DOI: https://doi.org/10.23939/10.23939/csn2024.02.250
УДК 004.5

FRONT-END ФРЕЙМВОРК ДЛЯ ПОБУДОВИ ЗАСТОСУНКІВ З АДАПТИВНИМ ГРАФІЧНИМ ІНТЕРФЕЙСОМ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ

І. В. Чаус, Т.А. Марусенкова

Національний університет "Львівська політехнка"
кафедра програмного забезпечення
E-mail: ivan.chaus.mnpzm.2023@lpnu.ua, tetiana.a.marusenkova@lpnu.ua

У статті розглядаються підходи до розробки front-end фреймворку для створення веб-застосунків з адаптивним графічним інтерфейсом, що динамічно підлаштовується під індивідуальні потреби користувачів за допомогою алгоритмів машинного навчання. Актуальність проблеми полягає в необхідності розробки інтерфейсів, здатних одночасно відповідати потребам різних демографічних груп, що вимагає гнучкості в налаштуванні користувацького досвіду (UX) та інтерфейсу (UI) сучасних веб-сайтів. Традиційні методи дизайну інтерфейсів не завжди враховують специфічні потреби кожного користувача, що знижує ефективність взаємодії з сайтом. У статті запропоновано підхід, який використовує алгоритми підкріпленого машинного навчання для аналізу взаємодії користувача з інтерфейсом і автоматичної адаптації інтерфейсу на основі поведінкових даних. Це дозволяє підвищити точність персоналізації інтерфейсів та покращити загальний користувацький досвід.

Метою роботи є розробка інструменту автоматизованої перебудови графічного інтерфейсу веб-застосунків залежно від індивідуальних потреб користувачів для покращення їхнього користувацького досвіду. Розроблено алгоритми для оптимізації взаємодії користувачів зі сторінками веб-застосунків та підвищення ефективності роботи з інтерфейсом.

Результати дослідження демонструють здатність фреймворку динамічно реагувати на поведінку користувачів, оцінювати їх рівень взаємодії та приймати обґрунтовані рішення щодо адаптації параметрів інтерфейсу, що, в свою чергу, допомагає розробникам зменшити обсяг роботи, необхідної для реалізації персоналізованого інтерфейсу, шляхом усунення потреби в ручній розробці варіантів веб-сайту. Використовуючи цей підхід, оцінене зменшення обсягу коду становить 40-50%.

Ключові слова: front-end, адаптивний інтерфейс, веб-дизайн, користувацький досвід, машинне навчання.

1. Вступ

У сучасній розробці веб-застосунків все більш актуальною стає проблема забезпечення ефективної взаємодії користувачів з графічними інтерфейсами. З огляду на різноманіття користувацьких аудиторій та зростаючі вимоги до персоналізації, традиційні підходи до створення інтерфейсів, засновані на фіксованих шаблонах, часто виявляються недостатньо ефективними. Вони не враховують індивідуальні потреби кожного користувача, що призводить до зниження загального рівня задоволення від взаємодії з веб-сайтом, меншої залученості та, в результаті, до втрати лояльності користувачів. Особливо це стосується великих веб-застосунків з різноманітною демографічною

аудиторією, де потреби користувачів можуть суттєво відрізнятися залежно від віку, професійної діяльності або цифрових навичок.

В умовах постійного розвитку технологій та вимог до індивідуалізації користувацького досвіду (UX), використання машинного навчання для автоматизації адаптації інтерфейсів стає особливо важливим. Технології машинного навчання дозволяють аналізувати поведінкові дані користувачів і на основі цього динамічно змінювати структуру та елементи інтерфейсу, покращуючи взаємодію та задовольняючи специфічні потреби кожного користувача. Це стає можливим завдяки алгоритмам, здатним не лише відстежувати дії користувача, але й адаптувати інтерфейс у реальному часі з урахуванням отриманих даних.

Актуальність проблеми полягає в тому, що сучасні веб-застосунки потребують ефективних інструментів для підвищення залученості користувачів, оптимізації користувацького досвіду та покращення загальної продуктивності системи. Розробка front-end фреймворку, який використовує алгоритми підкріпленого машинного навчання для перебудови графічного інтерфейсу, дозволяє вирішити ці завдання, що відкриває нові можливості як для наукових досліджень у галузі UX/UI, так і для практичної розробки сучасних веб-застосунків. Впровадження таких підходів не лише покращує якість взаємодії з користувачами, але й сприяє підвищенню їхньої задоволеності та тривалості користування ресурсами.

2. Огляд літературних джерел

У сучасному цифровому середовищі, де веб-сайти стають важливою складовою щоденного життя, розробка їхнього графічного інтерфейсу відіграє критичну роль. UI-дизайн визначає спосіб, яким користувачі сприймають та взаємодіють з веб-платформами, відображаючи вимоги сучасного веб-дизайну до естетики, інтуїтивності та ергономіки інтерфейсу. Напрямки розвитку UI впливають на сприйняття продукту або сервісу [1], а також на користувацьке задоволення та лояльність. Ефективний UI-дизайн забезпечує чітку навігацію, зворотний зв'язок та допомогу користувачам у вирішенні завдань і проблем, мінімізуючи помилки та відволікання.

Згідно з [2], дизайн графічного інтерфейсу користувача є ключовим фактором, що визначає залучення користувачів до веб-застосунків для онлайн-шопінгу. Якість взаємодії між користувачем і застосунком є критичною, оскільки користувачі віддають перевагу платформам, які надають зручний і довірчий досвід, а також ефективно організовують контент на сторінці.

Іншим ключовим аспектом якісної взаємодії з веб-сайтом є його ергономічність. Згідно з [3], стандартні підходи до дизайну веб-сайтів, такі як ефективність, естетика та простота використання, не завжди забезпечують оптимальний користувацький досвід. Особливості інтерфейсу безпосередньо впливають на психіку користувача та його результативність під час використання продукту. Тому важливо враховувати ергономічні теорії та методи дослідження при проектуванні інтерактивного інтерфейсу користувача. Часто розробники не приділяють достатньо уваги принципам ергономіки, що може впливати на ефективність інтерфейсу. Включення цих принципів сприяє покращенню якості взаємодії та швидкості освоєння продукту користувачами.

Останні дослідження [4] підкреслюють важливість адаптації інтерфейсу веб-сайтів, зокрема пошукових систем, для специфічних демографічних груп, таких як люди віком від 60 років. Наприклад, в контексті Google, який є найпопулярнішим глобальним пошуковим сервісом, виявлено, що його інтерфейс не завжди відповідає потребам старшого покоління, що використовує його менш інтенсивно. Зміни в інтерфейсі, спрямовані на покращення користувацького досвіду для цієї групи, можуть значно підвищити ефективність взаємодії та задоволення від користування сайтом.

Для оптимізації інтерфейсу, що б задовольняв більшу кількість користувачів, рекомендується використовувати індивідуалізований графічний інтерфейс. Дослідження показує, що такий підхід сприяє покращенню якості взаємодії з системою [5]. Для максимізації активності користувачів на веб-сайті можна розглянути два підходи: розробку універсальної системи, що враховує різні демографічні групи, або поступове налаштування інтерфейсу відповідно до індивідуальних особливостей користувацької взаємодії.

Іншим підходом є повна делегація створення веб-сайту штучному інтелекту за допомогою генеративної змагальної мережі (GAN) [6]. Цей підхід є подібним до попереднього тим, що також спочатку виконуємо збір даних або використовуємо наявні набори даних, навчаємо модель. Проте замість етапу перебудови система проводить нову генерацію сторінки, після чого виконується оцінка оновленої версії. Такий підхід є значно гнучкішим, проте він містить чимало невизначеності. Зокрема, розробник не матиме можливість керувати, яким чином перебудовуватиметься сторінка, і в результаті може виникнути ситуація, коли користувачу не відображаються всі елементи сторінки, оскільки для алгоритму перебудови сторінки їхнє відображення видається недоцільним. Також розробник не матиме можливості надавати елементам пріоритети або вказувати допустимі межі до адаптації, що хоч і може призвести до кращих показників залученості користувача з окремими елементами сторінки та підвищеної загальної задоволеності роботою сайту, проте не існує гарантії, що в такому випадку користувачу буде відображатися саме потрібний контент сторінки.

Важливо забезпечити поступові та ретельно обґрунтовані зміни в дизайні веб-сайту, оскільки раптові зміни можуть негативно вплинути на сприйняття кінцевого продукту користувачами. Розробка ефективної моделі винагородження адаптивного користувацького інтерфейсу (AUI) з використанням навчання з підкріпленням [7] у поєднанні з HCI дозволяє оптимізувати взаємодію користувачів з інтерфейсом. Особливо цінною є можливість отримання відгуків від користувачів, що сприяє швидкому тренуванню моделі. Проте важливо уникати частого запиту на відгуки, що може негативно позначитись на загальному користувацькому досвіді (UX). Тому перевага надається збору якісних даних з HCI для оптимального навчання моделі.

Для ефективної перебудови дизайну необхідно збирати дані про активність користувачів. Одним з найефективніших методів є відстеження погляду користувача, але цей підхід часто недоступний через відсутність відповідного обладнання. В таких випадках можна використовувати альтернативні методи, наприклад, аналіз виділення тексту [8], який дозволяє визначити, які елементи сторінки привертають увагу користувачів на основі їх текстових відміток. Для вивчення патернів навігації та взаємодії також корисний аналіз рухів мишки [9], який, хоч і менш точний, все ж надає цінну інформацію про поведінку користувачів на веб-сторінці.

3. Постановка задачі

Дослідження спрямоване на розробку front-end фреймворку для індивідуальної адаптації графічного інтерфейсу веб-сайтів з метою покращення користувацького досвіду. Основні завдання включають розробку алгоритму для персоналізованої перебудови сторінки після оцінки рівня якості взаємодії користувачів з нею.

4. Створення середовища для розробників

Цей розділ описує процес створення front-end фреймворку, зокрема акцентуючи увагу на розробці екосистеми для інтуїтивного налаштування інтерфейсу без потреби в технічних деталях та забезпеченні зворотної сумісності з існуючими веб-сайтами. Основна ідея полягає в інтеграції адаптивних елементів і контейнерів, які можуть динамічно налаштовуватися для покращення взаємодії користувачів, зберігаючи при цьому стабільність і функціональність існуючих ресурсів.

З огляду на те, що основною метою є створення front-end фреймворку, першочерговим завданням є розробка екосистеми, яка забезпечить розробнику можливість інтуїтивно керувати ключовими параметрами інтерфейсу без необхідності занурюватися у технічні деталі реалізації самого інструменту. Така екосистема повинна пропонувати гнучкі та зрозумілі механізми конфігурації, що дозволить спростити процес інтеграції та налаштування фреймворку, зберігаючи при цьому його функціональну гнучкість та ефективність.

Ще одним ключовим аспектом розробки є забезпечення зворотної сумісності фреймворку з існуючими веб-сайтами. Це означає, що адаптивний фреймворк повинен підтримувати інтеграцію свого функціоналу без необхідності повного переписування структури або коду вже створених веб-ресурсів. Такий підхід забезпечить плавний перехід до використання нового інструменту, мінімізуючи

ризики втрати функціональності або продуктивності сайту. Це також дозволить розробникам легко впроваджувати нові технології та інструменти в уже існуючі проекти, зберігаючи при цьому їхню стабільність та сумісність з поточними стандартами і технологіями.

Врахувавши вищезазначені вимоги, можемо сформувати фреймворк, що відповідає архітектурі, наведеній на рис. 1.

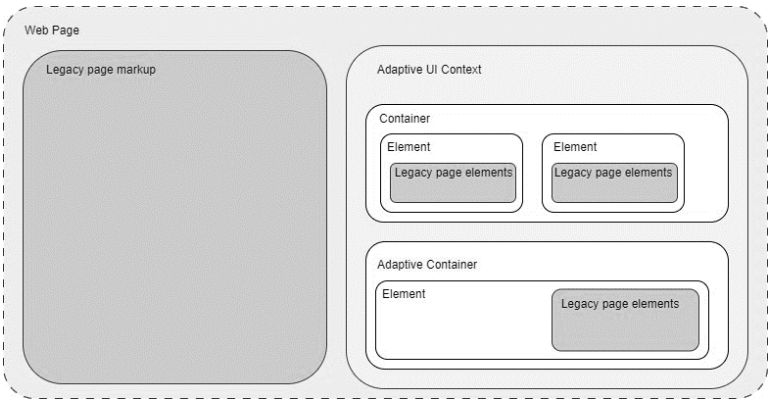

Рис. 1. Архітектура сайту з інтегрованим адаптивним фреймворком

Для реалізації функціоналу адаптивного фреймворку вся попередня розмітка веб-сторінки повинна бути інтегрована в спеціалізований контекст. Цей контекст не впливатиме на функціональність чи візуальну структуру інтерфейсу сайту, оскільки його основна мета полягає у збиранні та збереженні даних, необхідних для тренування моделі адаптації. Контекст дозволяє розробнику гнучко налаштовувати параметри збору даних, включаючи обсяг активності користувача, який має бути досягнутий перед початком тренування моделі. Додатково, розробник може вказати тип оптимізаційної функції, наприклад, AGRAD або ADAM, для оптимального навчання моделі, що підходить до специфіки проєкту.

Важливо зазначити, що немає потреби обгортати контекстом весь веб-сайт. Контекст застосовується лише до тих частин сторінки, на яких планується використовувати функціонал адаптивного графічного інтерфейсу. Це підвищує ефективність системи, мінімізуючи обсяг оброблюваних даних, і дозволяє точніше налаштувати фреймворк під конкретні ділянки інтерфейсу, де важлива індивідуалізація користувацької взаємодії.

Після ініціалізації контексту фреймворку, розробник може будувати адаптивний інтерфейс за допомогою доступних йому елементів:

1. Фіксовані контейнери слугують ключовими структурними елементами для організації та обгортки основного контенту веб-сторінки. З технічної точки зору, ці контейнери реалізуються як HTML-елементи, такі як , які можуть бути прив'язані до унікальних ідентифікаторів для забезпечення індивідуального контролю. Важливо, що кожен контейнер повинен включати тег, що визначає тип його вмісту, зокрема "interactable" для елементів, з якими користувачі можуть взаємодіяти, або "read-only" для статичного контенту. Крім того, контейнери оснащуються трекерами подій, які відстежують різні аспекти поведінки користувачів у межах цих контейнерів. Серед таких параметрів є кількість часу, проведеного користувачем у взаємодії з контейнером, тип та кількість цих взаємодій, а також відстеження руху мишки в межах контейнера. Важливо також відзначити, що контейнери мають фіксовану стилізацію, що означає, що вони не підлягають динамічній перебудові в процесі роботи. Їм можуть бути передані лише початкові стилі, які повинні відповідати вимогам responsive-дизайну для забезпечення коректної адаптації інтерфейсу на різних пристроях і екранах. Це забезпечує стабільність і прогнозованість поведінки контейнерів у межах веб-сайту, а також гарантує, що їхній зовнішній вигляд зберігатиметься незалежно від змін в інших компонентах сторінки.

2. Елементи у даному фреймворку виконують ключову роль у реалізації адаптивного графічного інтерфейсу. Вони є найменшою структурною одиницею системи і, на відміну від

контейнерів, можуть існувати без дочірніх елементів чи контейнерів (крім стандартних HTML елементів). Для їхньої коректної роботи на сторінці, елементи повинні бути обгорнуті у контейнер, який визначає межі перебудови сторінки. Це рішення спрямоване на запобігання хаотичній зміні структури сторінки, яку може згенерувати алгоритм перебудови, а також надання розробникам можливості контролю над тим, як і до якого ступеня алгоритм буде змінювати графічний інтерфейс користувача. Наприклад, дотримання елементами меж контейнера у процесі перебудови інтерфейсу демонструється на рис. 2.

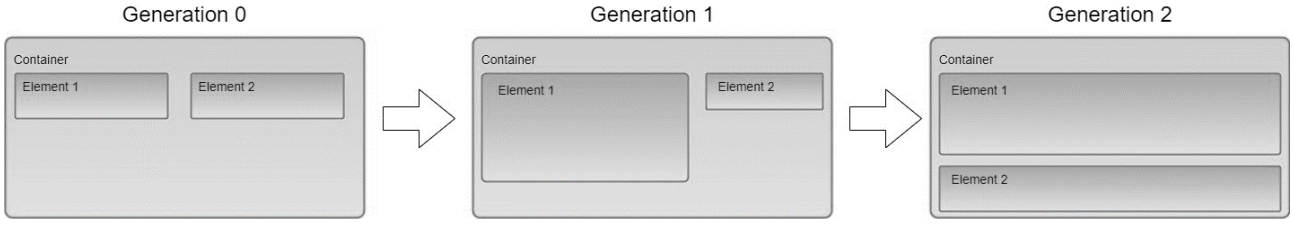

Рис. 2. Приклад дотримання елементами меж контейнера в процесі перебудови графічного інтерфейсу

Подібно до стандартних HTML елементів, адаптивні елементи мають власні стилі, проте їхня поведінка дещо відрізняється. Стилі адаптивних елементів розглядаються швидше як рекомендації від розробника, які слугують орієнтиром для алгоритму під час перебудови інтерфейсу, а не як жорсткий набір правил. Окрім базових стилів, розробник може вказати також граничні стилі, що дозволяє контролювати межі, в яких працюватиме алгоритм перебудови сторінки. Це зроблено для запобігання втраті важливого вмісту сторінки у випадку, якщо алгоритм вирішить, що відображення певних елементів є недоцільним. Такий підхід забезпечує баланс між автоматизованою адаптацією інтерфейсу і збереженням цілісності та функціональності сторінки.

3. Адаптивні контейнери функціонують як аналоги фіксованих контейнерів і виконують схожу роль, слугуючи обгорткою та базовою структурою для побудови адаптивного користувацького інтерфейсу. Однак, їхньою ключовою відмінністю є здатність до динамічної перебудови в процесі навчання моделі. Зокрема, адаптивні контейнери можуть змінювати свої стилі та приймати граничні стилі як параметри, що забезпечує більшу гнучкість у налаштуванні зовнішнього вигляду та поведінки інтерфейсу відповідно до змін у користувацькій взаємодії або умовах системи. Для забезпечення стабільності і запобігання небажаним змінам у макеті або позиціонуванні елементів, адаптивні контейнери повинні бути розміщені всередині фіксованого контейнера. Така конструкція дозволяє уникнути хаотичної перебудови інтерфейсу та зберігати його структурну цілісність, що особливо важливо в умовах частих змін у вхідних даних або поведінці користувачів.

5. Засоби відстеження та оцінювання поведінки користувача

У цьому розділі йдеться про вибір методів для оцінки взаємодії користувачів з веб-сторінкою, зокрема про порівняння явного і неявного підходів до збору даних. Основна ідея полягає в застосуванні неявного методу збору даних, зокрема через відстеження активності за блоками сторінки, для створення точних оцінок користувацького досвіду, що дозволяє оптимізувати дизайн інтерфейсу без втручання в процес взаємодії користувачів.

Для того, щоб модель перебудови базувалась на справжніх даних, а не працювала за принципом довільної перебудови, розглянуто та обрано необхідні методи відстеження та формування оцінки про взаємодію користувача з певними секціями веб-сторінки.

Для оцінки користувацького досвіду існують два основні підходи: явний і неявний. Явний підхід передбачає активне отримання зворотного зв'язку через опитування або анкетування, що дає точні дані, але може дратувати користувачів і знижувати кількість відповідей. Неявний підхід заснований на автоматичному зборі поведінкових даних, таких як рух мишки, виділення тексту та час,

проведений на сторінці, що дозволяє оцінювати користувацький досвід без втручання в процес взаємодії. Проте, цей метод може призводити до помилкових інтерпретацій, оскільки деякі дії користувачів можуть бути викликані не зацікавленістю, а складністю взаємодії з інтерфейсом. Неявне оцінювання дозволяє використовувати більше даних для автоматизованої перебудови інтерфейсу, що підвищує його адаптивність. Водночас, помилки у трактуванні даних можуть бути виправлені на наступних етапах оптимізації. Для кращої інтерпретації зворотного зв'язку можуть застосовуватися методи обробки природної мови, такі як великі мовні моделі (LLM), які дозволяють аналізувати текстові відповіді користувачів. Зважаючи на мету мінімізації складності інтеграції програмного забезпечення в існуючі проєкти, нами прийнято рішення використовувати неявний метод збору оцінок користувацького досвіду.

Для створення теплової карти веб-сайту на основі руху мишки необхідний спеціальний алгоритм, який відстежує динаміку переміщення курсора. Такий підхід дозволяє візуалізувати зони інтенсивної активності користувачів, що сприяє оптимізації дизайну та покращенню користувацького досвіду. Алгоритм фіксує події «onMouseMove» та відслідковує координати курсора. У разі відсутності руху мишки перевіряється поточна позиція через певні інтервали. Теплові карти, створені на основі цих даних, відображають області сторінки з найбільшою користувацькою активністю. Вони допомагають виявити елементи, що привертають увагу, та зони, які потребують покращення. Ці карти також сприяють налагодженню алгоритмів перебудови сторінки, наочно демонструючи їхню ефективність. Додатковою перевагою є можливість аналізу траєкторії курсора, що дозволяє ідентифікувати елементи, які відволікають користувачів від основної мети, і покращити навігацію. Проте підхід має недоліки. Зміни в роздільній здатності екранів можуть спричинити зсув точок уваги користувачів на сторінці, що ускладнює інтерпретацію теплової карти. Крім того, велика кількість подій «onMouseMove», які генеруються під час рендерингу, значно збільшує обсяг зібраних даних, що може створювати проблеми для зберігання та обробки інформації.

Альтернативним методом відстеження активності користувачів на веб-сайті є відстеження активності за блоками сторінки. Цей підхід схожий на алгоритм аналізу уваги користувача до окремих слів у тексті [10], оскільки він також базується на поділі сторінки на зони уваги. Він дозволяє розробникам визначати ключові елементи сторінки або встановлювати пріоритети для конкретних сегментів і відстежувати взаємодію користувачів із ними. Однією з переваг цього підходу є можливість відстежувати тривалість взаємодії з окремими сегментами без потреби у зберіганні великих обсягів даних у локальному сховищі. Також цей метод не має вад, які присутні в методі теплокарт, оскільки дані про взаємодію зберігаються стосовно елементів, а отже проблеми з координатами точок уваги та обсягами даних відсутні. Однак, цей метод має і певні недоліки. Основним із них є обмеження у створенні детальних теплових карт, які дозволяють візуалізувати зони інтенсивної взаємодії користувачів зі сторінкою. Крім того, відсутність можливості відстеження траєкторії руху курсора ускладнює оцінку рівня відволікання користувачів від основного змісту веб-сторінки [11].

Зважаючи на переваги та недоліки розглянутих методів, нами обрано метод відстеження активності користувачів за блоками. Усі дані про активність зберігатимуться в сховищі localStorage, оскільки воно дозволяє зберігати більші обсяги інформації порівняно з cookies і забезпечує збереження даних між сесіями. Це дозволяє використовувати інформацію з попередніх сесій для навчання моделі. Дані зберігаються у форматі мапи, де ключами виступають унікальні ідентифікатори елементів, а значеннями – масиви подій, згрупованих за сесіями. Нижче наведено приклад збережених даних про активність користувача:

```
activity_data: {
   element_id: [
      {
          sessionId: “2822b0b2-deff-b7fe-b7ca3c00b5”,
           entries: [...],
          frameworkGeneration: 0,
           prioGroup: 1,
           contentInteractionType: “view-only”,
           type: “element”
       }, ...
   ], ...
```

}

Тренування моделі та відповідна перебудова веб-сторінки здійснюються лише після накопичення достатньої кількості даних про активність користувача. Цей мінімальний обсяг даних може бути налаштований розробником під час ініціалізації контексту фреймворку. За замовчуванням, обсяг складає 100 записів про взаємодію з усіма елементами сторінки. Такий підхід дозволяє оптимізувати продуктивність, зокрема прискорити завантаження сторінки, оскільки модель не буде тренуватися після кожної взаємодії. Крім того, це запобігає можливій втомі користувача від постійних змін інтерфейсу, які могли б негативно вплинути на користувацький досвід. Такий механізм гарантує стабільність інтерфейсу, дозволяючи збирати достатньо даних для тренування моделі без зниження зручності використання веб-сайту.

Після того, як дані було успішно зібрано, на їх основі формуються часовий індекс g_{time}(1) та індекс взаємодії $g_{interaction}$(2), які вираховуються як відношення взаємодій з елементом до загальної кількості взаємодій на сторінці.

$$g_{time} = \frac{t_{element}}{t_{session}} \quad (1) \qquad g_{interaction} = \frac{n_{element}}{n_{session}} \quad (2)$$

Фінальна оцінка взаємодії за сесією формується у вигляді добутку g_{time} та $g_{interaction}$, за винятком ситуацій, коли елемент не має точок взаємодії користувача (тобто, його тип є “view-only”). У такому випадку оцінка взаємодії з елементом є рівною g_{time}.

Оцінки взаємодії користувачів формуються на початку нової сесії, що зумовлено обмеженнями браузера щодо виконання функцій при завершенні сесії. Ці оцінки зберігаються у локальному сховищі у вигляді мапи, де ключем виступає унікальний ідентифікатор кожного елемента інтерфейсу, а значенням є об'єкти, які включають ідентифікатор сесії та відповідну оцінку взаємодії з елементом. Такий підхід дозволяє ефективно зберігати і обробляти дані, забезпечуючи надійний механізм збору інформації для подальшого аналізу користувацького досвіду і тренування моделі адаптивного інтерфейсу.

6. Опис обраної моделі машинного навчання

У цьому розділі розглядається процес вибору моделі машинного навчання для вирішення задачі оптимізації користувацького інтерфейсу, а також наводиться детальний опис обраної моделі. Процес вибору моделі включав аналіз кількох підходів до навчання, зокрема регресійних моделей, дерев рішень, ансамблевих методів і нейронних мереж. Оскільки задача передбачає обробку великої кількості параметрів і побудову складних взаємозв'язків між стилями інтерфейсу та оцінками користувацької взаємодії, було прийнято рішення на користь використання нейронної мережі.

Основною причиною вибору нейронної мережі стало те, що вона здатна ефективно працювати з багатовимірними вхідними даними та виявляти нелінійні залежності між параметрами. Це робить її придатною для задачі, де кількість і складність взаємодій користувачів з інтерфейсом постійно змінюється.

Модель побудована на основі класичної послідовної архітектури нейронної мережі зокрема використано підхід згорткової нейронної мережі(CNN), що складається з одного вхідного шару, трьох прихованих шарів і одного вихідного шару, який передбачає два вихідні параметри.

Згорткова нейронна мережа (CNN) є оптимальним вибором [12] для вирішення даної задачі через кілька ключових особливостей, які роблять її ефективною при роботі з даними про користувацьку взаємодію з інтерфейсом. Основною перевагою CNN є її здатність автоматично виявляти локальні залежності в даних, що є критичним для аналізу складних багатовимірних взаємодій. У випадку роботи з інтерфейсами, ці залежності можуть відображати зв'язок між конкретними елементами дизайну та користувацькою поведінкою, що дозволяє мережі визначати, як зміни в стилях впливають на взаємодію користувачів.

Крім того, CNN може ефективно працювати з просторовими даними, навіть якщо мова не йде про традиційні зображення. Взаємодія користувачів з веб-інтерфейсом часто має схожі просторові

закономірності: розташування елементів, їхній контекст, а також спосіб, у який користувачі пересуваються по сторінці, мають значення для розуміння поведінки користувачів. Згорткові шари в CNN дозволяють моделі виділяти такі закономірності без необхідності явного зазначення цих зв'язків у даних, що робить її більш гнучкою та точною у виявленні залежностей.

Такий підхід дозволяє моделі ефективно обробляти багатовимірні вхідні дані, які містять різні параметри стилів інтерфейсу, і відповідно визначати пріоритети для кожного елемента на основі користувацької взаємодії. Кожен прихований шар цієї нейронної мережі використовує функцію активації ReLU (Rectified Linear Unit), що є стандартним підходом у сучасних нейронних мережах через її здатність швидко обчислювати градієнти і підтримувати нелінійність моделі.

Перший прихований шар містить 64 нейрони, другий — 32, а третій — 16 нейронів. Вихідний шар складається з 2 нейронів, що забезпечують можливість моделі приймати рішення на основі двох ключових параметрів, таких як групи пріоритетності елементів інтерфейсу або специфічні оцінки користувацької взаємодії. Така структура дозволяє моделі нейронної мережі виконувати глибоке навчання та оптимізацію інтерфейсу шляхом адаптації стилів у відповідь на змінювані потреби користувачів. Завдяки послідовній архітектурі з декількома прихованими шарами, модель може обробляти складні залежності між користувацькими діями і параметрами інтерфейсу, що підвищує її здатність до навчання і точність прогнозів. Архітектуру цієї нейронної мережі наведено на рис. 3.

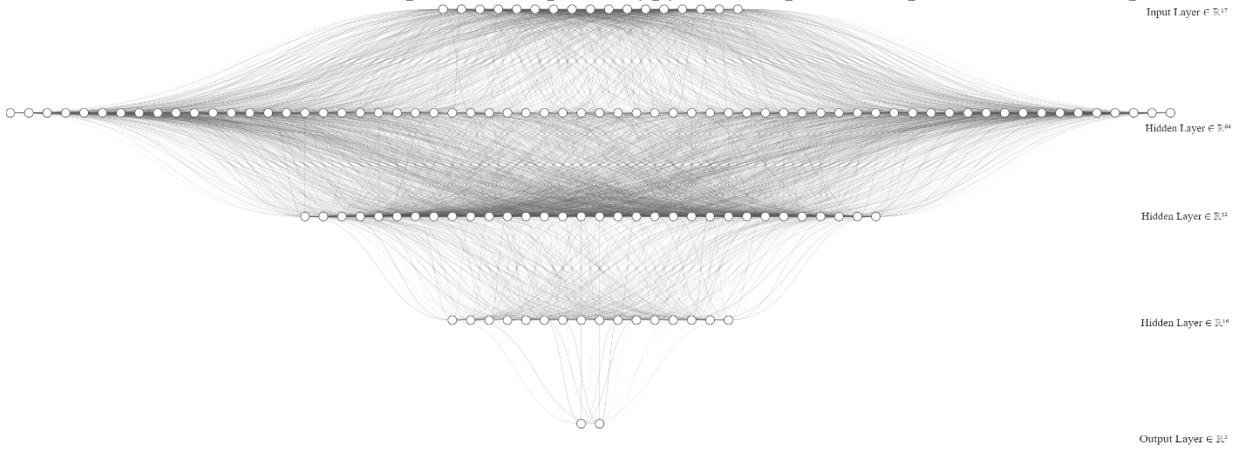

Рис. 3. Структура нейромережі моделі

7. Застосування нейронної мережі для формування стилів

У цьому розділі розглядається процес застосування нейронної мережі для генерації нових стилів елементів інтерфейсу на основі аналізу взаємодії користувачів. Використання нейронної мережі в даній задачі обумовлено її здатністю адаптивно змінювати стилі елементів залежно від реальних даних, зібраних під час користування веб-інтерфейсом. Процес генерації нових стилів за допомогою нейромережі передбачає кілька ключових етапів, кожен з яких спрямований на покращення користувацького досвіду за рахунок гнучкої і персоналізованої адаптації інтерфейсу.

Спочатку модель нейронної мережі отримує вхідні дані, які включають як характеристики поточного стилю елементів інтерфейсу, так і дані про користувацьку поведінку, такі як частота взаємодій, час затримки на елементах, рівень залученості тощо. Ці дані подаються на вхід нейронної мережі, де кожен прихований шар послідовно обробляє інформацію, виділяючи найбільш значущі взаємозв'язки між стилями елементів і реакцією користувача. Завдяки архітектурі з кількох шарів, модель здатна аналізувати як прості, так і складні, багатовимірні взаємозалежності. Детальніший опис процесу тренування моделі подано у 8 розділі цієї статті.

Після завершення етапу тренування моделі, наступним кроком є модифікація стилів веб-інтерфейсу відповідно до отриманих даних. Для цього застосовується спеціальна функція-оптимізатор, яка відіграє ключову роль у процесі коригування параметрів стилів на основі результатів прогнозування моделі. Функція-оптимізатор призначена для того, щоб адаптувати параметри стилів елементів інтерфейсу таким чином, щоб вони найкраще відповідали очікуванням користувачів, враховуючи динамічні зміни їх взаємодії з інтерфейсом. Процес адаптації стилів спрямований на

мінімізацію розбіжностей між передбаченими моделлю оцінками та реальними оцінками ефективності інтерфейсу, що дозволяє досягти більш точних і відповідних рішень щодо дизайну.

Розробник може самостійно вибирати конкретний оптимізатор залежно від характеру даних та задачі, проте за замовчуванням рекомендується використовувати алгоритм AdaGrad. Цей алгоритм є особливо ефективним у випадках, коли дані характеризуються розрізненістю або нестаціонарністю, оскільки AdaGrad адаптивно змінює швидкість навчання для кожного параметра, враховуючи історію його змін. Завдяки цьому алгоритм дозволяє більш ефективно працювати з рідкісними подіями чи даними, що зустрічаються нечасто, що є важливою характеристикою при роботі з неоднорідними користувацькими взаємодіями.

Також було розглянуто функцію-оптимізатор Adam. Для коректної роботи, ця функція-оптимізатор потребує більш детерміністичних результатів від нейронної мережі, а отже необхідні більші обсяги тренувальних даних для нейромережі. Проте результати роботи даної функції є більш точними, хоч і вимагають більше часу для їх отримання. Розробникам дозволено вибирати який оптимізатор вони бажають використовувати з врахуванням специфіки їхньої роботи.

Основна мета функції-оптимізатора полягає в тому, щоб мінімізувати помилку, яка виникає між передбаченими нейронною мережею оцінками для елементів інтерфейсу і реальними оцінками, отриманими на основі користувацької взаємодії. Для цього процес мінімізації виконується за допомогою середньоквадратичного відхилення (MSE) — одного з найпоширеніших методів у задачах регресії. MSE дає змогу обчислити середнє значення квадратів різниць між передбаченими моделлю та фактичними значеннями, що дозволяє точно оцінити відхилення моделі від реальних даних. Цей підхід забезпечує високу точність оцінки ефективності інтерфейсу і дозволяє вчасно виявляти слабкі місця в стилістичних параметрах елементів.

Після того, як функція-оптимізатор визначає рівень помилки, вона здійснює корекцію стильових параметрів елементів інтерфейсу. Цей процес включає внесення змін до таких характеристик, як колір, розмір, шрифти та інші візуальні елементи, щоб зробити інтерфейс більш зручним та інтуїтивно зрозумілим для користувача. Діаграма даного процесу наведена на рис. 4.

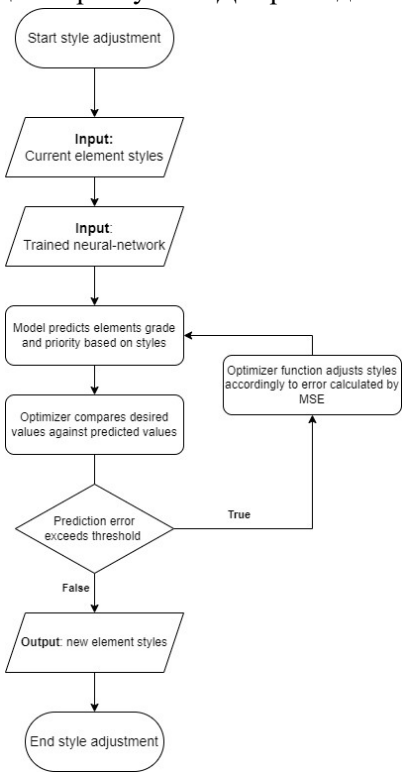

Рис. 4. Блок-схема процесу генерації нових стилів для елементу графічного інтерфейсу веб-сторінки.

8. Тренування моделі

У цьому розділі детально розглядається процес тренування нейронної мережі, що є ключовим етапом для забезпечення її ефективного функціонування. Тренування нейронної мережі полягає у поступовому навчанні моделі на основі вхідних даних, що дозволяє їй адаптуватися до розв'язання конкретної задач

Тренування моделі відбувається наступним чином: вхідні дані у вигляді стилів, застосованих до елементів інтерфейсу, та раніше сформовані оцінки користувачів щодо цих елементів розподіляються на навчальну та тестову вибірки у співвідношенні 4:1. Такий підхід гарантує, що модель може навчатися на більшості даних, а менша частина використовується для перевірки її здатності узагальнювати і робити точні прогнози на невідомих раніше зразках. Це дозволяє оцінити, наскільки добре модель може передбачати оцінки та визначати пріоритетну групу елементів на основі наданих стилів.

Під час тренування модель намагається встановити відповідність між стильовими параметрами елементів і їхніми оцінками, використовуючи методи градієнтного спуску для мінімізації різниці між передбаченими значеннями та фактичними оцінками. Основна мета — навчитися передбачати оцінку елементів інтерфейсу та їхню приналежність до певних пріоритетних груп, що дозволить у майбутньому адаптувати стилі для покращення користувацького досвіду. Модель аналізує залежності між різними параметрами стилів, як-от колір, розмір, відступи, та оцінює, які саме параметри найбільше впливають на сприйняття елементів користувачем.

Процес створення модифікованих стилів для елементів інтерфейсу виконується на початку кожної нової сесії користувача. Після генерації стилів, адаптованих на основі попередньо зібраних даних про взаємодію користувача з веб-сайтом, вони зберігаються у спеціальному сховищі. Це сховище слугує джерелом, звідки елементи сторінки отримують відповідні стилі, використовуючи свої унікальні ідентифікатори. Такий механізм гарантує, що кожен елемент інтерфейсу застосовує індивідуально модифіковані стилі, що відображають результати оптимізації моделі. Однак, у випадку, якщо в сховищі відсутні модифіковані стилі для певного елемента, відбувається автоматичне присвоєння початкових стилів, заданих розробником. Це може статися в кількох ситуаціях: коли елемент не отримав достатньо користувацької взаємодії для тренування моделі або коли користувач не досяг мінімального порогу взаємодії, необхідного для створення коригованих стилів. Мінімальний поріг визначається кількістю взаємодій, які є критичними для збору достовірних даних і ефективної роботи алгоритму навчання.

Важливо зазначити, що процес збору оцінок враховує саме ті стилі, які були застосовані до елементів інтерфейсу під час поточної сесії користувача. Тобто, якщо модель вирішила змінити певні параметри стилю елемента, ці зміни також фіксуються в сховищі активності користувача. Це дозволяє зберігати інформацію про модифіковані стилі для використання в подальших сесіях. У наступних сесіях ці стилі стають частиною даних для оцінювання, на основі яких формується подальше навчання моделі. Таким чином, нейронна мережа тренується на стилях, які були адаптовані в процесі її роботи, але враховує оцінки користувачів для цих стилів. Це реалізує принцип підкріпленого навчання, де модель отримує зворотний зв'язок у вигляді "винагород" або "покарань" залежно від того, позитивно чи негативно оцінюються користувачем внесені зміни. Винагорода надається, коли зміни стилів покращують користувацький досвід, а покарання – коли зміни призводять до погіршення взаємодії. Такий механізм забезпечує динамічну корекцію стилів на основі реальних даних і поведінкових реакцій користувачів, що дозволяє моделі постійно вдосконалюватися. Діаграму послідовності процесу тренування та модифікації стилів наведено на рис. 5.

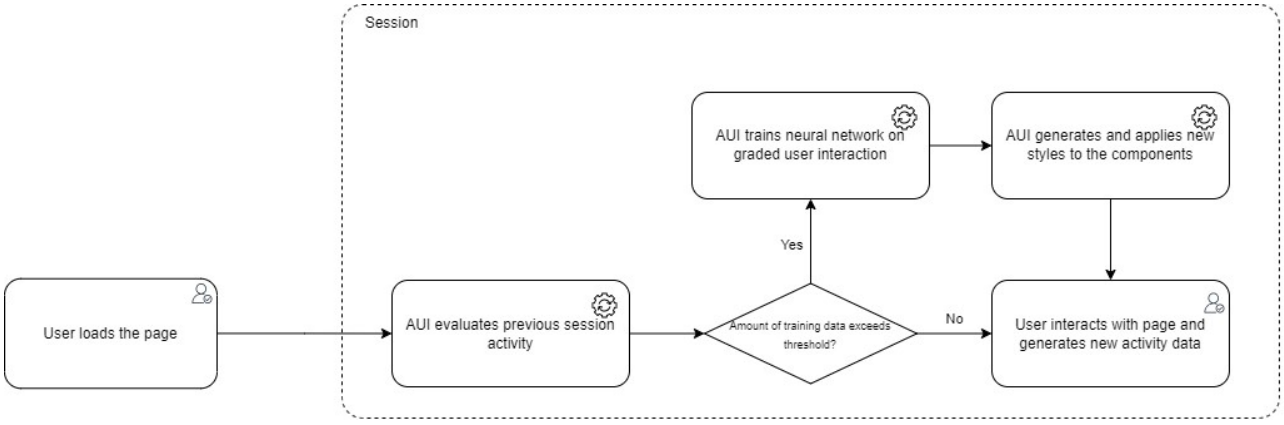

Рис. 5. Процес тренування та застосування стилів моделі

9. Результати досліджень

Для оцінки працездатності розробленого фреймворку проведено тестування на чотирьох веб-сторінках. Три з цих сторінок створені «з нуля» з метою оптимізації під специфіку роботи фреймворку, що дозволило провести детальне дослідження його можливостей в умовах контрольованого середовища. Четверта веб-сторінка представляла собою вже існуючий веб-сайт, до якого фреймворк було інтегровано без внесення значних змін у вихідний код, що дало змогу перевірити його зворотну сумісність та здатність до інтеграції в реальні проекти. Обрані веб-сторінки включали сайт туристичного агентства, сайт для пошуку роботи, платформу для прогнозу погоди та новинний портал, що представляло широкий спектр сценаріїв взаємодії користувачів з різними типами контенту та функціональними компонентами.

На рис. 6 представлені результати адаптації фреймворку після збору достатнього обсягу даних про користувацьку взаємодію. Зокрема, можна спостерігати, що фреймворк виявив підвищену активність користувачів з елементами, які використовують більший розмір шрифту, що призвело до їхнього подальшого збільшення. Крім того, блоки країн були автоматично збільшені в розмірах та позиціоновані більш рівномірно, оскільки їхній контент привернув значну увагу користувачів. Однак, кнопка замовлення, зважаючи на низький індекс взаємодії з елементами з більшими внутрішніми відступами, отримала зменшені відступи, що стало результатом аналізу поведінки користувачів, які надавали перевагу компактнішим елементам інтерфейсу.

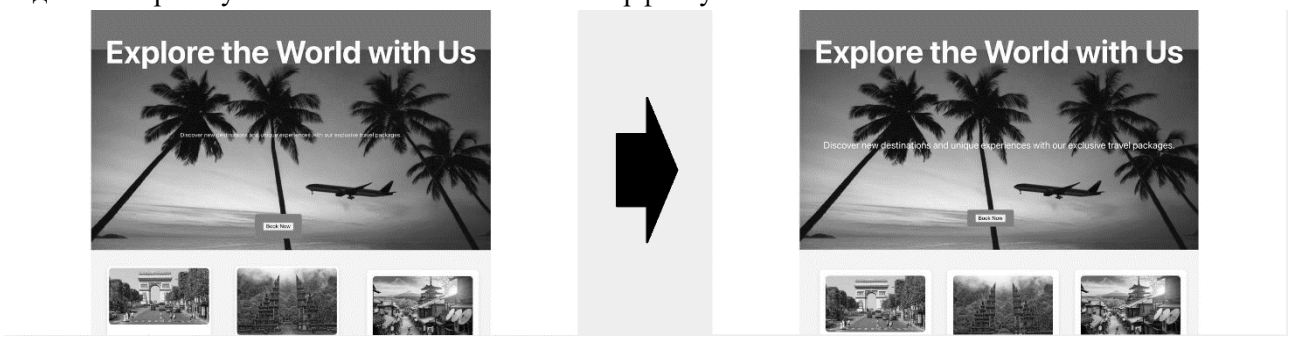

Рис. 6. Зміни внесені фреймворком до інтерфейсу сайту туристичного агентства

На рис. 7 продемонстровано зміни, внесені фреймворком до дизайну сайту пошуку роботи після аналізу користувацької взаємодії. Однією з помітних проблем початкового дизайну стала кнопка пошуку, яка, ймовірно, отримала низький індекс взаємодії через недоліки у візуальному оформленні. На основі отриманих даних фреймворк вирішив застосувати стилі, аналогічні тим, що використовувалися для карток з вакансіями, зокрема, вирівнювання кнопки по правому краю та зменшення внутрішніх відступів. Крім того, загальна тенденція до мініатюризації поширилася на інші ключові елементи інтерфейсу, такі як блок фільтрів і блоки вакансій, де було зменшено висоту ліній

тексту та падінги. Такі зміни були обґрунтовані аналізом поведінкових даних, що вказували на те, що користувачі надають перевагу більш компактним і менш громіздким елементам інтерфейсу, що сприяє покращенню взаємодії та підвищенню загальної зручності використання сайту.

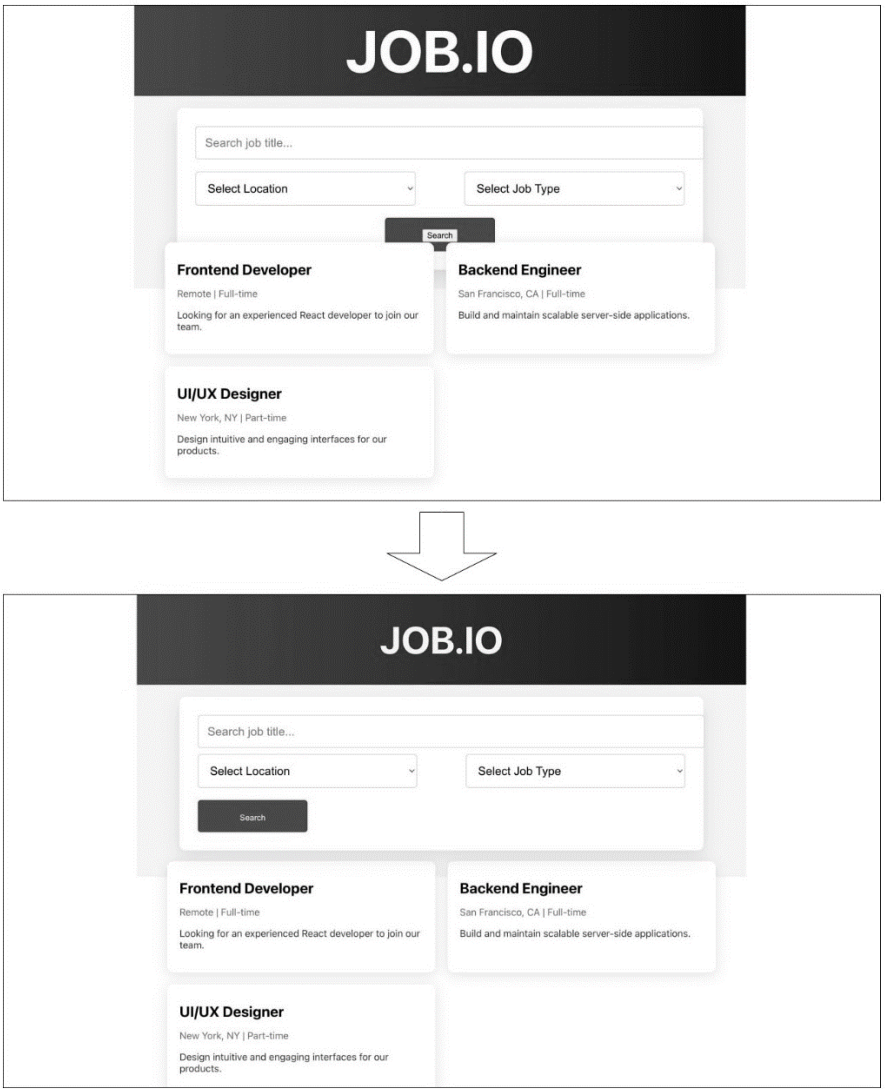

Рис. 7. Зміни внесені фреймворком до графічного інтерфейсу сайту пошуку роботи

На рис. 8 представлено приклад змін, які фреймворк вніс до дизайну сайту прогнозу погоди після аналізу користувацької взаємодії. Однією з ключових змін стало збільшення розміру шрифту для відображення поточної погоди. Оскільки елементи з більшими шрифтами продемонстрували вищий індекс взаємодії, а блок з поточною погодою має високий пріоритет, фреймворк адаптував його стиль відповідно до стилів найбільш затребуваних елементів сторінки. Крім цього, модифікацій зазнали блоки з прогнозом погоди на наступні три дні. Зокрема, прогноз на завтрашній день також був адаптований за аналогічним принципом, оскільки він є другим за пріоритетом елементом після поточної погоди. Це рішення було обґрунтоване високим рівнем інтересу користувачів до цього блоку, що зробило його важливим для оптимізації.

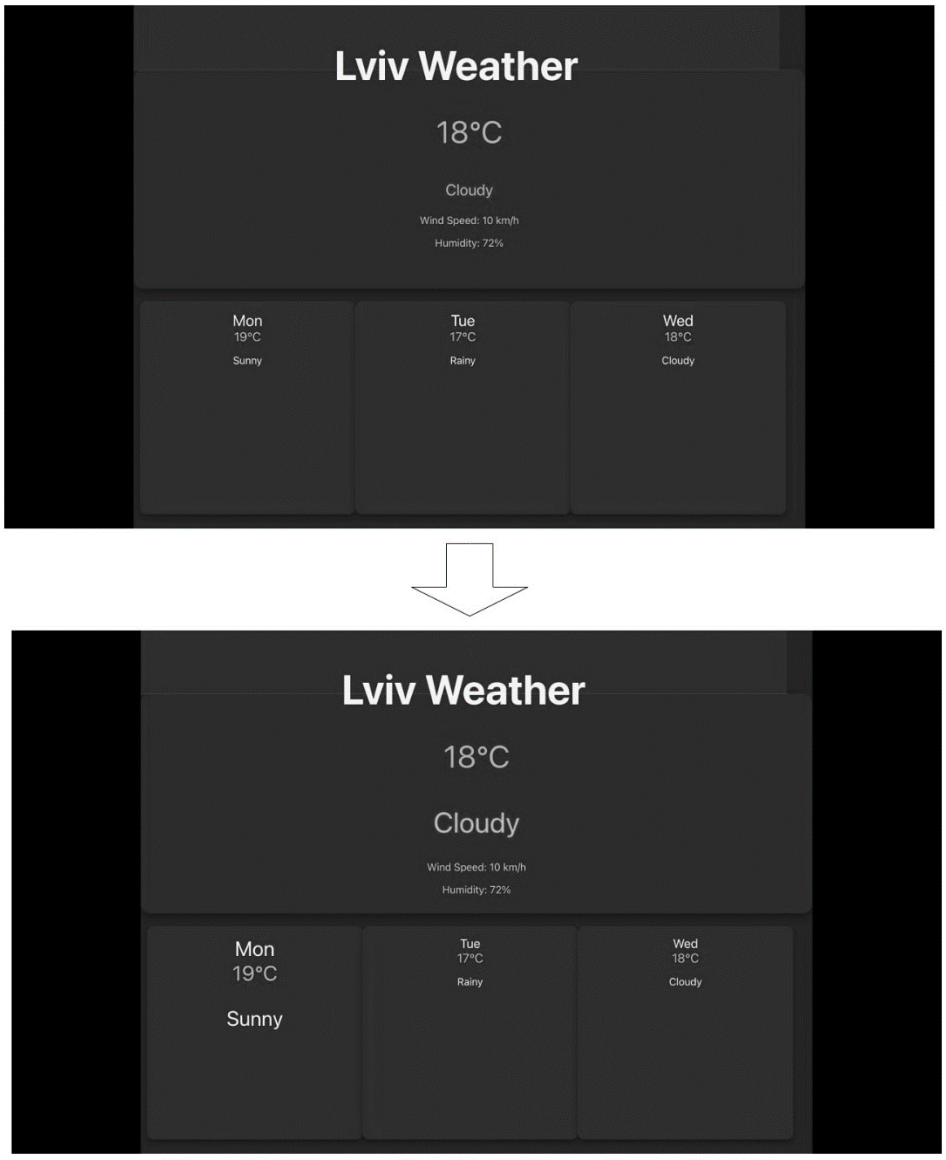

Рис. 8. Зміни внесені фреймворком до сайту погоди

На рис. 9 показано зміни, які фреймворк вніс до сайту новин. Для цього дослідження використовувався клон сайту новин BBC, написаний на React, на який були інтегровані елементи фреймворку у вигляді обгорток для вже існуючих компонентів. Одна з ключових змін стосувалася заголовка сайту: його розмір було зменшено через низький пріоритет цього елемента в загальній ієрархії. Крім того, були внесені невеликі коригування до блоків новин, де зменшили внутрішні відступи та збільшили розмір шрифту заголовків новин. Ці зміни викликані вищим рівнем взаємодії з елементами, що мають більший шрифт, і підвищеним пріоритетом заголовків новин порівняно з іншими елементами сторінки. Приклад наочно демонструє здатність фреймворку інтегруватися в уже наявні веб-сторінки без необхідності їх переписування «з нуля». При цьому зберігаються функціональність і гнучкість для адаптації під індивідуальні потреби користувачів.

Рис. 9. Зміни внесені фреймворком до сайту новин

На основі отриманих варіацій веб-сторінки ми можемо обчислити необхідні зміни, які розробнику доведеться внести для створення лише однієї варіації певного графічного інтерфейсу. Однак для досягнення суттєвих переваг у дизайні та функціональності графічного інтерфейсу необхідно[13] розробити принаймні три варіації інтерфейсу, що дозволяє провести більш глибокий аналіз взаємодії користувачів з різними елементами. Це свідчить про те, що в умовах динамічного розвитку веб-технологій і змінних потреб користувачів важливо не лише адаптувати існуючі стилі, а й створювати нові варіації для всебічної оцінки їх впливу на користувацький досвід.

Враховуючи цю інформацію, ми можемо зробити припущення, що кількість коду, згенерованого адаптивним фреймворком, можна домножити на кількість необхідних варіацій, що безпосередньо вплине на обсяг розробки та час, необхідний для впровадження інтерфейсу. Цей підрахунок не лише підкреслює важливість створення декількох варіацій, але й ілюструє, як адаптивні фреймворки можуть спростити цей процес, автоматизуючи генерування стилів та забезпечуючи гнучкість у розробці.

На основі цих даних ми формуємо діаграму обсягів коду, необхідних для реалізації різних підходів проектування графічного інтерфейсу. Діаграма надає візуальне представлення співвідношення між кількістю варіацій і обсягом коду, що підкреслює важливість стратегічного підходу до проектування та оптимізації інтерфейсу. Це також може слугувати основою для подальшого аналізу ефективності розробки та можливостей впровадження адаптивних рішень у різних сценаріях використання. Діаграму наведено на рис. 10.

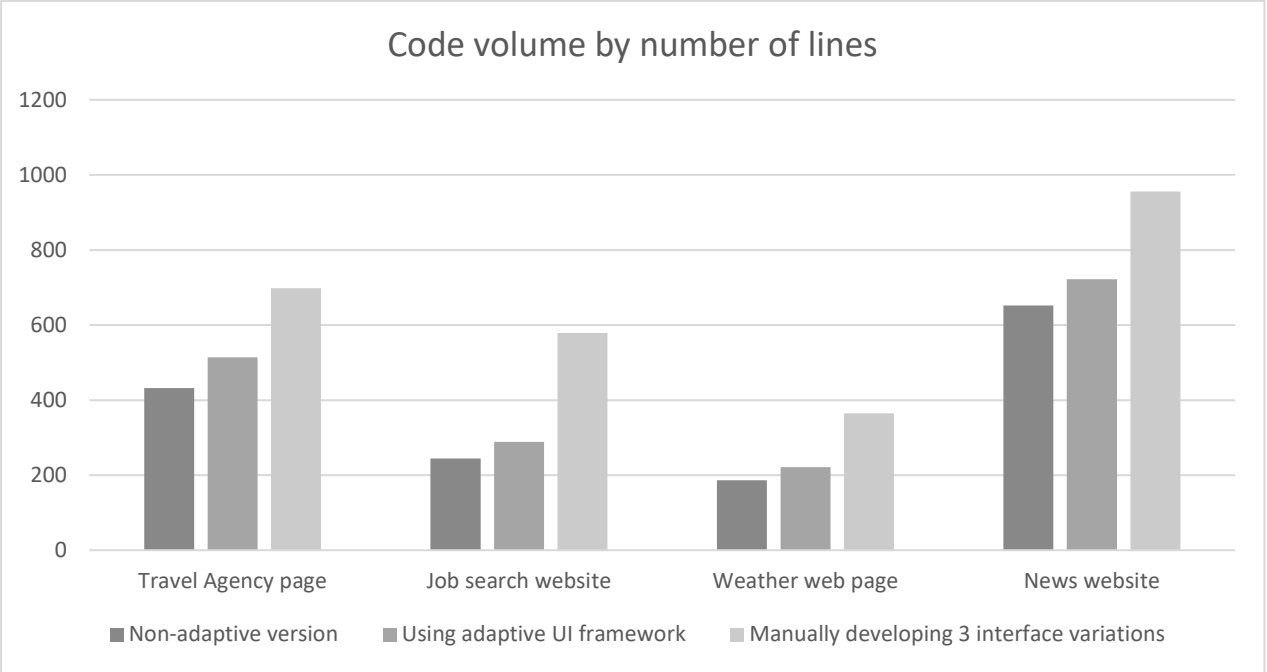

Рис. 10. Порівняння в обсягах вихідного коду різних веб-сторінок.

Тут ми можемо побачити, що інтеграція фреймворку, що реалізує адаптивний графічний інтерфейс, потребує додаткових зусиль від програмістів та загалом збільшує кодову базу проекту. В середньому інтеграція фреймворку веде до збільшення обсягу проекту на 10%. Це може бути обумовлено необхідністю впровадження додаткових модулів, конфігураційних файлів, а також налаштувань, що забезпечують адаптацію інтерфейсу до різних умов використання.

Проте, якщо розробнику довелося б вручну створювати три варіанти того самого інтерфейсу, йому б довелося в середньому написати на 60% більше коду. Цей значний приріст обсягу коду викликаний необхідністю повторно реалізувати однакові або подібні функціональні елементи, а також можливими різницями в реалізації стилів, що можуть виникнути внаслідок відмінностей у дизайні для кожного варіанту.

Крім того, ці показники можуть дещо варіюватися в залежності від кількості варіантів інтерфейсу, їх складності та змін, внесених фреймворком. Важливо зазначити, що вартість інтеграції фреймворку не лише полягає у збільшенні обсягу коду, а також включає витрати на тестування, налагодження та підтримку, які можуть суттєво вплинути на загальну ефективність проекту.

10. Висновки

У роботі досліджено методи аналізу активності користувачів на веб-сторінці та розроблено засоби для динамічної перебудови графічного інтерфейсу відповідно до індивідуальних потреб користувачів. Особливу увагу приділено аналізу взаємодії з інтерфейсом і можливості його адаптації на основі поведінкових даних користувачів. Показано, що динамічна перебудова інтерфейсу може бути ефективно реалізована за допомогою підходів, які використовують підкріплене машинне навчання для формування адаптивних стилів і елементів сторінки на індивідуальному рівні. Доведено доцільність застосування таких методів для персоналізації користувацького досвіду. Продемонстровано здатність фреймворку до зменшення обсягу необхідної роботи розробників з метою реалізації адаптивного користувацького інтерфейсу. Використання автоматизованих алгоритмів для генерування стилів і елементів інтерфейсу суттєво знижує час, витрачений на ручну розробку, а також зменшує ймовірність виникнення помилок, пов'язаних із людським фактором. Подальші етапи дослідження передбачають проведення експериментів із залученням реальних користувачів для отримання емпіричних даних про ефективність запропонованої перебудови графічного інтерфейсу на практиці.

Список літератури

1. Nilakshi Jain. (2023) UI Design : Key to Captivate User Understanding, вересень 2023. 173 ст. ISBN: 8195131948.

2. John Albert Balansag, Rommel A. Canoy, Troyd B. Puquiz, Honey Marjey V. Curay, Denver F. Divino, Luz Clarisse E. Pejera, Mark Van M. Buladaco.(2021) User Engagement and User Design on Online Shopping Apps. International Journal of Advanced Trends in Computer Science and Engineering Volume 10, 3077–3083. https://doi.org./10.30534/ijatcse/2021/011062021

3.Zhiyong Chen. (2023). Research on the application of ergonomics in UI interface design. Applied Mathematics and Nonlinear Sciences. https://doi.org/10.2478/amns.2023.2.00338.

4.Khalid Krayza, Nor Azman Ismail, Layla Hasan, Wad Ghaban, Nadhmi A. Gazem, Maged Nasser. (2023) Adaptable Web Search User InterfaceModel for the Elderly. KSII Transactions on Internet and Information Systems, 2436 – 2457. http://doi.org/10.3837/tiis.2023.09.008.

5.Ahmad Muktamar, Cindy Sandra Lumingkewas, Agus Rofi'i. (2023). The Implementation of User Centered Design Method in Developing UI/UX. JISTE Journal of Information System, Technology and Engineering, 26–31. https://doi.org/10.1017/CBO9781107415324.004.

6.Sadik Khan. (2023) Role of Generative AI for Developing Personalized Content Based Websites. International Journal of Innovative Science and Research Technology, volume 8, Issue 9, 1–5. http://doi.org/10.5281/zenodo.8328205.

7.Daniel Gaspar-Figueiredo, Silvia Abrahao, Marta Fernandez-Diego, Emilio Insfran. (2023). A Comparative Study on Reward Models for UI Adaptation with Reinforcement Learning.

8.Ilan Kirsh. (2021). Visualizing Web Users' Attention to Text With Selection Heatmaps. ICWE 2021, 517–520. https://doi.org/10.1007/978-3-030-74296-6_42.

9.Ilan Kirsh. (2020). Directions and Speeds of Mouse Movements on a Website and Reading Patterns: A Web Usage Mining Case Study. 10th International Conference on Web Intelligence, Mining and Semantics (WIMS'20). https://doi.org/10.1145/3405962.3405982

10. Ilan Kirsh. (2020). Using Mouse Movement Heatmaps to Visualize User Attention to Words. 11th Nordic Conference on Human-Computer Interaction. https://doi.org/10.1145/3419249.3421250.

11. Jörn Hurtienne, Maximilian Landeck. (2014). Beyond Eye Tracking Analogies: Cursor Trajectories as Subtle Cues to Detect Distracting UI Elements. CHI '14 Extended Abstracts on Human Factors in Computing Systems. http://doi.org/10.1145/2559206.2581363.

12. Ana Kešelj Dilberović, Mario Milicevic, Krunoslav Zubrinic, Željka Car. (2022). The Application of Deep Learning for the Evaluation of User Interfaces. http://doi.org/10.3390/s22239336

13. Jakob Nielsen. (1993). Iterative User-Interface Design. https://doi.org/10.1109/2.241424

Front-end фреймворк для побудови застосунків з адаптивним графічним інтерфейсом засобами машинного навчання

FRONT-END FRAMEWORK FOR BUILDING APPLICATIONS WITH ADAPTIVE USER INTERFACES USING MACHINE LEARNING METHODS

I.V. Chaus, T.A. Marusenkova

Lviv Polytechnic National University,
Department of Software Engineering
E-mail: ivan.chaus.mnpzm.2023@lpnu.ua, tetiana.a.marusenkova@lpnu.ua

The article examines approaches to developing a front-end framework for creating web applications with an adaptive graphical interface that dynamically adjusts to the individual needs of users through machine learning algorithms. The relevance of the problem lies in the need to develop interfaces capable of simultaneously meeting the needs of different demographic groups, which requires flexibility in customizing the user experience (UX) and user interface (UI) of modern websites. Traditional interface design methods do not always account for the specific needs of each user, which reduces the effectiveness of interaction with the site. The article proposes an approach that utilizes reinforcement learning algorithms to analyze user interaction with the interface and automatically adapt the interface based on behavioral data. This enhances the accuracy of interface personalization and improves the overall user experience.

The goal of the work is to develop a tool that enables the automated restructuring of the graphical interface of web applications based on individual user needs to improve their user experience. The research develops algorithms to optimize user interaction with web application pages and improve interface efficiency.

The research results demonstrate the framework's ability to dynamically respond to user behavior, assess their level of interaction, and make informed decisions regarding interface parameter adaptation, which in turn helps developers to reduce amount of work needed to implement personalized interface by eliminating the need to manually develop website variants. Using this approach the estimated code base reduction is 40-50%.

Keywords: adaptive interface, front-end, machine learning, user experience, web design.