль до розмітки HTML. Мета цього варіанту використання полягає в тому, щоб користувач міг легко визначити зовнішній вигляд сторінки, зберігаючи при цьому швидкість реагування та зручність використання.

Передумови: користувач успішно створив проект (через варіант використання «Створити новий проект»). Користувач знаходиться в робочій області проекту і готовий застосовувати стилі до елементів сторінки. Система здатна редагувати та застосовувати CSS до структури HTML.

Основний потік:

1. Користувач переходить до розділу налаштування стилю інтерфейсу проекту. Це може бути окрема вкладка «Стиль» або кнопка з написом «Установити стилі» чи «Дизайн» у робочій області проекту;

2. Система представляє список або візуальне представлення структури HTML проекту, наприклад заголовки, абзаци, зображення, кнопки та контейнери.;
3. Користувач вибирає елементи, які хоче стилізувати;
4. Коли користувач вибирає елемент для стилізації, система відображає доступні властивості CSS, які можна змінити;
5. Користувач може налаштувати властивості шрифту, кольори, інтервал, межі;
6. Після налаштування бажаних властивостей CSS користувач натискає кнопку «Застосувати» або «Зберегти», щоб застосувати вибрані стилі до елементів сторінки. Потім система оновлює структуру HTML за допомогою відповідних правил CSS, які або вбудовані, або пов’язані з файлом HTML;
7. Система негайно виконує попередній перегляд стилізованої сторінки у призначеному вікні попереднього перегляду. Користувач може взаємодіяти зі сторінкою, щоб побачити, як виглядають зміни в режимі реального часу. Користувач може додатково налаштувати стилі на основі візуального зворотного зв’язку;
8. Якщо користувача не влаштовують застосовані стилі, він може внести додаткові зміни. Цей крок повторюється, доки користувач не буде задоволений зовнішнім виглядом сторінки;
9. Коли користувач задоволений стилями, він зберігає налаштування, а система оновлює файл CSS проекту, щоб постійно зберігати зміни. Система може повідомити користувача про те, що стилі успішно застосовані та збережені;
10. Нові стилі тепер інтегровані з HTML-розміткою проекту. Користувач може переходити до інших завдань, таких як редагування вмісту або коригування макета, знаючи, що стиль сторінки готовий.

Альтернативний потік:

1. Якщо користувач вводить неправильні або несумісні властивості CSS (наприклад, неіснуючий код кольору або недійсний розмір шрифту), система виділяє недійсний введений текст і пропонує виправлення або автоматично скасовує зміни;
2. У будь-який момент користувач може вирішити скасувати процес налаштування стилю. У цьому випадку система скасовує будь-які незбережені зміни та повертає користувача до попереднього екрана без зміни проекту.

Цей варіант використання описує, як користувач взаємодіє з системою, щоб налаштувати та застосувати стилі до елементів інтерфейсу веб-сторінки. Система дозволяє легко модифікувати та отримати зворотний зв'язок у режимі реального часу, завдяки чому користувачі можуть інтуїтивно зрозуміло створювати візуально привабливі та функціональні веб-сторінки.

2.2.2 Діаграма послідовності. Діаграма послідовності — це тип діаграми Уніфікованої мови моделювання (UML), яка ілюструє взаємодію між різними компонентами або об'єктами в системі, підкреслюючи порядок, у якому обмінюються повідомленнями. Він відіграє вирішальну роль у візуалізації того, як різні суб'єкти співпрацюють, і послідовності дій з часом.

На діаграмі послідовності актори представляють зовнішні сутності, такі як користувачі, зовнішні системи або інші компоненти, які взаємодіють із системою. У цих взаємодіях беруть участь об'єкти, які є компонентами системи (наприклад, класи чи екземпляри). Лінії життя — це вертикальні пунктирні лінії, які вказують на присутність об'єкта протягом усієї взаємодії та зображують проміжок часу, протягом якого об'єкт задіяний. Повідомлення між цими об'єктами представлено стрілками, що показує, як вони спілкуються один з одним. Ці повідомлення можуть бути або

синхронними, коли відправник очікує на відповідь, або асинхронними, коли відправник не чекає на відповідь. Панелі активації — це горизонтальні панелі, які показують, коли об’єкт активно обробляє повідомлення. Зворотні повідомлення часто зображуються пунктирними лініями, що показують відповідь або повернення до вихідного повідомлення[15].

Діаграми послідовності служать багатьом цілям. Вони допомагають візуалізувати взаємодію всередині системи, забезпечують детальне уявлення про потік виконання для конкретних випадків використання та пояснюють поведінку системи, показуючи, як різні компоненти співпрацюють для досягнення мети. Ці діаграми особливо корисні на етапі проектування, оскільки вони показують, як компоненти взаємодіють один з одним і які повідомлення передаються. Вони також можуть допомогти у вирішенні проблем, відстежуючи послідовність подій, які призвели до проблеми чи помилки. Крім того, діаграми послідовності є цінними інструментами документування, оскільки вони пропонують стисле та зрозуміле уявлення про те, як функції чи взаємодії працюють у системі.

Діаграма послідовностей, зображена на рис. 4, містить такі об’єкти:

- user;
- system interface;
- style setup panel live preview;
- CSS handler.

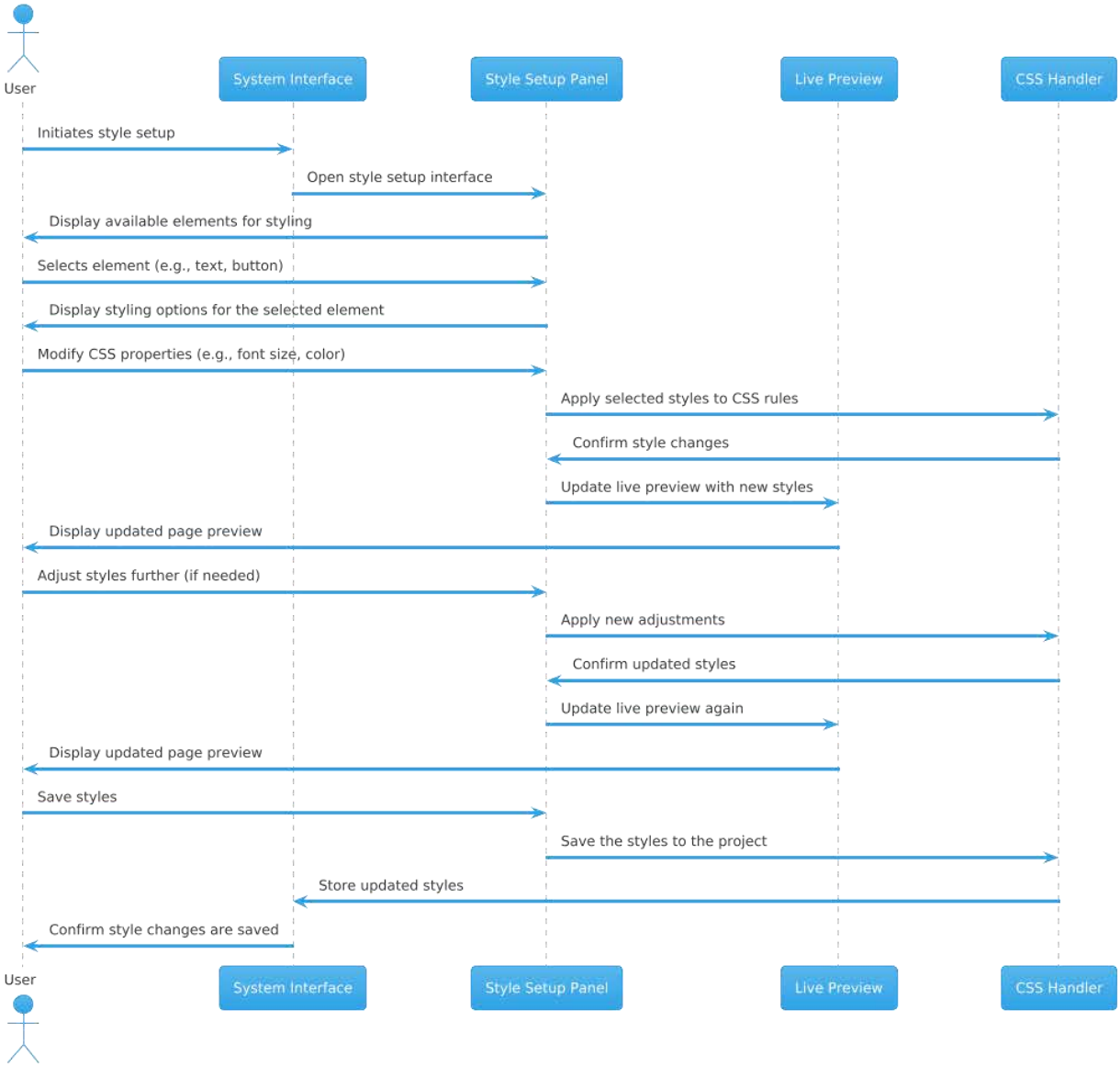


Рис. 4 Діаграма послідовності

Ця діаграма послідовності UML описує процес налаштування та застосування стилів до інтерфейсу веб-сторінки за допомогою інтелектуальної системи. Головною дійовою особою на діаграмі є користувач, який взаємодіє з системою, щоб налаштувати та переглянути стилі для веб-сторінки.

Взаємодія починається, коли користувач ініціює налаштування стилю, взаємодіючі з системним інтерфейсом. Система у відповідь відкриває панель налаштування стилю, яка надає користувачеві доступні елементи для стилізації, наприклад текст або кнопки. Коли користувач

вибирає елемент для стилізації, на панелі відображаються параметри стилю, характерні для вибраного елемента[16].

Користувач змінює властивості CSS (наприклад, змінює розмір шрифту, колір) за допомогою панелі налаштування стилю. Потім панель зв’язується з обробником CSS, щоб застосувати вибрані стилі до правил CSS, пов’язаних з елементом. Після застосування стилів обробник CSS підтверджує зміни на панелі.

Потім панель надсилає оновлені стилі до компонента Live Preview, який виконує попередній перегляд оновленої сторінки для перегляду користувачем. Якщо потрібні додаткові коригування, користувач може внести додаткові зміни через панель, які потім надсилаються обробнику CSS для оновлення правил CSS. Процес повторюється, при цьому попередній перегляд оновлюється кожного разу, щоб відобразити останні зміни.

Коли користувач буде задоволений стилями, він дає команду Панелі зберегти зміни. Після цього Panel зберігає оновлені стилі в проекті, надсилаючи їх обробнику CSS. Обробник CSS підтверджує успішне збереження стилів, а Система зберігає оновлені стилі. Система підтверджує Користувачеві, що зміни стилю були успішно збережені.

2.2.3 Діаграма активності. Діаграма діяльності — це тип діаграми UML (Unified Modeling Language), який використовується для представлення робочих процесів, процесів і послідовності дій у системі. Він візуально фіксує динамічні аспекти системи, показуючи, як діяльність перетікає від однієї до іншої, а також висвітлює рішення, умови та паралельні процеси. Ці діаграми є цінними для ілюстрації поведінки

системи, зосереджуючись на тому, як різні компоненти або актори взаємодіють для виконання завдань або операцій.

Основна мета діаграми діяльності полягає в моделюванні робочих процесів і серії кроків, залучених до певного процесу. Він може представляти широкий діапазон дій, від простих завдань до більш складних послідовностей. Використовуючи цю діаграму, розробники та аналітики можуть краще зрозуміти, як завдання взаємопов’язані, допомагаючи визначити вузькі місця або області, які потрібно вдосконалити в робочому процесі. У проектуванні системи діаграми активності часто використовуються для визначення поведінки системи, показуючи дії, які виконуються у відповідь на конкретні вхідні дані, рішення або умови.

З точки зору структури, діаграма діяльності зазвичай включає кілька ключових елементів. Діяльність є основними компонентами діаграми та зображується як округлені прямокутники, що представляють завдання або операції в системі. Початковий вузол, суцільне заповнене коло, позначає початок потоку діяльності, тоді як кінцевий вузол, коло з суцільною рамкою, вказує на завершення процесу. Потік керування або даних між діями представлено стрілками, які направляють користувача через послідовність кроків[17].

Розроблена діаграма активності представлена на рис.5.

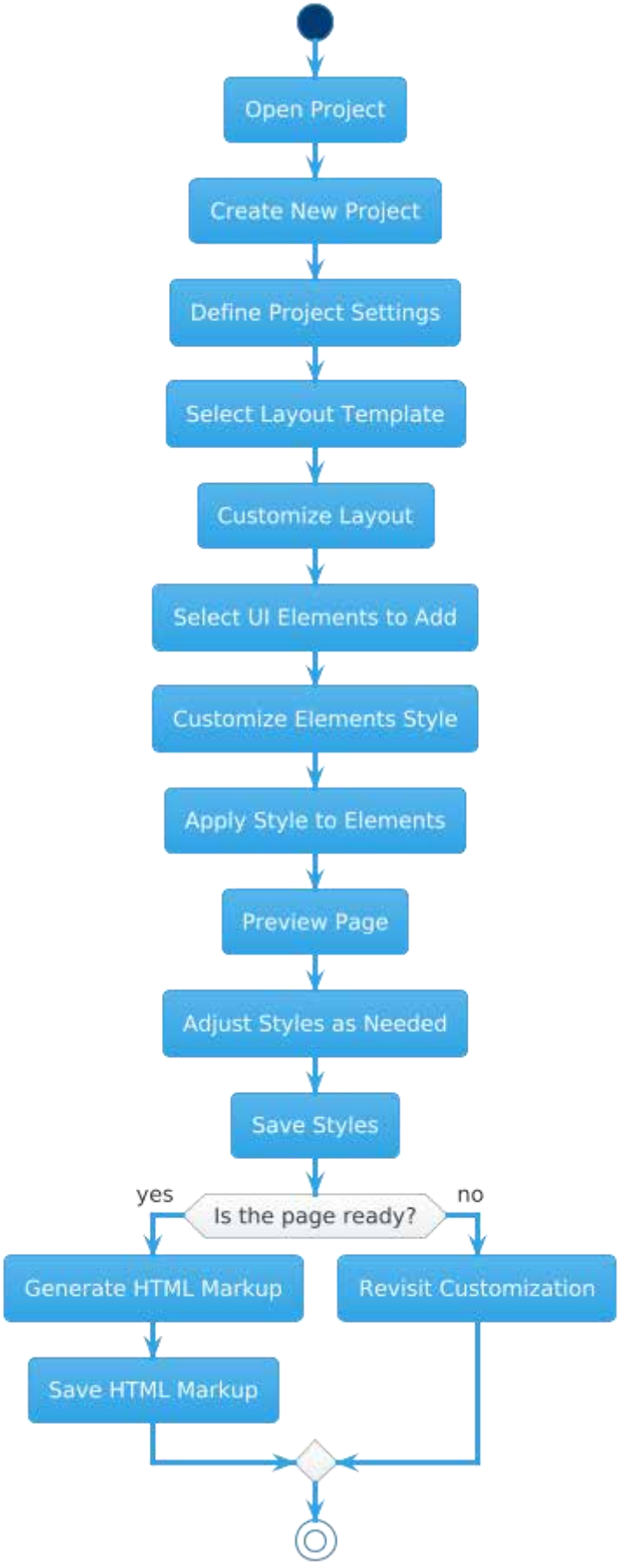


Рис. 5 Діаграма активності

Діаграма активності представляє процес створення розмітки HTML для інтерфейсу веб-сторінки в інтелектуальній системі. У ньому описано кроки, пов’язані з налаштуванням нового проекту, налаштуванням його макета та елементів інтерфейсу користувача та створенням остаточного результату HTML.

Користувач починає з відкриття існуючого проекту або запуску нового. Якщо користувач вирішує створити новий проект, він переходить до визначення основних параметрів проекту, таких як назва проекту, структура та початкова конфігурація. На цьому кроці користувач визначає деталі проекту, включаючи його структуру макета, цільову платформу або інші відповідні налаштування. Користувач вибирає шаблон для загального макета сторінки. Цей шаблон може містити попередньо визначені розділи, як-от верхні, нижні колонтитули та області вмісту.

Після вибору шаблону користувач налаштовує макет, налаштовуючи розділи або додаючи/видаляючи різні елементи, такі як стовпці, сітки або області вмісту. Користувач вибирає елементи інтерфейсу користувача, які він хоче додати на сторінку, наприклад кнопки, текстові поля, зображення або форми. Потім кожен вибраний елемент інтерфейсу користувача налаштовується відповідно до вподобань користувача щодо дизайну, що може включати налаштування таких властивостей, як колір, розмір шрифту, фон, межі тощо. Система застосовує налаштовані стилі до елементів інтерфейсу користувача, гарантуючи, що вони відповідають вибору дизайну користувача. Система забезпечує попередній перегляд сторінки в реальному часі, щоб користувач міг бачити, як виглядають внесені зміни в режимі реального часу. Переглянувши попередній перегляд, користувач може внести подальші коригування стилів, якщо це необхідно

для вдосконалення дизайну. Коли користувач задоволений дизайном, він зберігає остаточну конфігурацію стилю.

2.3 Структура джерела інформації для інтелектуального аналізу

У контексті інтелектуальної системи для створення розмітки HTML для інтерфейсів веб-сторінок структура інформації, яка використовується для інтелектуального аналізу, відіграє життєво важливу роль у продуктивності системи. Ця структура охоплює різні джерела даних, які допомагають системі обробляти вхідні дані та створювати оптимізований HTML-код.

Основним джерелом інформації є користувач, включаючи переваги дизайну, вибір компонентів і конкретні вимоги до стилю. Користувачі можуть вибрати попередньо визначені елементи інтерфейсу користувача, такі як кнопки, текстові поля або форми, і надати детальну інформацію про те, як вони хотіли б оформити ці компоненти, включаючи розмір шрифту, кольори та інші властивості CSS. Крім того, система повинна обробляти дані користувача щодо того, як дизайн має реагувати на різні розміри екрану, забезпечуючи зручність для мобільних пристроїв або адаптацію до різних пристроїв[18].

Система також спирається на колекцію попередньо визначених шаблонів і компонентів, які пропонують багаторазові елементи дизайну та структури. Вони можуть включати часто використовувані компоненти інтерфейсу користувача, такі як панелі навігації, картки та бічні панелі, які можна налаштувати відповідно до потреб користувача. Шаблони дизайну, що містять загальні макети веб-сторінок, включаючи верхні, нижні колонтитули та сітки, можна використовувати для автоматичного пристосування до вимог проекту користувача. Крім того, CSS-фреймворки,

такі як Bootstrap або Tailwind, можуть бути інтегровані, забезпечуючи узгодженість стилю та адаптивний дизайн.

Для створення HTML, CSS і JavaScript система повинна мати доступ до останніх стандартів і правил. Специфікації HTML визначають структуру та семантику розмітки, тоді як стандарти CSS керують дизайном та розташуванням елементів. Слід використовувати медіа-запити та методи компонування, такі як Flexbox і Grid, щоб переконатися, що сторінка правильно відображається на всіх розмірах екрана. У випадках, коли потрібна інтерактивність, знання JavaScript необхідні для інтеграції таких функцій, як перевірка форми, динамічний вміст або анімація.

Система також повинна інтегрувати найкращі практики дизайну UI/UX. До них входять рекомендації щодо доступності (WCAG), принципи дизайну, спрямовані на мобільність, і загальна зручність використання інтерфейсу. Дотримання умов кодування, таких як стандарти іменування елементів HTML, класів CSS і функцій JavaScript, гарантує, що згенерований код буде чистим і придатним для обслуговування.

Крім того, дані про користувачів, такі як попередні проекти, параметри дизайну та налаштування, є цінним ресурсом. Аналізуючи минулі проекти, система може передбачити дизайнерські тенденції користувача та запропонувати рекомендації на основі їх історичного вибору. Крім того, налаштування користувача, такі як темний режим або певні параметри шрифту, повинні зберігатися та автоматично застосовуватися до майбутніх проектів, спрощуючи процес.

У деяких випадках система може також використовувати зовнішні джерела даних. Це може включати веб-скрапінг для вилучення елементів дизайну з існуючих веб-сайтів або інтеграцію сторонніх служб, таких як бібліотеки шрифтів або значків, для покращення дизайну сторінки. Доступ

до онлайн-документації з таких ресурсів, як MDN або W3C, гарантує, що система залишається в курсі нових стандартів HTML, CSS і JavaScript.

Штучний інтелект відіграє важливу роль, аналізуючи великі набори даних взаємодії користувачів і макетів веб-сторінок. Моделі машинного навчання можуть запропонувати або передбачити найбільш підходящі компоненти чи стилі, пропонуючи користувачеві індивідуальні рекомендації. Обробка природної мови (NLP) також може бути використана для інтерпретації команд користувача, поданих простою мовою, які потім перетворюються на відповідний код HTML і CSS.

Коли система генерує розмітку, дуже важливо, щоб код відповідав поточним стандартам веб-розробки. Система повинна автоматично структурувати та форматувати HTML, CSS і JavaScript, щоб забезпечити бездоганний, чуйний і естетично привабливий інтерфейс користувача. Крім генерації коду, система повинна забезпечувати попередній перегляд дизайну в реальному часі, дозволяючи користувачам негайно переглядати внесені ними зміни.

Нарешті, система повинна забезпечити належне збереження всіх даних і конфігурацій, пов’язаних із проектом. Це включає не лише компоненти та стилі, вибрані користувачем, але й історичні файли та налаштування проекту. Слід запровадити контроль версій, щоб користувачі могли повернутися до попередніх ітерацій своєї роботи, а структуровану базу даних слід використовувати для ефективного зберігання та керування профілями користувачів, уподобаннями та деталями проекту.

Ефективно організувавши ці джерела інформації, інтелектуальна система для створення HTML-розмітки зможе створювати високоякісні, адаптивні та зручні для користувача інтерфейси веб-сторінок. Поєднуючи дані користувача, найкращі практики проектування, можливості штучного

інтелекту та зовнішні ресурси, система покращує як ефективність, так і якість процесу проектування, одночасно гарантуючи, що згенерована розмітка є сумісною та перспективною.

В ході роботи було створено сховище даних, що дозволить проводить аналіз в різних розрізах. Структура сховища даних представлена на рисунку 6.

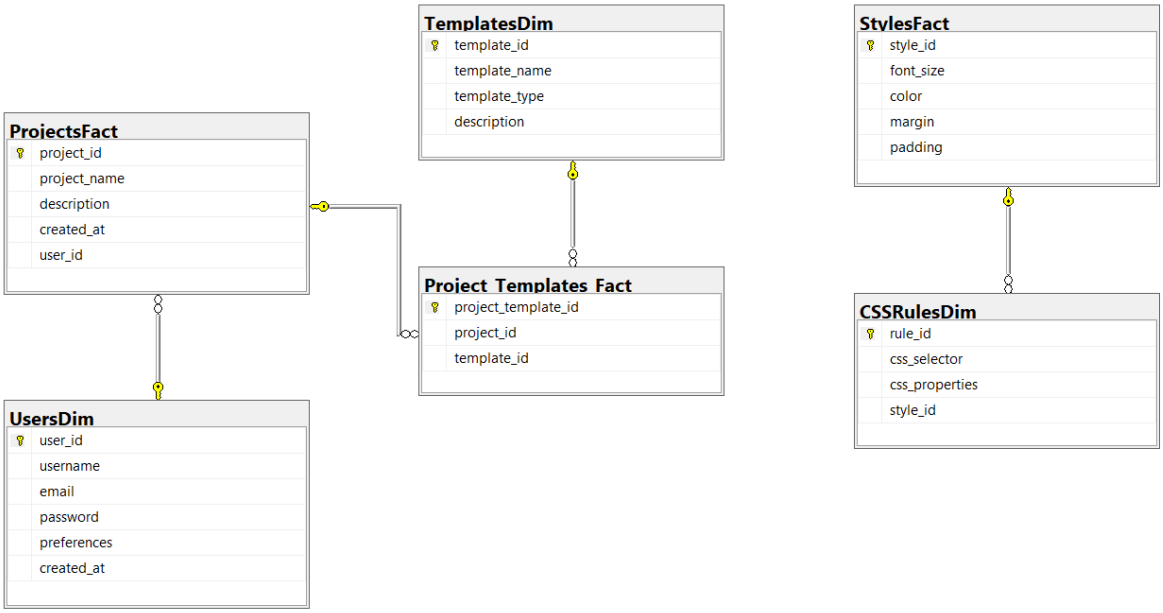


Рис. 6 Сховище даних

UsersDim: зберігає інформацію про користувачів, включаючи ім’я користувача, електронну адресу, пароль, налаштування та дату приєднання.

ProjectsFact: Кожен користувач може створити багато проектів. У цій таблиці зберігається така інформація про проект, як назва, опис і користувач, який його створив.

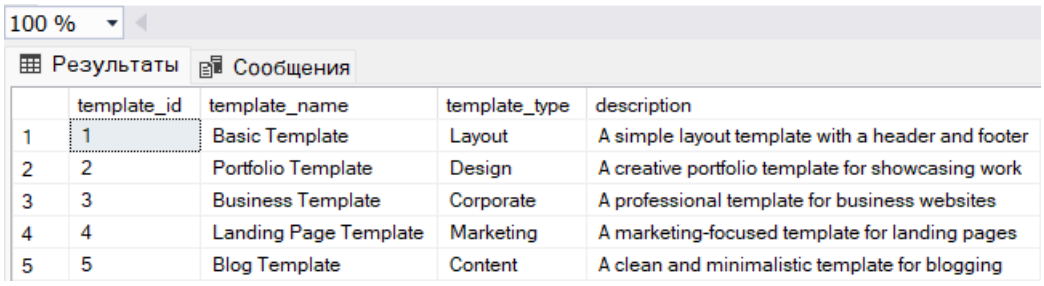
StylesFact: зберігає такі стилі, як розмір шрифту, колір, поля та відступи для компонентів.

CSS_Rules_Dim: зберігає правила CSS для компонентів, кожне правило пов’язане зі стилем.

TemplatesDim: містить попередньо визначені шаблони для макета або дизайну, які можна застосовувати до проектів.

Project_Templates_Fact: це об’єднана таблиця, яка дозволяє багатьом проектам використовувати багато шаблонів, створюючи зв’язок «багато-до-багатьох» між таблицями «Проекти» та «Шаблони».

Запити, по створенню таблиць-вимірів та таблиць-фактів написано на SQL та представлено у додатку А.


100 %

Результаты | Сообщения

	template_id	template_name	template_type	description
1	1	Basic Template	Layout	A simple layout template with a header and footer
2	2	Portfolio Template	Design	A creative portfolio template for showcasing work
3	3	Business Template	Corporate	A professional template for business websites
4	4	Landing Page Template	Marketing	A marketing-focused template for landing pages
5	5	Blog Template	Content	A clean and minimalistic template for blogging

Рис. 7 Збережені дані у виміру TemplatesDim

Результаты Сообщения

	style_id	font_size	color	margin	padding
1	1	16px	#333	10px	5px
2	2	14px	#555	15px	8px
3	3	18px	#000	20px	10px
4	4	12px	#777	12px	6px
5	5	20px	#222	25px	15px

Рис. 8 Збережені дані у виміру StylesDim

2.4 OLAP-технології

Технології OLAP (онлайн-аналітична обробка) необхідні для швидкого аналізу великих обсягів багатовимірних даних. У контексті інтелектуальної системи для створення розмітки HTML для веб-інтерфейсів технології OLAP можуть бути інструментальними в аналізі та управлінні даними, пов’язаними з елементами веб-сторінки, стилями, шаблонами та взаємодією користувачів. Ці технології дозволяють системі надавати інформацію в реальному часі та оптимізувати розмітку HTML на основі зібраних даних, підтримуючи ефективне прийняття рішень.

За допомогою OLAP дані можна організовувати в кілька вимірів, таких як шаблони, стилі дизайну, уподобання користувача та час. Наприклад, ця система може відстежувати та аналізувати популярність різних шаблонів або стилів, допомагаючи зрозуміти тенденції та шаблони в дизайні веб-сторінок. Упорядковуючи дані таким чином, OLAP дозволяє проводити більш динамічний і комплексний аналіз елементів дизайну, допомагаючи створювати ефективнішу розмітку HTML.

Центральне місце в OLAP займають куби даних, які зберігають інформацію в багатовимірному форматі. Для системи розробки веб-інтерфейсу ці куби можуть містити дані про стилі (наприклад, шрифти, кольори, поля тощо) у різних вимірах, дозволяючи дизайнерам або автоматизованим системам запитувати й аналізувати ці дані. Цей підхід

полегшує визначення елементів дизайну, які добре поєднуються, або які стилі найчастіше використовуються в успішних дизайнах сторінок.

Ще однією перевагою технологій OLAP є аналіз у реальному часі. Забезпечуючи миттєвий зворотний зв'язок, дизайнери або інтелектуальна система можуть швидко визначити найефективніші варіанти дизайну та застосувати їх до розмітки HTML. Цей швидкий аналіз допомагає оптимізувати веб-сторінки на основі поведінки користувачів і тенденцій дизайну, гарантуючи, що кінцевий продукт буде привабливим і функціональним[19].

Інтелектуальний аналіз даних і прогнозний аналіз додатково покращуються завдяки OLAP. Завдяки аналізу минулого вибору дизайну та взаємодії користувачів система може передбачити тенденції та запропонувати елементи дизайну, які, ймовірно, добре працюватимуть у майбутньому. Ця можливість прогнозування допомагає системі генерувати розмітку HTML, яка не тільки базується на поточних уподобаннях, але також є перспективною та адаптується до мінливих вимог користувачів.

OLAP також забезпечує інтерактивні запити, дозволяючи дизайнерам легко досліджувати та маніпулювати великими наборами даних. У випадку інтелектуальної системи HTML це означає, що користувачі можуть досліджувати різні комбінації стилів, шаблонів і поведінки користувачів, приймаючи обґрунтовані рішення щодо макета та вмісту веб-сторінок. Ця інтерактивна функція робить процес проектування більш інтуїтивно зрозумілим і керованим користувачем.

Крім того, технології OLAP дозволяють оптимізувати розмітку HTML. Аналізуючи різні параметри дизайну та їхній вплив на залучення користувачів, продуктивність і зручність використання, система може створити найбільш відповідний HTML-код. Ця оптимізація гарантує, що згенерована розмітка є ефективною, візуально привабливою та покращує взаємодію з користувачем.

Інтеграція OLAP із інструментами бізнес-аналітики може ще більше покращити процес аналізу. Поєднуючи інформацію про поведінку користувачів, уподобання та ринкові тенденції, система може генерувати розмітку HTML, яка відповідає бізнес-цілям, покращуючи взаємодію з користувачем. Це допомагає компаніям створювати більш персоналізовані та ефективні веб-сторінки.

Наприклад, OLAP може аналізувати ефективність різних шаблонів з часом, визначаючи, які з них призводять до кращого залучення користувачів або коефіцієнтів конверсії. Так само, досліджуючи взаємодію користувача з різними елементами дизайну (такими як кнопки, заголовки та зображення), система може запропонувати найефективніші стилі та структури HTML для майбутніх веб-сторінок.

Крім того, технології OLAP можуть підтримувати тестування A/B, дозволяючи системі порівнювати різні версії дизайну та оцінювати їх продуктивність. Це тестування допомагає визначити, які комбінації стилів і елементів є найефективнішими, надаючи на основі даних розуміння для майбутніх дизайнів.

Таким чином, технології OLAP відіграють вирішальну роль у створенні інтелектуальних систем для генерації розмітки HTML. Організовуючи дані в багатовимірних форматах, забезпечуючи аналіз у реальному часі та підтримуючи прогнозну інформацію, OLAP дозволяє системі створювати оптимізовані та орієнтовані на користувача веб-сторінки. Здатність аналізувати тенденції, передбачати майбутні потреби в дизайні та оптимізувати розмітку HTML гарантує, що кінцеві веб-інтерфейси будуть не тільки візуально привабливими, але й функціональними, ефективними та відповідатимуть очікуванням користувачів.

3 РОЗРОБКА СИСТЕМИ

3.1 Структура бази даних

Для визначення структури бази даних доцільно спочатку визначити функції, пов’язані зі збереженням даних – інформаційних потреб користувачів, представлених у вигляді концептуальних запитів. Визначити потреби користувачів можливо, спираючись на розроблену раніше діаграму прецедентів використання. Отже, отримаємо наступні концептуальні запити:

1. Перевірити відповідність аутентифікаційних даних (логін, пароль);
2. Зареєструвати нового користувача (ім’я, e-mail, логін, пароль);
3. Вивести інформацію про користувача (ім’я, логін);
4. Створити новий проєкт (ID, назва, дата створення, автор);
5. Вивести інформацію про проєкт (ID, назва, дата створення, автор, адреса файлу);
6. Видалити проєкт (ID, назва, дата створення, автор, адреса файлу).

Як видно, концептуальні запити до бази даних не описують повну взаємодію користувача з системою. Це обумовлено тим, що деякі інформаційні процеси виконуються виключно на стороні клієнта, а деякі не потребують обов’язкового використання бази даних. При цьому варто враховувати можливість подальшого розширення бази даних.

Отже, можна зробити висновок, що база даних буде обмежена лише двома таблицями: Користувачі та Проєкти. Один користувач може створювати декілька проєктів, а кожен проєкт є самостійною вебсторінкою, оскільки система дозволяє створювати односторінкові веб-інтерфейси.

Розглянемо детальніше ці дві таблиці. Таблиця Користувач має наступні колонки: ID – ідентифікатор користувача в системі, ім’я, e-mail, логін і пароль. Всі ці поля є обов’язковими для заповнення і створюються під час реєстрації нового користувача в системі. Це необхідно для коректної роботи всіх функцій сайту, оскільки доступ до проєкту повинен бути забезпечений тільки для автора і обмежений для всіх інших користувачів. Таблиця Проєкт має колонки: ID, назва проєкту, дата створення проєкту, ID користувача (автора проєкту), адреса файлу, в якому зберігаються всі внесені зміни.

За цими сутностями можна скласти модель бази даних у вигляді ER-діаграми. Такі моделі дозволяють описувати структуру та взаємозв’язки між даними в системі. Використання ER-моделі дозволяє абстрагувати ключові аспекти даних та їх зв’язки, щоб створити зрозумілу та легко адміністровану базу даних. В рамках ER-моделі бази даних виокремлюються кілька основних понять, зокрема: сутності, відношення та атрибути. Цю модель можна представити у вигляді діаграми, що ілюструє структуру та взаємозв’язки між сутностями бази даних. При подачі предметної області за допомогою ER-моделі за правилами побудови концептуальних схем сутності зображуються у вигляді прямокутників, асоціації – ромбів, зв’язки – ненаправлених ребер, над якими може проставлятися ступінь зв’язку та необхідне пояснення. Розроблена ER-модель для інтелектуальної системи створення HTML-інтерфейсів веб-сторінок наведена на рис. 9.

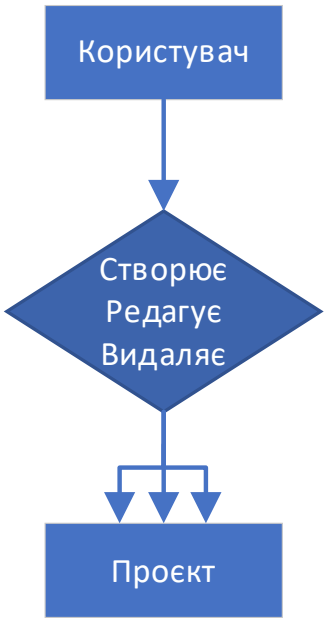


Рис. 9 ER-модель бази даних

Дана модель описує структуру бази даних, відповідно до описаних вище запитів, складених на основі потреб користувачів системи. На рисунку відображено, що користувач може створювати, редагувати та видаляти декілька проєктів. Інші сутності при цьому не передбачені. З точки зору структури таблиць, це обумовлено тим, що проєкт має посилання на ідентифікатор користувача, утворюючи зв'язок.

Зв'язки між таблицями можна зручно представити у вигляді діаграми бази даних, яку можна скласти у системі управління базами даних. Але для цього необхідно спочатку створити таблиці, відповідно до опису.

Створимо таблицю «Користувачі»:

```
CREATE TABLE Users(
    ID INT AUTO_INCREMENT PRIMARY KEY,
    Login VARCHAR(50) NOT NULL,
    Passw VARCHAR(100) NOT NULL,
    Email VARCHAR(320) NOT NULL,
```

Name VARCHAR(100) NOT NULL

);

Створимо таблицю «Проєкти»:

CREATE TABLE Projects(

ID INT AUTO_INCREMENT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

CreateDate DATE NOT NULL,

FileName VARCHAR(2048) NOT NULL,

UserID INT NOT NULL, FOREIGN KEY (UserID) REFERENCES Users(ID)

);

Створені таблиці можна переглянути у вигляді діаграми, наведеної на рис. 10.

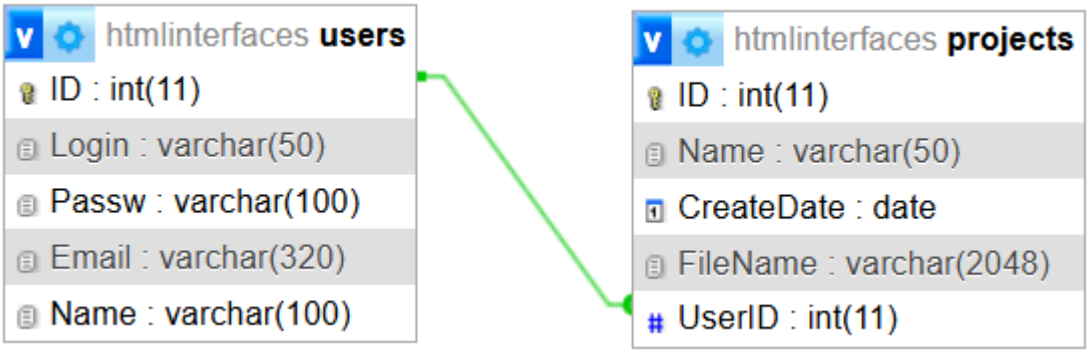


Рис. 10 Структура бази даних

На рисунку видно відношення між елементами бази даних.

3.2 Клієнтська частина

Для розробки програмного забезпечення клієнта прийнято рішення використовувати HTML, CSS, JavaScript та Bootstrap. При цьому код клієнта частково формуватиметься на сервері, оскільки серверна PHP дозволяє зручно працювати з обома мовами одночасно. Таке рішення дозволить спростити процес розробки програмного забезпечення і зробити генерування сторінок більш динамічним.

В першу чергу визначимо окремі інтерфейси, які мають бути винесені на відокремлені сторінки (розділи сайту). Для роботи сайту необхідно створити такі сторінки:

1. Головна сторінка з описом сайту та простим навігаційним меню;
2. Форма авторизації, де користувач може увійти в систему;
3. Форма реєстрації, де може зареєструватись на сайті;
4. Розділ проєктів, де можна переглядати всі свої проєкти, створювати нові, редагувати та видаляти існуючі;
5. Розділ редагування інтерфейсу, де користувач може додавати та редагувати елементи вебсторінки, змінювати її налаштування, переглядати та зберігати внесені зміни.

Для коректного функціонування системи на стороні клієнта важливо дотримуватись вимог до безпеки, інтерактивності та адаптивності. Адаптивність забезпечується за допомогою Bootstrap, а інші функції за допомогою скриптів, розроблених з використанням мови JavaScript.

Для всіх даних, що вводяться користувачем, у системі має бути передбачена можливість валідації. Це необхідно для того, щоб на сервер відправлялись тільки коректні дані. Валідація здійснюється завдяки JS-коду на сторінці форми. Цей код відстежує стан форми та перевіряє дані перед відправленням на сервер.

```
const togglePassword = document.getElementById('togglePassword');
const passwordField = document.getElementById('password');
const loginField = document.getElementById('login');
const form = document.querySelector('form');
const errorMessage = document.getElementById('error-message');
togglePassword.addEventListener('click', () => {
    const type = passwordField.getAttribute('type') === 'password' ? 'text' : 'password';
    passwordField.setAttribute('type', type);
    togglePassword.textContent = type === 'password' ? 'Показати' : 'Приховати';
});
form.addEventListener('submit', (event) => {
    const login = loginField.value;
    const regex = /^[a-zA-Z0-9]+$/;
    if (!regex.test(login)) {
      event.preventDefault();
      errorMessage.classList.remove('d-none');
    } else {
      errorMessage.classList.add('d-none');
    }
});
```

Рис. 10 Валідація даних з використанням JavaScript на прикладі форми входу

Наведений код відстежує вміст полів введення перед відправленням запитів на сервер, уникаючи таким чином виникнення помилок на стороні сервера та підвищуючи якість програмного забезпечення в цілому. При цьому повідомлення про помилку може відображатись за допомогою HTML-коду з використанням бібліотек Bootstrap:

Логін може містити лише букви латиниці та цифри.

Також система має підтримувати динамічне оновлення всіх елементів, що відображаються на екрані. Для цього може використовуватись технологія AJAX. При цьому окремі частини клієнтського коду мають генеруватись на стороні сервера, тому доцільно розглянути детальніше розробку серверного ПЗ.

3.3 Архітектура програмного забезпечення

Для розробки інтелектуальної системи створення HTML-інтерфейсів вебсторінок однією з першочергових задач є визначення архітектури майбутньої системи. На цьому етапі важливо конкретизувати вимоги до функціоналу системи, визначити її складові та їх взаємозв'язки, розробити алгоритми та сформувати структуру бази даних.

В рамках виконання роботи доцільно застосувати як функціональний, так і об'єктно-орієнтований підхід до моделювання майбутньої системи. Це дозволить обрати оптимальний підхід до розробки кожного з елементів системи.

Архітектура системи створення інтерфейсів вебсторінок буде виконана у вигляді веб-додатку. Тобто буде реалізована клієнт-серверна архітектура. Це одна з основних моделей програмного забезпечення, що передбачає чітку взаємодію між компонентами системи через обмін даними. Вона складається з трьох основних елементів: клієнта, який звертається до сервера для отримання ресурсів чи послуг; сервера, який обробляє ці запити і надає необхідні ресурси; та мережі, що забезпечує їх взаємодію [13].

Клієнт у цій архітектурі зазвичай є персональним комп'ютером або іншим пристроєм, що ініціює запити до сервера для отримання інформації або виконання певних операцій. Сервер, в свою чергу, є потужним комп'ютером або спеціалізованим обладнанням, призначеним для обробки запитів клієнтів, зберігання даних у базах і надання доступу до них. Розроблювана система має реалізує взаємодію між клієнтом і сервером: клієнт надсилає запит на сервер, той обробляє його і відправляє результат назад. Сервер здатен одночасно обробляти кілька запитів від різних клієнтів, організовуючи чергу для запитів з більшим навантаженням. У разі необхідності сервер може обробляти запити за пріоритетом [13].

На серверній стороні реалізуються функції збереження, обробки даних, резервного копіювання та захисту інформації. Сервер також забезпечує виконання запитів і передачу результатів клієнту. Клієнт, у свою чергу, відповідає за створення графічного інтерфейсу для користувача, формування запитів і передачу їх на сервер, а також обробку отриманих відповідей. Для динамічного оновлення даних без необхідності повного перезавантаження сторінки клієнт може надсилати додаткові запити, такі як оновлення, додавання або видалення даних [13].

Взаємодія між клієнтом і сервером визначається протоколами, що регулюють порядок обміну даними. Вибір протоколу залежить від конкретних вимог до системи, таких як швидкість, надійність чи безпека передавання інформації [13].

Згідно з описом архітектури системи, можна зробити висновок про взаємодію користувача з системою. Графічно це можна представити у вигляді схеми, наведеної на рис. 11

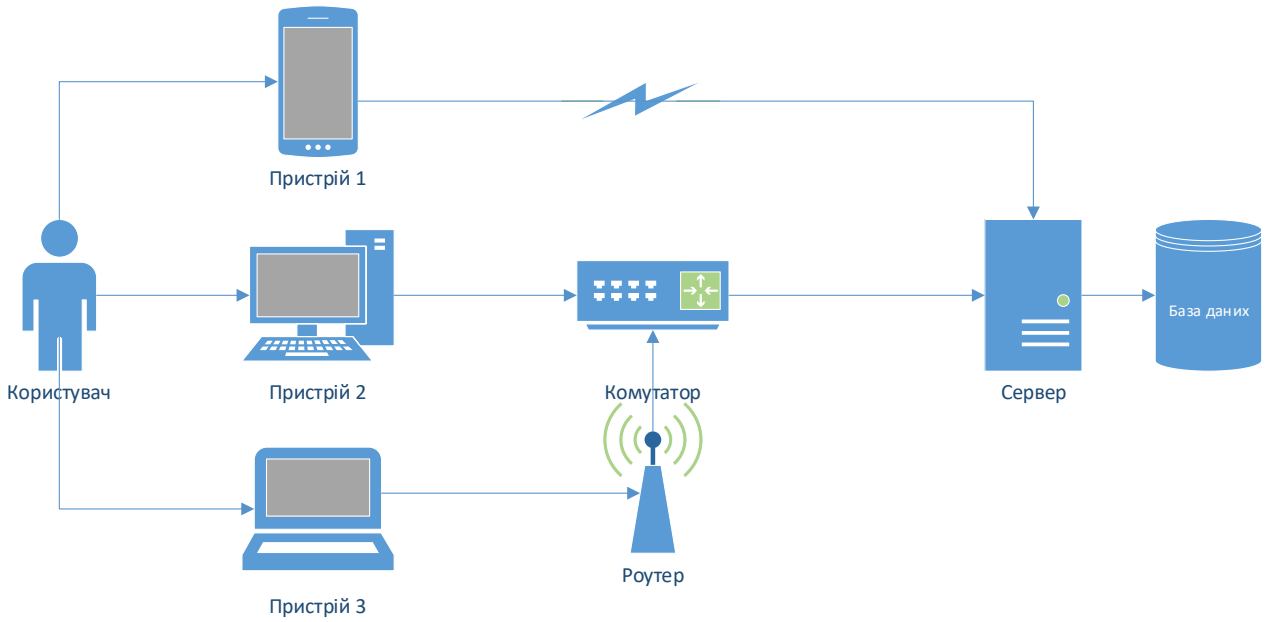


Рис. 11 Взаємодія користувача з системою

На рисунку видно, що користувач може використовувати різні типи пристроїв від звичайного настільного ПК до смартфону, які через мережеве обладнання, включаючи лінії бездротового зв’язку звертаються до сервера, який взаємодіє з базою даних. При цьому база даних може знаходитись як на цьому сервері, так і на іншому. Тоді взаємодія сервера з базою здійснюється аналогічно до взаємодії клієнта з сервером.

3.4 Серверна частина

Серверна частина сайту розробляється з використанням мови програмування PHP. При цьому запити можуть оброблятися за допомогою окремих модулів. Для коректної роботи сайту доцільно створити окремий модуль для авторизації, роботи з проєктами, роботи з інтерфейсами, а також інших функцій. При цьому запити від клієнта обробляються за допомогою спеціальних контролерів.

Важливою частиною роботи сервера є забезпечення регулювання прав користувачів та інші заходи захисту. Всі дані, що обробляються сервером, мають бути перевірені, а всі форми захищені від SQL-ін’єкцій та CSRF-атак. Для забезпечення обмежень доступу та захисту від CSRF-атак використовуються сесії. Розглянемо на прикладі:

```php
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
$DBhost = 'localhost';
$DBuser = 'root';
$DBpassword = '';
$DBname = 'HTMLinterfaces';
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
try {
    $db = new mysqli($DBhost, $DBuser, $DBpassword, $DBname);
} catch (mysqli_sql_exception $e) {
    $error = 'Помилка відкриття бази даних: ' . $e->getMessage();
}

```

Рис. 12 Створення csrf-токену

Наведений вище уривок коду створює сесію, генерує CSRF-токен, перевіряє роль користувача і виконує підключення до бази даних. Також, з метою підвищення безпеки, всі паролі зберігаються виключно у хешованому вигляді.

Розглянемо приклад коду перевірки пароля:

```php
if (password_verify($_POST['password'], $hashedPassword)) {
    $_SESSION['name'] = $userName;
    header('Location: index.php');
    exit();
} else {
    $error = 'Неправильний пароль.';
    }if (password_verify($_POST['password'], $hashedPassword)) {
    $_SESSION['name'] = $userName;
    header('Location: index.php');
    exit();
} else {
    $error = 'Неправильний пароль.';
}

```

Рис. 13 Перевірка паролю користувача

Таким чином здійснюється перевірка правильності введення облікових даних для входу в акаунт. У випадку помилки користувач отримає відповідне сповіщення.

В цілому серверне програмне забезпечення працює за наступним алгоритмом: на сервер надходить запит від клієнта, сервер перевіряє отримані дані, виконує запит до бази даних, формує і повертає клієнту відповідь. Тобто розроблений алгоритм є доволі простим. Проте система ускладняється зі збільшенням кількості інтерфейсів, пов’язаних з обробкою різних типів даних, що робить неможливим використання універсальних рішень.

Таким чином розробляється все серверне програмне забезпечення, яке обробляє запити клієнтів.

3.5 Вибір інструментарію для створення програмного забезпечення

Під час розробки інтелектуальної системи для створення розмітки HTML для інтерфейсів веб-сторінок вибір інструментів має вирішальне значення для продуктивності системи, її масштабованості та взаємодії з користувачем. Для ефективного досягнення системою поставлених цілей необхідна комбінація технологій для розробки серверної та зовнішньої частини.

PHP обрано як основну мову сценаріїв на стороні сервера через його широке поширення та потужну підтримку спільноти. Він чудово справляється зі створенням динамічного вмісту, обробкою форм та інтеграцією даних, що робить його ідеальним вибором для керування функцією серверної частини. PHP взаємодіятиме з компонентами інтерфейсу для обробки запитів

користувачів, таких як створення проекту, вибір шаблону та коригування стилю, динамічно генеруючи вміст HTML.

Для зберігання даних MySQL є кращою системою керування реляційною базою даних. Він забезпечує надійну, швидку та безпечну платформу для керування структурованими даними. Це включає зберігання проектів, налаштувань користувача, шаблонів і стилів. Здатність MySQL обробляти великі набори даних у поєднанні з бездоганною інтеграцією з PHP забезпечує ефективний пошук і маніпулювання даними. База даних підтримує потреби системи в масштабованості та цілісності даних, що робить її відповідним вибором для цього проекту.

На інтерфейсі HTML утворює основу структури веб-сторінки. Це дозволяє системі визначати вміст і макет інтерфейсу користувача. Елементи HTML, такі як кнопки, форми та таблиці, забезпечують базову структуру, необхідну для створення інтерактивного інтерфейсу користувача. Ця мова розмітки необхідна для динамічного відтворення вмісту та бездоганної інтеграції з CSS для стилізації та JavaScript для інтерактивності.

CSS, або каскадні таблиці стилів, відіграє ключову роль у визначенні візуального вигляду та макета веб-сторінок. Це дозволяє системі застосовувати узгоджені стилі, такі як розмір шрифту, кольори та поля, до елементів HTML. CSS також гарантує, що система реагує, що означає, що інтерфейси адаптуються до різних розмірів екрана, забезпечуючи узгоджену взаємодію з користувачем на різних пристроях. Розділення вмісту (HTML) і презентації (CSS) покращує зручність обслуговування та гнучкість.

JavaScript використовується для обробки динамічних взаємодій на стороні клієнта. Це додає системі інтерактивність, дозволяючи користувачам змінювати стилі, вибирати елементи та переглядати зміни в реальному часі без оновлення сторінки. JavaScript також виконує такі завдання, як перевірка форми, маніпулювання об’єктною моделлю документа (DOM) і керування

асинхронними операціями, що сприяє плавнішому та чутливішому взаємодії з користувачем.

Laravel, фреймворк PHP, обрано за його потужні функції та елегантний синтаксис. Він відповідає архітектурі Model-View-Controller (MVC), яка допомагає розділити логіку програми на чіткі рівні для кращої організації та масштабованості. Laravel спрощує маршрутизацію, взаємодію з базою даних і обробку запитів користувачів, забезпечуючи надійну та ефективну базову структуру. Його Eloquent ORM полегшує керування базами даних, а вбудовані функції безпеки забезпечують безпечну обробку даних. Підтримка Laravel для створення RESTful API дозволяє інтерфейсу без проблем взаємодіяти з сервером, покращуючи загальну взаємодію з користувачем.

Разом ці технології — PHP, MySQL, HTML, CSS, JavaScript і Laravel — утворюють комплексне рішення для розробки інтелектуальної системи, яка генерує розмітку HTML для веб-інтерфейсів. Поєднання PHP і Laravel буде керувати серверною обробкою, тоді як MySQL ефективно керуватиме та зберігатиме дані. HTML, CSS і JavaScript працюватимуть у тандемі, щоб забезпечити динамічний, адаптивний і зручний інтерфейс, що дозволить користувачам легко створювати, налаштовувати та переглядати веб-сторінки. Цей набір інструментів гарантує, що система є не тільки функціональною та надійною, але й масштабованою для задоволення майбутніх вимог.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Вимоги до апаратного та програмного забезпечення

У розробці інтелектуальної системи для створення розмітки HTML для інтерфейсів веб-сторінок вимоги до апаратного та програмного забезпечення відіграють важливу роль у забезпеченні ефективності, стабільності та масштабованості системи. Вибір вимог повинен враховувати потреби розробки, розгортання та досвід кінцевого користувача.

Вимоги до обладнання:

Апаратна конфігурація, необхідна для розробки та роботи інтелектуальної системи, повинна бути достатньо надійною, щоб відповідати як обчислювальним вимогам системи, так і зберіганню даних. Специфікації апаратного забезпечення повинні підтримувати етапи розробки, тестування та живого розгортання програми.

Процесор (ЦП): рекомендується сучасний багатоядерний процесор (Intel i5 або вище або еквівалентний процесор AMD). Це забезпечує плавне виконання завдань розробки, таких як запуск локальних серверів, тестування та компіляція коду. Для виробничих середовищ процесор серверного рівня,

наприклад Intel Xeon або AMD EPYC, забезпечить кращу продуктивність і масштабованість.

Оперативна пам’ять: для цілей розробки пропонується мінімум 8 ГБ оперативної пам’яті, чого буде достатньо для запуску необхідних інструментів розробки програмного забезпечення, локальних серверів і баз даних. Для робочих середовищ, особливо коли система обслуговуватиме кількох користувачів або оброблятиме великі набори даних, може знадобитися 16 ГБ або більше для забезпечення безперебійної роботи.

Жорсткий диск: системі потрібно принаймні 500 ГБ пам’яті для цілей розробки та тестування, зокрема для зберігання всіх відповідних файлів проекту, вихідного коду, резервних копій бази даних і журналів. Для виробничого використання ідеально підходять масштабовані рішення для зберігання даних, такі як твердотільні накопичувачі (SSD), оскільки вони забезпечують більш високу швидкість читання/запису, покращуючи швидкість реакції системи. Рішення для зберігання на основі хмари також можна використовувати для більшої масштабованості.

Мережа: стабільне високошвидкісне підключення до Інтернету є важливим як для розробки, так і для розгортання. Для виробничих середовищ необхідно забезпечити достатню пропускну здатність для обробки запитів користувачів і передачі даних, особливо якщо програма обробляє завантаження чи завантаження великих файлів.

Графічний процесор (GPU): хоча графічний процесор високого класу не є необхідним для розробки серверної частини, його наявність може покращити можливості графічного рендерингу системи під час попереднього перегляду інтерфейсів веб-сторінок у режимі реального часу, особливо якщо ви працюєте зі складними стилями та анімацією[25].

Системи резервного копіювання та резервування: для критично важливих даних і для забезпечення надійності системи рекомендується

використовувати автоматизовану систему резервного копіювання, таку як конфігурації RAID або хмарні резервні копії, щоб запобігти втраті даних через апаратні збої.

4.2 Тестування системи

Запустивши систему, бачимо головну сторінку програмного забезпечення (рис. 14).

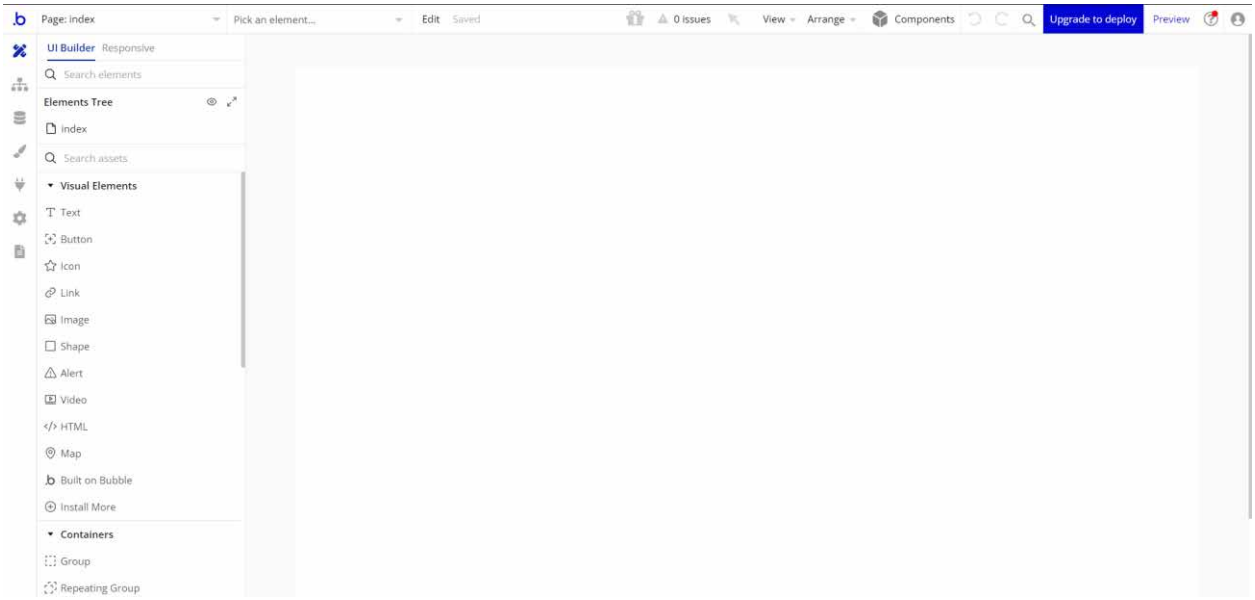


Рис. 14 Головна сторінка

Тут у нас представлена головна робоча панель з усіма інструментами в лівому кутку екрану та самим аркушем веб-сторінки по центру.

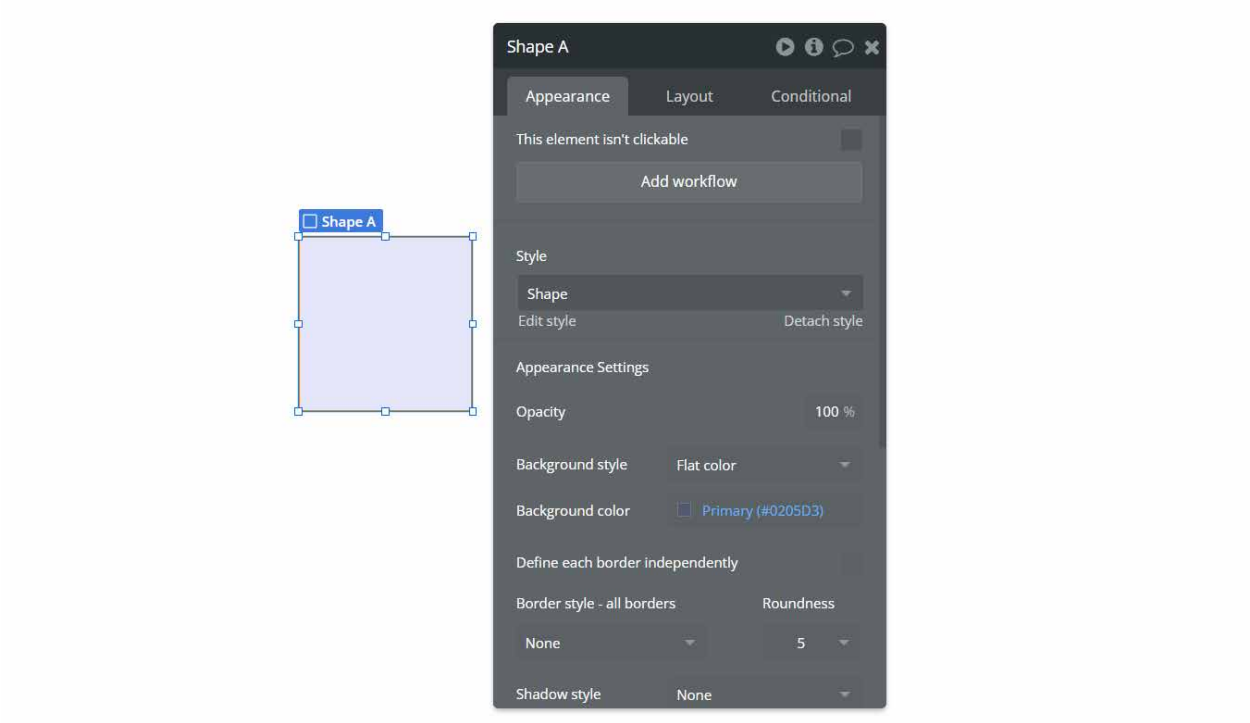


Рис. 15 Створення елементу

Ми можемо перетягувати будь-які елементи з лівої панелі прямо на лист і потім редагувати їх за власними уподобаннями.

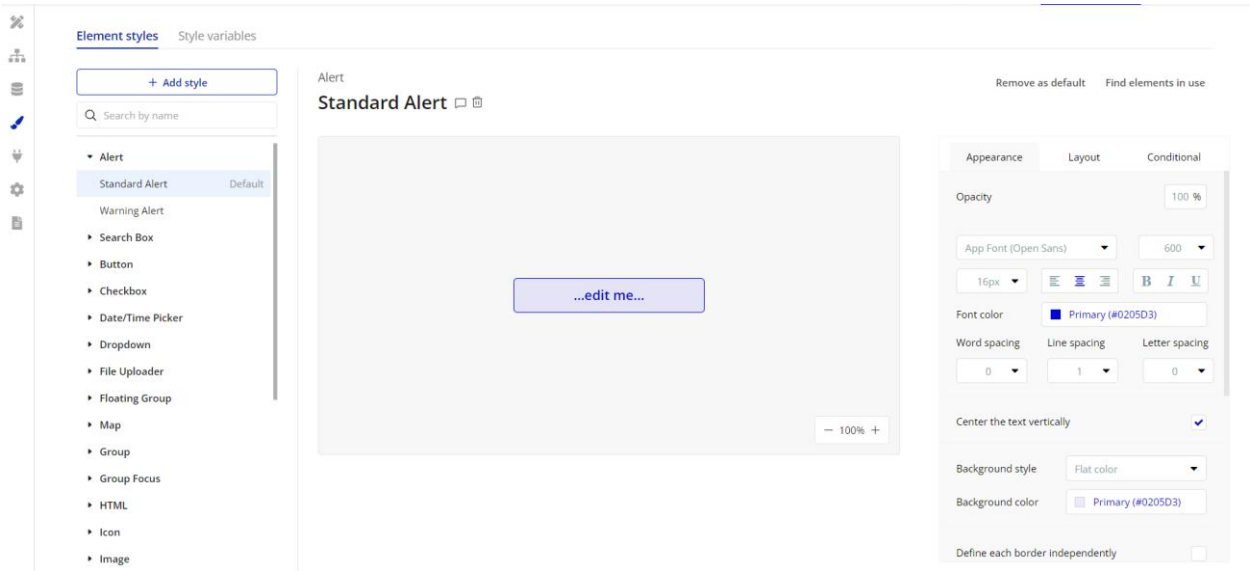


Рис. 16 Панель керування стилями

Наступна функція – управління стилями. Ми можемо створювати власні стилі, змінювати їх налаштування і підбирати вподобане нам елементи.

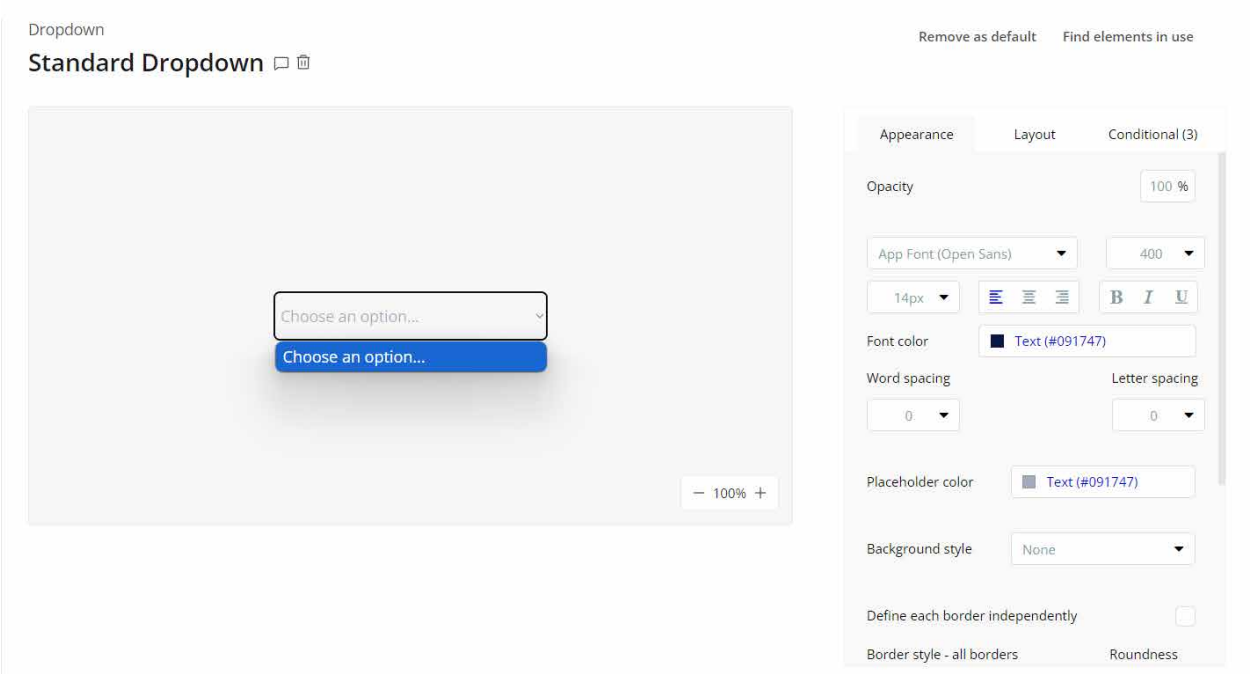


Рис. 17 Створення стилю

Зараз був створений стиль для поля для введення зі списком, що випадає. Ми призначили невеликі візуальні зміни та поведінку при натисканні.

4.3 Аналіз отриманих результатів

Інтелектуальна система для створення HTML-розмітки для інтерфейсів веб-сторінок успішно досягла своєї основної мети автоматизації та вдосконалення процесу розробки веб-сторінки та створення розмітки. Завдяки застосуванню передових технологій і продуманому дизайну системи було досягнуто кількох ключових досягнень, зокрема покращення ефективності, користувацького досвіду та адаптивності. Нижче наведено детальний аналіз отриманих результатів.

Одним із найважливіших результатів цієї системи є автоматизація створення розмітки HTML. Використовуючи інтелектуальні алгоритми, система здатна динамічно створювати структурований HTML-код на основі введених користувачем даних, таких як вибір шаблонів, визначення стилів і елементів. Ця автоматизація значно зменшує потребу в ручному кодуванні,

заощаджуючи значний час і зусилля для веб-розробників і дизайнерів. Користувачі можуть просто взаємодіяти з інтерфейсом системи, і система створить оптимізований HTML-код, готовий до розгортання, що зробить процес веб-розробки швидшим і ефективнішим.

Система має інтуїтивно зрозумілий, зручний інтерфейс, який дозволяє користувачам легко вибирати елементи, застосовувати стилі та візуалізувати структуру HTML у реальному часі. Зосередившись на простоті та доступності, система дозволила користувачам без великих знань програмування створювати макети веб-сторінок професійного рівня. Цей дизайн, орієнтований на користувача, гарантує, що навіть ті, хто має обмежені технічні навички, можуть ефективно створювати веб-сторінки з індивідуальним дизайном і стилями, тим самим демократизуючи процес веб-дизайну.

Головним досягненням є інтеграція функції попереднього перегляду в реальному часі. Коли користувачі змінюють елементи та застосовують стилі, система оновлює попередній перегляд інтерфейсу веб-сторінки в реальному часі. Ця функція дозволяє користувачам негайно побачити наслідки внесених змін, забезпечуючи ефективність та повторюваність процесу проектування. Користувачам більше не потрібно перемикатися між редактором і браузером, щоб перевірити, як виглядатимуть їхні зміни, що спрощує робочий процес дизайну та покращує загальну взаємодію з користувачем.

Функція налаштування стилю системи дозволяє користувачам визначати різні властивості CSS, такі як розмір шрифту, колір, поля, відступи тощо. Ці стилі можна налаштувати на рівні елемента, забезпечуючи гнучкість і контроль над зовнішнім виглядом веб-сторінки. Система також підтримує масштабованість, тобто користувачі можуть застосовувати різні стилі до кількох елементів, не порушуючи загальний дизайн. Гнучкість налаштування стилів як для окремих компонентів, так і для всієї сторінки розширює можливості налаштування для користувачів, роблячи її адаптованою до різних потреб і вподобань дизайну.

ВИСНОВКИ

Розробка інтелектуальної системи для створення HTML-розмітки для інтерфейсів веб-сторінок успішно вирішила ключові проблеми сучасного веб-дизайну, запропонувавши інноваційне рішення для автоматизації створення структур веб-сторінок. Ця система поєднує в собі потужність передових алгоритмів з інтуїтивно зрозумілим інтерфейсом користувача для створення ефективного та доступного інструменту, що значно підвищує швидкість і якість веб-розробки.

Здатність системи автоматично генерувати HTML-розмітку на основі введених користувачами — чи то за допомогою попередньо визначених шаблонів, налаштованих стилів чи вибору елементів — виявилася надзвичайно корисною для зменшення ручних зусиль, які традиційно потрібні для веб-розробки. Інтеграція попереднього перегляду в режимі реального часу гарантує, що користувачі можуть миттєво побачити наслідки своїх дизайнерських рішень, забезпечуючи більш динамічний і повторюваний процес проектування.

Зосереджуючись на взаємодії з користувачем, система забезпечує бездоганну та доступну платформу для користувачів із різним рівнем досвіду у веб-дизайні. Незалежно від того, новачок чи досвідчений розробник, гнучкість системи та простота використання гарантують, що будь-хто зможе швидко створювати функціональні та візуально привабливі веб-сторінки, не потребуючи глибоких знань HTML, CSS або JavaScript.

Крім того, зосередженість системи на масштабованості та налаштуванні робить її адаптованою до широкого діапазону потреб дизайну, від простих веб-сайтів до більш складних веб-додатків. Можливість ефективного керування стилями, шаблонами та даними проекту забезпечує узгодженість і скорочує час, витрачений на зміну конфігурації елементів дизайну. Це особливо важливо в динамічній сфері веб-розробки, де час і економічна ефективність є вирішальними.

Хоча система вже досягла значних успіхів у автоматизації створення розмітки HTML, її потенціал для майбутнього розвитку залишається перспективним. Подальші удосконалення можуть включати рекомендації стилю, керовані штучним інтелектом, розширені параметри налаштування користувача та більше інтеграції з іншими інструментами веб-розробки, що додатково розширить можливості системи.

Підсумовуючи, інтелектуальна система для створення розмітки HTML являє собою значний крок вперед у спрощенні та автоматизації процесу веб-розробки. Він надає цінний інструмент для веб-дизайнерів і розробників, пропонуючи їм спрощене рішення для ефективного створення розмітки HTML, водночас дозволяючи високий ступінь налаштування. Ця система не тільки підвищує продуктивність, але й демократизує веб-дизайн, роблячи його більш доступним для ширшої аудиторії та сприяючи більш творчому та спільному підходу до веб-розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клієнт-серверна архітектура. QATestLab training center. [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
2. HTML, CSS і JavaScript: основи веб-розробки. Pravda твого міста. [Електронний ресурс] – Режим доступу: https://pravda.if.ua/html-css-i-javascript-osnovy-veb-rozrobky/
3. What is PHP? The PHP Group. [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/intro-whatis.php
4. Організація баз даних. Тема 6 – Теорія нормалізації реляційної моделі даних. Сумський державний університет. [Електронний ресурс] – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44494/index.html
5. Wix – Create a Website You'll Love. Wix. [Електронний ресурс] – Режим доступу: https://www.wix.com
6. Squarespace – Create a Website. Squarespace. [Електронний ресурс] – Режим доступу: https://www.squarespace.com

7. WordPress.com – Create a Free Website or Blog. WordPress. [Електронний ресурс] – Режим доступу: https://www.wordpress.com
8. Weebly – Website Builder. Weebly. [Електронний ресурс] – Режим доступу: https://www.weebly.com
9. Маклафлін, Т. (2020). Веб-розробка за допомогою HTML, CSS і JavaScript. O'Reilly Media. ISBN: 978-1492079076.
10. Джон Дакетт. (2014). HTML і CSS: дизайн і створення веб-сайтів. Wiley. ISBN: 978-1118008188.
11. W3C. (2023). Специфікація HTML5. Консорціум World Wide Web. [Електронний ресурс] – Режим доступу: https://www.w3.org/TR/html5/
12. Мурач, Я. (2015). HTML5 і CSS3 від Murach. Murach & Associates. ISBN: 978-1943872111.
13. Шей, М. (2014). Програмування HTML5 Pro. Apress. ISBN: 978-1430244220.
14. Грассо, Дж. (2019). Повний посібник із CSS. Apress. ISBN: 978-1484253665.
15. W3Schools. (2023). Підручники з HTML і CSS. [Електронний ресурс] – Режим доступу: https://www.w3schools.com/
16. Макфарланд, Д. (2020). HTML і CSS: візуальний короткий посібник. Peachpit Press. ISBN: 978-0135264049.
17. SitePoint. (2017). HTML і CSS для початківців. SitePoint. [Електронний ресурс] – Режим доступу: https://www.sitepoint.com/
18. Самбротто, А. (2020). Вивчення веб-дизайну: посібник для початківців. O'Reilly Media. ISBN: 978-1492049161.
19. Документація Bootstrap. (2023). Bootstrap Framework. [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/
20. Парнелл, С. (2018). Створення веб-додатків за допомогою HTML5 і CSS3. Wiley. ISBN: 978-1119370766.

21. Мартінес, М. (2022). JavaScript для веб-розробників. O'Reilly Media. ISBN: 978-1449341378.
22. Вальдес, А. (2018). Технології та інструменти веб-розробки. Пірсон. ISBN: 978-0135452639.
23. Документація Laravel. (2023). Laravel Framework. [Електронний ресурс] – Режим доступу: https://laravel.com/docs
24. Мережа розробників Mozilla (MDN). (2023). Документація HTML, CSS і JavaScript. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web
25. Лука Врублевський. (2011). Mobile First. Книга окремо. ISBN: 978-1937557104.
26. Зельдман, Дж. (2010). Проектування за веб-стандартами. Нові Вершники. ISBN: 978-0321616953.
27. O'Reilly Media. (2022). JavaScript: хороші сторони. O'Reilly Media. ISBN: 978-0596517748.
28. Мохаммаді, М. (2021). Веб-дизайн для розробників: Посібник програміста щодо інструментів і методів дизайну. Wiley. ISBN: 978-1119249344.
29. Студія Bootstrap. (2023). Офіційний сайт Bootstrap Studio. [Електронний ресурс] – Режим доступу: https://bootstrapstudio.io/
30. Пауелл, Т. (2021). Вивчення шаблонів проектування JavaScript. O'Reilly Media. ISBN: 978-1449331812.

ДОДАТОК А

Створення бази даних та таблиць

```
use webConstructorApp_db

CREATE TABLE UsersDim (
    user_id INT PRIMARY KEY IDENTITY(1,1),
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    preferences TEXT,
    created_at DATETIME DEFAULT GETDATE()
);

CREATE TABLE ProjectsFact(
    project_id INT PRIMARY KEY IDENTITY(1,1),
    project_name VARCHAR(255) NOT NULL,
    description TEXT,
    created_at DATETIME DEFAULT GETDATE(),
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES UsersDim(user_id)
);

CREATE TABLE ComponentsDim (
    component_id INT PRIMARY KEY IDENTITY(1,1),
    component_name VARCHAR(255) NOT NULL,
    component_type VARCHAR(255) NOT NULL,
    properties TEXT
);

CREATE TABLE StylesFact (
    style_id INT PRIMARY KEY IDENTITY(1,1),
    font_size VARCHAR(50),
    color VARCHAR(50),
    margin VARCHAR(50),
    padding VARCHAR(50)
);
```

```
CREATE TABLE CSSRulesDim (
    rule_id INT PRIMARY KEY IDENTITY(1,1),
    css_selector VARCHAR(255) NOT NULL,
    css_properties TEXT,
    style_id INT,
    FOREIGN KEY (style_id) REFERENCES StylesFact(style_id)
);

CREATE TABLE TemplatesDim (
    template_id INT PRIMARY KEY IDENTITY(1,1),
    template_name VARCHAR(255) NOT NULL,
    template_type VARCHAR(255) NOT NULL,
    description TEXT
);

CREATE TABLE Project_Templates_Fact (
    project_template_id INT PRIMARY KEY IDENTITY(1,1),
    project_id INT,
    template_id INT,
    FOREIGN KEY (project_id) REFERENCES ProjectsFact(project_id),
    FOREIGN KEY (template_id) REFERENCES TemplatesDim(template_id)
);
```

ДОДАТОК Б

Фрагменти програмного коду. Функція аналізу даних та створення висновків

```
<?php

namespace App;

class PageElement
{
    private $db;

    public function __construct()
    {
        $this->db = Database::getInstance()->getConnection();
    }

    public function create(string $type, string $content, string $styles = '',
string $attributes = ''): bool
    {
        if (empty($type) || empty($content)) {
            throw new InvalidArgumentException("Type and content cannot be
empty.");
        }

        $sql = "INSERT INTO elements (type, content, styles, attributes) VALUES
(:type, :content, :styles, :attributes)";
        $stmt = $this->db->prepare($sql);
```

```
        $stmt->bindParam(':type', $type);
        $stmt->bindParam(':content', $content);
        $stmt->bindParam(':styles', $styles);
        $stmt->bindParam(':attributes', $attributes);

        try {
            return $stmt->execute();
        } catch (PDOException $e) {
            throw new RuntimeException("Failed to create page element: " . $e-
>getMessage());
        }
    }


class PageElement
{
    private $db;

    public function __construct()
    {
        $this->db = Database::getInstance()->getConnection();
    }

    public function create(string $type, string $content, string $styles = '',
string $attributes = ''): bool
    {
        if (empty($type) || empty($content)) {
            throw new InvalidArgumentException("Type and content cannot be
empty.");
        }

        $sql = "INSERT INTO elements (type, content, styles, attributes) VALUES
(:type, :content, :styles, :attributes)";
        $stmt = $this->db->prepare($sql);

        $stmt->bindParam(':type', $type);
        $stmt->bindParam(':content', $content);
        $stmt->bindParam(':styles', $styles);
        $stmt->bindParam(':attributes', $attributes);

        try {
            return $stmt->execute();
        } catch (PDOException $e) {
            throw new RuntimeException("Failed to create page element: " . $e-
>getMessage());
        }
    }
}
```

```php
$type = $_POST['type'] ?? '';
$content = $_POST['content'] ?? '';
$styles = $_POST['styles'] ?? '';
$attributes = $_POST['attributes'] ?? '';

try {
    $element = new PageElement();
    $isCreated = $element->create($type, $content, $styles, $attributes);

    if ($isCreated) {
        echo json_encode(['status' => 'success', 'message' => 'Element created
successfully.']);
    } else {
        echo json_encode(['status' => 'error', 'message' => 'Failed to create
element.']);
    }
} catch (Exception $e) {
    echo json_encode(['status' => 'error', 'message' => $e->getMessage()]);
}
```