Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Медіасистем та технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження технології PWA для створення веб-додатку "Планер"
(тема)

Виконав:
студент 2 курсу, групи ТЕМВм-21-1

Калініченко Ю.В.
(прізвище, ініціали)

Спеціальність 186 Видавництво та поліграфія
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма
Технології електронних мультимедійних видань
(повна назва освітньої програми)

Керівник  Колесникова Т.А.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри МСТ _____ Дейнеко Ж.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Технології електронних мультимедійних видань
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
« 31» жовтня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Калініченко Юлії Володимирівні*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження технології PWA для створення веб-додатку "Планер"*

Затверджена наказом по університету від 31 жовтня 2022 р. №1432 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 грудня 2022 р.

3. Вихідні дані до роботи
вид, мета використанн і призначення PWA технології; характеристики веб-сервісів та додатків; варіант поширення; текстові і графічні матеріали.

4. Перелік питань, що потрібно опрацювати в роботі
Мета і завдання роботи; Огляд літератури за темою дослідження; Аналіз стану проблеми; Постановка задачі дослідження Аналіз розробки додатків; Порівняльний аналіз PWA, web-додатку та web-сайту; Технологічні перспективи та обмеження;; Експериментальна частина; Гіпотеза; Економічна частина; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри)
Мета і завдання роботи; Огляд літератури за темою дослідження; Аналіз стану проблеми; Постановка задачі дослідження Аналіз розробки додатків; Порівняльний аналіз PWA, web-додатку та web-сайту; Технологічні перспективи та обмеження;; Експериментальна частина; Гіпотеза; Економічна частина; Висновки.

6. Консультанти розділів роботи (п. 6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п. 1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	Колесникова Т.А.		13.12.2022
Економічна частина	проф. Полозова Т.В.		03.12.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз завдання на кваліфікаційну роботу	31.10.2022	Виконано
2	Огляд літератури за темою дослідження	03.11.2022	Виконано
3	Аналіз стану проблеми і постановка задачі	05.11.2022	Виконано
4	Аналіз розробки додатків	07.11.2022	Виконано
5	Порівняльний аналіз PWA, web-додатку та web-сайту	15.011.2022	Виконано
6	Експериментальна частина	27.11.2022	Виконано
7	Економічне обґрунтування роботи	03.12.2022	Виконано
8	Підготовка пояснювальної записки	10.12.2022	Виконано
9	Підготовка презентації та доповіді	20.12.2022	Виконано

Дата видачі завдання 31 жовтня 2022 р

Студент  Калініченко Ю.В.
(підпис)

Керівник роботи  Колесникова Т.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 73 с., 14 табл., 17 рис., 20 джерел.

PWA ДОДАТОК, НИТИВНИЙ ДОДАТОК, ВЕБ-САЙТ, РОЗМОВНІ СЕРВІСИ, РОЗРОБКА.

Метою магістерської роботи є дослідження створення прогресивних веб-додатків, порівняння PWA з нативними додатками й адаптивними сайтами, а також визначення доцільності використання PWA як уніфікованої технології кросплатформної розробки.

Об’єктом дослідження є технологія прогресивного веб-додатку (PWA), як тип прикладного програмного забезпечення.

Проведення дослідження з’ясовує вплив зручності користування додатком як основного фактору, що формує поведінку користувачів. Формує висновки про зручність веб-додатків, на прикладі створення веб-додатку “Планер”.

ABSTRACT

Explanatory note of qualification work: 73 p., 14 tabl., 17 pic., 20 sources.

PWA APPLICATION, NATIVE APPLICATION, WEBSITE, CONVERSATIONAL SERVICES, DEVELOPMENT.

The purpose of the master's work is to study the creation of progressive web applications, compare PWA with native applications and adaptive sites, as well as determine the feasibility of using PWA as a unified cross-platform development technology.

The object of research is the technology of progressive web application (PWA) as a type of application software.

The study finds out the impact of usability of the application as the main factor shaping user behavior. Forms conclusions about the convenience of web applications, on the example of creating a web application "To do list".

ЗМІСТ

С.

ВСТУП

В наш час Інтернет відіграє величезну роль у житті кожної людини. Він необхідний для безлічі послуг. Велика роль лягає на інформативну роль Інтернету. З його допомогою люди з легкістю дізнаються багато нового і знаходяться в курсі останніх подій.

Люди мають здатність за допомогою інтернету (зокрема веб-сайтів) дізнатися про будь-яку цікаву для них технологію або ж поліпшити свої знання в певній сфері.

Як і більшість речей у житті, веб-індустрія схильна до постійних змін і знаходиться в стані постійного пошуку кращих рішень. Сьогодні використання мобільних пристроїв значно збільшилася, і ця тенденція тільки збирається зберегтися в майбутньому.

Саме з цих причин з’явилася нова технологія Progress Web Apps (PWA), яка надзвичайно перспективна, щоб безповоротно змінити спосіб взаємодії людей із веб-сайтами. Це спеціальні веб-додатки, до яких здійснюється доступ як до звичайних веб-сайтів, але вони пропонують такі переваги, як зручність використання в автономному режимі, push-повідомлення та доступ до обладнання пристрою – речі, які раніше були доступні тільки для нативних додатків.

PWA – це веб-сайт із усіма перевагами програми. PWA надають швидшу, надійнішу та привабливішу версію веб-сайту.

Мобільна електронна комерція вже багато років є стандартом, і з кожним днем вона набуває все більшого поширення. За допомогою PWA, метою якого є трансформація електронної комерції в першу чергу в мобільну, можливо зробите це найзручнішим і найшвидшим способом і з найменшими витратами.

Google є хрещеним батьком PWA, що не дивно. PWA, поєднуючи веб- і мобільний UX, має шанс зруйнувати стіну в саду даних, який охороняють

Google і Apple. Ідея поставити мобільний користувацький досвід в центр уваги була фактично висунута Стівом Джобсом більше десяти років тому.

Генеральний директор Apple представив цю концепцію під час презентації iPhone у 2007 році, оскільки на початку мобільної революції здавалося очевидним, що зовнішні додатки стануть одним із способів використання популярності нового пристрою Apple. Джобс хотів заохотити розробників створювати їх.

Однак ця ідея виявилася недовговічною. У липні 2008 року Apple заморозила концепцію "універсальних додатків". Натомість компанія презентувала App Store, і мобільні додатки почали домінувати в Інтернеті.

У 2015 році Френсіс Берріман та Алекс Рассел, автори терміну PWA, написали у передмові до книги Джейсона Грігсбі "Прогресивні веб-додатки" (Progressive Web Apps):

"Ідея нативних додатків завжди здавалася регресом. Обгороджені стінами сади з жахливим пошуком, сумнівною безпекою і нескінченним податком на оновлення – так відчувалися 1990-ті".

Берріман і Рассел помітили новий тип веб-сайтів, які забезпечують набагато кращий користувацький досвід, ніж традиційні веб-додатки. Вони назвали їх "Прогресивні веб-додатки". Через рік, під час конференції Google IO, Ерік Бідельман, старший інженер з розробки програм для штатних розробників, представив прогресивні веб-додатки як новий стандарт у веб-розробці.

На початку 2018 року компанія Apple надала підтримку базових функцій PWA, але все ж таки наклала певні обмеження щодо обсягу кешу або нативних push-повідомлень.

Основні переваги впровадження PWA включають покращену швидкість та продуктивність, користувацький інтерфейс, схожий на додаток, та багатоплатформне використання. Однак це лише кілька прикладів з широкого спектру переваг PWA.

Замість того, щоб створювати веб-сайт і нативний мобільний додаток окремо, використовуючи технологію PWA, можливо створити лише один

додаток, який буде безперебійно працювати на будь-якому пристрої. З Progressive Web Apps не потрібно створювати окремий нативний додаток.

Прогресивні веб-додатки можуть забезпечити узгоджену роботу з додатком на будь-якому пристрої. Залежно від можливостей браузера, PWA автоматично і поступово розширює вбудовані функції, щоб виглядати і відчувати себе як нативний додаток.

Замість того, щоб розробляти три сутності – додаток для iOS, Android та веб-сайт можливо створити лише одну, яка добре працює на будь-якому пристрої. Крім того, PWA не вимагає присутності в магазинах додатків, а значить, не вимагає жодних платежів.

PWA дозволяє створювати веб-сайти, які просять користувача встановити додаток безпосередньо з мобільного браузера. Це означає, що користувачам не потрібно відвідувати магазин додатків і завантажувати нативний додаток, щоб мати можливість використовувати його. Це збільшує шанси на те, що вони спробують його використовувати.

Таким чином, прогресивні веб-додатки – це все про користувацький досвід, спрямований на підвищення залученості користувачів та всі переваги, які з цим пов'язані. Прогресивні веб-додатки здатні прискорити роботу веб-сайтів. Крім того, вони є стандартом, підтримуваним самою компанією Google, що автоматично робить їх ідеальною парою для голосових рішень.

Користувацький досвід повинен розглядатися в загальній картині, включаючи кожну окрему точку дотику, як в офлайні, так і в онлайні. Якщо подорож клієнта не керована, користувачі можуть відчувати себе приголомшеними і розгубленими, що не є найкращою основою для здійснення покупок. PWA використовує переваги звичок, які користувачі вже набули за допомогою нативних додатків, дозволяючи розробникам використовувати функції мобільних телефонів для збагачення користувацького інтерфейсу.

1 АНАЛІЗ ЛІТЕРАТУРИ ВІДПОВІДНО ЗАВДАНЬ ДОСЛІДЖЕННЯ

Прогресивні веб-додатки існують не так давно, тому ця тема ще мало досліджена. Традиційно мобільні додатки розробляються як нативні додатки. Розвиток веб-технологій відкрив нові можливості розвитку веб-технологій відкрив нові можливості, які можуть навіть замінити традиційний спосіб розробки додатків.

Нативні додатки орієнтовані лише на певну платформу, наприклад, Android, iOS або Windows. При створенні нативного додатку необхідно створювати окремий додаток для кожної платформи яку потрібно підтримувати.

При написанні нативних додатків можна отримати доступ до всіх нативних ресурсів пристрою. Зазвичай нативні додатки також швидші та працюють більш плавно порівняно з гібридними або веб-додатками.

Поширення нативних додатків відбувається через централізовані маркетплейси, специфічні для кожної платформи. Однією з переваг таких централізованих маркетплейсів є те, що додатки, представлені на них, охоплюють ширшу аудиторію.

Гібридні додатки намагаються поєднати найкраще з нативних та веб-додатків. Зазвичай цей процес включає в себе написання програми як веб-додатку, який потім обгортається в нативну оболонку для кожної з цільових платформ за допомогою елемента керування веб-переглядом. Ідея створення гібридних додатків полягає в тому, що всі нативні можливості є в основному доступні, чого часто не можна сказати про веб-додатків.

Веб-додатки – це особливий вид інтерактивних веб-сайтів, які функціонують як традиційні настільні або мобільні додатки.

Не існує точного визначення того, що робить веб-сайт веб-додатком. Зазвичай веб-сайт, який функціонує подібно до настільного або мобільного додатку, називають веб-додатком.

Однією з відмінностей є те, що веб-додаток має єдину мету використання, в той час як веб-сайти зазвичай містять багато розділів і служать декільком цілям. Основною перевагою в порівнянні з нативними додатками є те, що веб-додатки не вимагають ніякого встановлення і доступ до них можна отримати миттєво з будь-якого веб-браузера. Це, з іншого боку, означає, що користувач повинен мати працююче мережеве з'єднання. Ця вимога не поширюється на прогресивні веб-додатки [1].

Основним недоліком веб-додатків раніше був той факт, що багато нативних функцій, таких як фотографування за допомогою камери пристрою або отримання даних GPS, були недоступні.

Сьогодні всі основні браузери підтримують більшість з цих можливостей пристрою, що є однією з причин зростаючої популярності веб-додатків.

1.1 Огляд літератури

У 2015 році дизайнер Френсіс Берріман (Frances Berriman) та інженер Google Chrome Алекс Рассел (Alex Russell) ввели термін "прогресивні веб-додатки" для опису додатків, що використовують переваги нових функцій, які підтримуються сучасними браузерами, включаючи сервісні працівники та маніфести веб-додатків, які дозволяють користувачам оновлювати веб-додатки в рідній операційній системі маніфести додатків, які дозволяють користувачам оновлювати веб-додатки до прогресивних веб-додатків у своїй рідній операційній системі (ОС) [2].

У блозі "Прогресивні веб-додатки: Escaping Tabs Without Losing Our Soul" Рассел пояснює, як вони прийшли до цієї фрази коли вони обговорювали найкращі практики створення веб-додатків. Необхідні технології вже існували деякий час необхідні технології існували вже деякий час, але ніхто не давав їм назву [3].

У своєму блозі Рассел визначає PWA як веб-додаток, який має наступні характеристики:

– адаптивність: Додаток буде масштабуватися і працювати з будь-яким розміром або формою екрану, наприклад, мобільного пристрою, планшета, телевізора тощо;

– незалежність від підключення: додаток буде працювати навіть за відсутності;

– за допомогою сервісного працівника. Це, звичайно, обмежує доступ до додатку лише вміст, який був завантажений раніше;

– свіжість: Додаток завжди буде актуальним завдяки тому, що це по суті веб-сторінка, яка може бути отримана знову, коли користувач перезавантажить її;

– безпечність: додаток буде обслуговуватися через захищене з'єднання HTTPS;

– відкритість: додаток може бути знайдений пошуковими системами і буде;

– класифікується як "додаток", а не як веб-сторінка;

– можливість повторного залучення: Додаток зможе надсилати користувачеві push-повідомлення;

– встановлюваність: Додаток може бути доданий на домашній екран користувача;

– можливість поділитися: Додаток має свій унікальний URL-адресу, щоб ним міг поділитися і отримати доступ будь-хто.

Це перше визначення, яке було дано прогресивним веб-додаткам [3-4].

1.2 Огляд досліджень

Ще у 2007 році Стів Джобс представив веб-додатки як основний спосіб розробки додатків для iPhone. Реліз App Store для нативних додатків для iOS відбувся роком пізніше. На той час веб-технології були недостатньо потужними, і від ідеї підтримки виключно веб-додатків, оскільки нативні додатки працювали набагато краще.

У 2015 році Міжнародний союз електрозв'язку підрахував, що близько 3,2 мільярда людей, або майже половина населення світу, будуть перебувати в Інтернеті до кінця цього року. Для веб-розробників спроба задовольнити всі ці різні сценарії здається щонайменше складним завданням. Ось тут і з'являються прогресивні веб-програми (PWAS). Вони надають розробникам, можливість створювати швидші, стійкіші та цікавіші веб-сайти, до яких можуть мати доступ мільярди людей по всьому світу [4].

У 2014 році кількість користувачів у світі, які виходили в Інтернет з мобільних пристроїв, перевищила кількість тих, хто робив це з настільних комп'ютерів.

Згідно з дослідженням, охоплення мобільного інтернету значно перевищує охоплення нативних додатків. Воно склало 11,4 мільйона унікальних відвідувачів на місяць у порівнянні з 4 мільйонами відвідувачів. При цьому статистика взаємодії користувачів з сервісами показала, що користувачі схильні проводити більше часу в нативних мобільних додатках порівняно зі стандартним веб-додатком. В середньому це було 188,6 хвилин в додатку проти 9,3 хвилин в інтернеті [5].

Отже, ідея була зрозумілою. Вони хотіли надати користувачам мобільного інтернету такий самий цікавий досвід, як і в мобільному додатку. Таким чином, були розроблені прогресивні веб-додатки.

Результати дослідження, проведеного Fransson у 2017 році, показують, що фотографування за допомогою нативного додатку є значно швидшим порівняно з аналогічним рішенням з використанням PWA.

З іншого боку, додатки, які потребують геолокаційних сервісів, насправді працюють дещо швидше з PWA порівняно з нативним додатком.

Основною перевагою PWA є те, що додаток потрібно закодувати один раз, і він буде працювати практично з будь-яким сучасним браузером на будь-якому пристрої. З нативними додатками доводиться створювати окремі додатки – по одному для кожної платформи, яку вони хочуть підтримувати.

Розробка також відбувається швидше і коштуватиме менше грошей у порівнянні зі створенням нативного додатку. Існує набагато більше людей, які здатні займатися веб-розробкою, ніж тих, хто вміє кодувати нативні додатки для Android або iOS [6].

PWA працюють скрізь і не залежать від якогось конкретного ринку, а також не вимагають від користувача завантаження перед використанням.

Вони також завжди актуальні, якщо користувач не перебуває в режимі офлайн, оскільки додаток – це просто розширена веб-сторінка.

Це також робить процес розробки швидшим і трохи дешевшим, оскільки розробнику не потрібно відправляти додаток в будь-який магазин додатків. Розгортання додатку вимагає лише налаштування веб-сервера.

PWA займають значно менше місця на пристрої користувача у порівнянні з аналогічними нативними додатками. Одним із прикладів цього є Twitter. Їхній прогресивний веб-додаток Twitter Lite, займає менше 3% місця в пам'яті пристрою в порівнянні з нативним додатком. Ще одна перевага полягає в тому, що PWA не завантажують все, що пов'язано з додатком, коли користувач вперше заходить на сторінку. Замість цього вони поступово завантажують всі необхідні ресурси на основі того, що користувач робить у додатку. PWA також зазвичай запускаються швидше порівняно з нативними додатками [7].

До того ж, PWA легше знайти порівняно з нативними додатками. Прогресивним веб-додатком можна ділитися так само легко, як і веб-сторінкою. Перейшовши за посиланням, новий користувач отримує миттєвий доступ до додатку, оскільки його не потрібно встановлювати.

Оскільки всі PWA є звичайними веб-сайтами, вони також можуть бути проіндексовані пошуковими системами, що робить можливим для будь-кого легко знайти їх, провівши пошук в Інтернеті.

З іншого боку наскільки добре працюють PWA, залежить від браузера, який використовує користувач. Наприклад, Safari для iOS не підтримує Push API. Це означає, що ці користувачі не можуть отримувати push-сповіщення

через PWA. Це одна з причин, чому хтось захоче зробити нативний додаток замість PWA [8].

Також не так просто монетизувати прогресивні веб-додатки, як нативні додатки. Тому що PWA – це, по суті, веб-сторінки, до яких може отримати доступ кожен, хто має URL-адресу. З нативними додатками можна встановити ціну, яку користувачі повинні заплатити, перш ніж вони зможуть встановити додаток.

Нативні додатки також вважаються більш безпечними з точки зору захисту даних. А недавнє дослідження показує деякі з причин, що стоять за цими твердженнями. Вони з'ясували, що пуш-повідомлення можна легко використовувати для обману людей, видаючи себе за когось іншого відправник пуш-повідомлення може налаштувати зовнішній вигляд повідомлення. Також помітили, що працівники служб можуть бути використані для зловживань через те, що вони можуть продовжувати працювати у фоновому режимі [9].

У таблиці 1.1 наведено порівняння між нативним додатком, PWA та стандартним веб-сайтом за різними важливими параметрами.

Таблиця 1.1 – Порівняння нативного додатку, PWA та веб-сайту

Функції	Нативний додаток	PWA	Веб-сайт
Установка	Необхідно зайти в App store або Play Store, натиснути завантажити	Просто натиснути кнопку, щоб додати їх на головний екран свого телефону (тільки на Android)	Установка не потрібна
Оновлення	Потрібно зайти в App store або Play Store, після чого завантажити користувачем	Оновлення відбуваються миттєво	Оновлення відбуваються миттєво
Розмір	Займають час для завантаження на пристрої користувача	Невеликі та швидкі	Невеликі та швидкі

Продовження таблиці 1.1

Функції	Нативний додаток	PWA	Веб-сайт
Офлайн доступ	Доступний	Потрібно один раз скористатися додатком онлайн, після чого повинна бути можливість доступу до кешованого контент в автономному режимі	Не потрібно
Користувацький досвід (UX)	Добре, коли додаток добре розроблений	Заплутаний че рез подвійне меню (меню програми і меню браузера)	Те ж саме, що і в прогресивному веб-додатку
Push-повідомлення	Так	Так (тільки для Android)	Так (можливо тільки з сторонніх сервісів)

Наразі деякі функції все ще недоступні для веб-додатків. У таблиці 1.2 наведена коротка інформація про те, які функції вже підтримуються в Safari для iOS та Chrome для Android.

З результатів видно, що в iOS Safari відсутня підтримка Web Bluetooth, Push API та Background Sync API. Ці функції вже працюють в Chrome для Android.

Таблиця 1.2 – Функції веб-додатків

Функції	iOS Safari	Chrome для Android
Сервісний працівник (Service Worker)	Так	Так
Push API	Ні	Так
API фонової синхронізації	Ні	Так
Індексованість БД	Так	Так
File API	Так	Так
MediaStream API	Так	Так
Web Bluetooth	Ні	Так
API геолокації	Ні	Так
Web Share API	Так	Так

На рисунку 1.1 показано технології та цільові показники PWA.

Рисунок 1.1 – Схема «Технології та цільові показники PWA»

Швидкість (Fast) – швидко відбувається взаємообмін даними, UI-плавний і чуйний.

Надійність (Reliable) – незалежно від статусу і якості мережевого з'єднання додаток завантажується швидко.

Привабливість (Engaging) – працювати з додатком комфортно.

Архітектура оболонки додатків обслуговується сервісним працівником, а потім доставляється вміст. Вони часто кешується сервісним працівником з його джерела за допомогою запитів API [10].

Сайти, які люди відвідують частіше, зможуть утримувати останній контент, який людина відвідала, чекаючи, поки мережа динамічно завантажить останнє оновлення. У моделі архітектурної оболонки основна увага приділяється утриманню оболонки інтерфейсу програми та вмісту всередині неї окремо, і вони кешуються окремо. В ідеалі, контент кешується таким чином, щоб він завантажувалася якомога швидше, коли користувач відвідує і повертається пізніше. Теоретично, роздільне завантаження оболонки та вмісту покращує сприйняття користувачем продуктивності і зручність використання програми.

1.3 Технології та цільові показники

Масштабних досліджень на тему прогресивних веб-додатків ще не проводилося. З цієї причини більша частина інформації, базується на онлайн-

джерелах. З певних причин визначення прогресивного веб-додатку не є зрозумілим для багатьох. Визначення варіюються від суворих технічних визначень до тих, що описують нетехнічні аспекти, такі як користувацький досвід.

Прогресивні веб-додатки, досить добре конкурують з нативними додатками. Єдиним основним браузером, який ще не зовсім підтримує всі основні функції PWA, є Safari на iOS. Push-сповіщення поки що не підтримуються Safari, що може призвести до необхідності створення нативного додатку для iOS, хоча інші платформи можуть працювати з PWA як повна заміна нативних додатків.

На даний момент відсутні деякі з найбільш важливих браузеру Chrome є можливість доступу до файлової системи та додавання значка на іконку бейджа на іконку додатку. В iOS Safari відсутня можливість перегляду контактів користувача. Safari також не підтримує Push API, але для цього є цікаве обхідне рішення – використання Push-повідомлень Facebook Messenger. Для цього потрібно, щоб на пристрої користувача був встановлений додаток Facebook Messenger [11].

1.4 Постановка задачі дослідження

Стрімка діджиталізація економічних процесів значно впливає на функціонування підприємств, держави та суспільства. У таких умовах з метою отримання конкурентних переваг і домінуючих позицій на ринку компанії зацікавлені в пошуку нових каналів взаємодії зі своїми клієнтами у веб-просторі.

У зв’язку з цим особливої актуальності набувають питання створення сайтів і мобільних додатків, дослідження сучасних технологій веб-розробки.

Веб-сайти не можуть запропонувати користувачам ту взаємодію (User Expiriense, UX), яка притаманна нативним додаткам (наприклад, робота без підключення до інтернету, запуск у фоновому режимі, push-сповіщення тощо)

Незважаючи на значну зацікавленість науковців і дослідників сутністю, особливостями та перевагами розробки PWA, відсутність комплексного підходу до визначених питань стримує широке розповсюдження технології.

Метою є дослідження створення прогресивних веб-додатків, порівняння PWA з нативними додатками й адаптивними сайтами, а також визначення доцільності використання PWA як уніфікованої технології кросплатформної розробки.

Актуальність даної роботи полягає в тому, що у наш час мережа буквально всюди: на годиннику, смартфонах, планшетах, комп'ютерах, в автомобілях. Завдяки сучасним браузерам, програмне забезпечення для відображення веб-контенту на всіх цих пристроях може бути абсолютно однаковим.

Хоча браузери залишаються головним способом взаємодії для користувачів, кожен пристрій тепер має свій власний магазин додатків. На Android, Chrome OS, Mac або Windows в магазині можна встановити програмне забезпечення, призначене для кожного пристрою.

Суть встановленого програмного забезпечення полягає в тому, що програмне забезпечення буде інтегруватися з пристроєм так, як цього очікує користувач.

Об’єктом дослідження є технологія прогресивного веб-додатку (PWA), як тип прикладного програмного забезпечення.

Таким чином, поставлена в роботі мета, обумовила необхідність вирішення наступних задач:

– підбір та аналіз літератури за темою дослідження;

– обґрунтування методу експерименту, що буде використаний в дослідженні, а саме метод експертних оцінок, адже зручність використання є суб’єктивною думкою;

– розробка анкети опитування для проведення експерименту методом експертних оцінок;

– проведення аналізу результатів, експерименту за отриманими даними;

Необхідно виявити які переваги від використання PWA можуть отримати компанії у сфері e-commerce й інформаційного бізнесу. Яка зручність використання таких додатків, наскільки високим є рівень UI/UX, їх швидкодія та невибагливість до ресурсів, можливість роботи без підключення до інтернету та чи сприяє це розширенню мобільної присутності компаній у веб-просторі.

Чи є невисока вартість розробки та супроводу PWA у порівнянні з їх нативними аналогами, перевагою.

Отже, необхідно провести дослідження впливу зручності користування додатком як основного фактору, що формує поведінку користувачів. Необхідно перевірити який додаток вибрати для бізнесу – Прогресивні веб-додатки або нативні. Відповідь на це питання залежить від функцій та можливостей, які бізнес вимагає від додатків. Тобто при проведенні дослідження, можна сформувати деякі висновки про зручність веб-додатків, на прикладі створення веб-додатку “Планер”.

2 АНАЛІЗ РОЗРОБКИ ДОДАТКІВ

2.1 Базовий рівень: Нативна розробка та веб-додатки

Природним вибором для розробки додатків є нативний комплект розробки програмного забезпечення (SDK). Зазвичай він надається вендором мобільної платформи. SDK пропонує досвід розробки, адаптований до платформи: Зазвичай є кілька мов програмування. У випадку з iOS, можуть бути бути використані Objective-C та Swift. На Android нещодавнє додавання мови Kotlin, що входить до набору Javasuperset, було позитивно сприйнято спільнотою розробників, яка в іншому випадку покладається на Java. При розробці для платформи Windows, так званого додатку для універсальної платформи Windows можна використовувати декілька мов, включаючи C++, C#, Visual Basic та JavaScript. Доступ до функцій пристрою можна отримати через власні інтерфейси прикладного програмування (API) платформи [12].

Графічний інтерфейс користувача (GUI) складається з нативних елементів інтерфейсу. Нативний зовнішній вигляд та короткий час реакції є прямим наслідком. Додатки з'являються і з ними можна взаємодіяти так само, як і з додатками для конкретної платформи відповідно до самої мобільної операційної системи; продуктивність є високою, принаймні, для ретельно розроблених додатків.

Хоча функції, впроваджені в мобільні платформи, незабаром знаходять свій шлях до всіх сучасних платформ, якщо вони вважаються корисними, платформи залишаються несумісними. Несумісність має глибоке коріння. Мало того, що графічний інтерфейс виглядає по-різному (коли він відповідає керівним принципам проектування), але і весь досвід розробки розходиться. Як правило, API не розроблені однаково, мови програмування нав'язують певний стиль, а екосистеми платформи, включаючи інструменти та парадигми відрізняються.

Оскільки все це робить дуже дорогим розробку одного і того ж програмного забезпечення для нативних додатків з декількох платформ, популярними стали крос-платформні фреймворки для розробки. Ті ж труднощі, з якими стикається людина, ускладнюють і створення такого фреймворку, про що буде розказано в наступному розділі. Сам по собі термін "веб-додаток" є досить незрозумілим. Нам не відомо його узгодженого визначення. Загалом, багато додатків, які надаються через Інтернет і працюють за моделлю клієнт-сервер, вважаються веб-додатками; такі додатки існували ще до появи смартфонів. У літературі, присвяченій мобільним обчисленням, Webapp зазвичай позначає мобільний Web-додаток [13].

Перші веб-додатки просто використовували вбудовані браузер платформ. Їм не вистачало функціональних можливостей, таких як розміщення у списку додатків для смартфонів та розповсюдження за допомогою магазинів додатків. Крім того, вони не мали або мали обмежений доступ до функціональних можливостей конкретних пристроїв і покладалися на загальний веб-вигляд. Хоча веб-додатки все ще розробляються з використанням тієї ж базової веб-технології, фактичне програмування збагатилося величезною кількістю різноманітних фреймворків. Крім того, були покращені можливості і продуктивність. Завдяки своїй залежності від браузера, веб-додатки сумісні на всіх платформах, на яких працює достатньо потужний браузер. Однак, оскільки браузери відрізняються за реалізацією та відповідністю специфікаціям CCS HTML та JavaScript, веб-додатки не функціонують так, як очікується, у всіх браузерах. Крім того, у більшості сучасних браузерів існує можливість відключення JavaScript; таким чином, ефективне відключення логічного рівня і часто рівня користувальницького інтерфейсу сучасних веб-додатків призводить до того, що додатки відображають порожні сторінки. До того, як розробка крос-платформних додатків стала популярною, розробники стикалися з вибором між нативними додатками та веб-додатками. Для "екстремальних" випадків рекомендації щодо вибору залишилися незмінними, але для "середньостатистичних"

додатків вони змінилися. Високопродуктивні додатки, близькі до апаратного забезпечення, вимагають нативної розробки. Досить прості, засновані на формах додатки швидко розробляються як веб-додатки. Завдяки HTML5 та сучасному JavaScript було досягнуто кращої продуктивності та став доступним ширший спектр функцій пристроїв. Фреймворки – зокрема, для графічного інтерфейсу і для JavaScript – пропонують багато додаткових можливостей, наприклад, веб-додатки, які імітують нативний вигляд. Як наслідок, веб-додатки користуються популярністю. Нативна розробка, з одного боку, і веб-додатки, з іншого боку, залишаються орієнтиром для будь-яких конкурентів. Це стосується не тільки кросплатформних фреймворків розробки, але й будь-якої іншої технології розробки, навіть якщо вона не є специфічною для багатоплатформної розробки. Тому можливості, розгалуженість та якість нативних додатків та універсальних веб-додатків повинні бути зважені при обговоренні уніфікованої розробки [14].

2.2 Крос-платформна розробка додатків

Кросплатформені підходи до розробки додатків не є чимось новим, проте технологічні можливості з часом змінилися.

Існує одна базова характеристика, яка є спільною для всіх фреймворків для розробки крос-платформних додатків: код розробляється один раз, в той час як додатки можуть бути надані для декількох платформ. Крім того, фреймворки суттєво відрізняються за базовим підходом та парадигмою, а також за сферою застосування. Основні парадигмальні можливості були узагальнені на рисунку 3.2.

Очевидно, що початковий вибір полягає в тому, чи використовувати крос-платформну використовувати кросплатформенну розробку, чи ні. З точки зору високого рівня, це залишає тільки нативну розробку при прийнятті рішення про відмову від кросплатформної розробки додатків. При більш детальному розгляді, варіанти трохи більш розмиті. PWA, можливо,

досі не розглядалися як крос-платформний підхід до розробки, хоча за своєю концепцією вони прагнуть підтримувати кілька платформ з однієї кодової бази (по суті, будучи веб-орієнтованими).

Крос-платформну розробку можна розділити на дві основні парадигми: середовища виконання та генеративні підходи. В останньому випадку кінцевою метою є забезпечення нативного додатку для кожної підтримуваної платформи. Те, як цей нативний додаток був згенерований з єдиної кодової бази, можна ще більше диференціювати. Підходи, керовані моделями (позначені на рисунку як MDSD: розробка програмного забезпечення, керована моделями), покладаються на незалежну від платформи модель, на основі якої генеруються додатки. Як правило, вони використовують – зазвичай текстову або, як альтернатива, графічну – специфічну для домену мову (DSL) для опису програми. Транслятори беруть (зазвичай нативний) код, написаний для однієї платформи, і перетворюють його на нативний код для іншої платформи.

Підходи, які не генерують додатки, покладаються на середовище виконання. Замість запуску коду безпосередньо на платформі, що вимагає, щоб код був нативним для цієї платформи, середовище виконання є мостом між додатком і платформою. Існує ряд способів технологічної реалізації цієї парадигми. Веб-додатки, які вже назвали базовим підходом, просто запускаються у веб-браузері, що надається платформою. Гібридні додатки, як правило, покладаються на веб-технології, але надають нативну упаковку. Традиційно це було корисно, наприклад, з точки зору доступу до специфічних функцій пристрою, але обмежувало ці додатки зовнішнім виглядом, який щонайбільше наближався до нативних додатків [15].

Нарешті, додатки, засновані на автономному середовищі виконання, схожі на гібриди, але використовують нативні елементи графічного інтерфейсу.

На рисунку 2.1 представлені парадигмальні можливості розробки додатків.

Рисунок 2.1 – Парадигмальні можливості

2.3 Прогресивні веб-додатки

PWA є новими за своєю концепцією, хоча вони здебільшого покладаються на існуючі технологічні артефакти та концепції.

Існують певні характеристики, які визначають PWA, і які відрізняють їх від звичайних веб-сайтів та нативних або крос-платформних мобільних додатків. Основною відмінністю між звичайним веб-сайтом та PWA є додаткова функціональність та користувацький досвід (UX), який надає останній. Якщо звичайний веб-сайт вимагає від користувача відкриття браузера, введення URL-адреси та очікування завантаження всього контенту при кожному відвідуванні, що фактично унеможливлює роботу в режимі офлайн, то PWA вимагає виконання цих кроків лише при першому відвідуванні. Після встановлення домашнього екрану всі необхідні статичні файли, включаючи HTML, CSS, JavaScript, зображення та шрифти для веб-сайту, тепер зберігаються на телефоні користувача, готові до використання в автономному режимі. Всі динамічні дані можуть бути закешовані для використання в автономному режимі (або при низькій якості зв'язку) і повторно завантажені за необхідності, наприклад, коли з'являються нові дані і телефон має якісне мережеве з'єднання.

Якщо звичайний веб-сайт буде загорнутий у браузер (наприклад, Chrome Android) з видимими артефактами браузера (такими як адресний

рядок та меню), PWA буде аналогічно працювати в екземплярі браузера, але без цих артефактів.

Таким чином, PWA буде виглядати як звичайний додаток. Якщо PWA стилізовано правильно, відповідно до рекомендацій щодо дизайну кожної мобільної платформи, відрізнити звичайний нативний або крос-платформний додаток від PWA за зовнішнім виглядом буде досить складно.

Якщо порівнювати PWA зі звичайним нативним або кросплатформним мобільним додатком, то однією з основних характеристик та переваг PWA є мінімальний слід, який він залишає на телефоні користувача. У будь-якому випадку, їх PWA буде на два порядки меншим, ніж у порівнянних нативних додатків – без шкоди для функціональності [16].

З технічної точки зору, прогресивний веб-додаток – це, простіше кажучи, звичайний веб-сайт з документом мета-даних (маніфестом) на основі JSON, оболонкою додатка і фоновим сценарієм (Service Worker), написаним на JavaScript. Оболонка додатку – це мінімальний статичний графічний інтерфейс і логіка, необхідні для відображення додатку без динамічного контенту, що залежить від з'єднання, тобто в автономному режимі. Цей графічний інтерфейс і логіка можуть включати частини програми, такі як логіка маршрутизації та навігації, а також елементи інтерфейсу користувача. Завдяки тому, що оболонка програми доступна в автономному режимі і завантажується з кешу програми, вона рендериться негайно (вимірюється як час до першого зафарбовування), роблячи додаток доступним швидше, ніж звичайний веб-сайт, що працює тільки в режимі онлайн без PWA. Далі сценарій Service Worker обробляє і проксі-серверує всі мережеві з'єднання, логіку кешування і фонові завдання, забезпечуючи таким чином автономну роботу, яку може надати PWA. Service Worker може вирішити, чи потрібно отримувати новий вміст з веб-служби, або якщо будь-який кешований вміст все ще актуальний для користувача, таким чином уникаючи непотрібних мережевих дзвінків, або навіть для відображення кешованого вмісту в автономному режимі. Крім того, веб-сайт повинен обслуговуватися за

протоколом HTTPS та реалізовувати адаптивний дизайн, тобто задовільно працювати на пристроях з різним розміром екрану та роздільною здатністю.

З точки зору розробника, при розробці прогресивного веб-додатку слід враховувати безліч факторів. Як приклад, компанія Google пропагує використання (експериментального) патерну PRPL (абревіатура від Push – Render – Pre-cache – Lazy-load, вимовляється як Purple). Шаблон ґрунтується на ідеї, що Інтернет працює повільно на мобільних пристроях, і що час, необхідний для того, щоб додаток став інтерактивним (вимірюється як час до інтерактивності), є надзвичайно важливим для оптимізації. Це досягається за допомогою таких методів, як інтенсивне кешування динамічного контенту, новий стандарт HTTP/2 для більш розумного запиту і отримання статичних файлів з веб-сервера, Application Shell, а також розумне завантаження контенту. Крім того, компанія Google випустила орієнтований на розробників інструмент тестування PWA під назвою Lighthouse. Цей інструмент допомагає тестувати веб-сайти на відповідність вимогам PWA, полегшуючи тим самим розробникам процес розробки PWA-сайтів.

2.4 Подальші підходи

Межі між крос-платформними додатками, PWA і навіть веб-додатками є розмитими. Це можна пояснити відсутністю визначень та диз'юнктивним розподілом деяких технологічних аспектів. Крім того, деякі підходи не ставлять за основну мету багатоплатформенність, але все ж таки сприяють їй. Концепція автономної роботи веб-сайтів на мобільних телефонах не є новою. Певний час кеш додатків (Application Cache, AppCache) був рішенням для досягнення саме цієї мети.

Однак, з впровадженням Service Workers, AppCache знаходиться в процесі припинення використання на користь Service Workers і, т.ч., PWA.

Інший альтернативний підхід відстоює компанія Microsoft під назвою Hosted Web Apps (HWA). Ця концепція займає проміжне місце між веб-

додатками та нативними додатками, подібно до Progressive Web Apps. Основною перевагою використання HWA над PWA є тісна інтеграція з платформою Windows. Це дозволяє використовувати такі функції, як виклик власних API платформи Windows з кодової бази JavaScript HWA. HWA також буде працювати на різних платформах Windows, включаючи такі пристрої, як ПК, пристрої Windows Phone та Xbox. Це дійсно цікавий тип гібридного рішення, оскільки він поєднує API платформи виклики платформи у веб-налаштування. Цей підхід можна порівняти з Cordova для гібридних мобільних пристроїв. порівняти з Cordova для гібридної розробки мобільних додатків, хоча і обмежений платформою Windows [16], [17].

Вимоги до уніфікованої розробки можна розглядати з двох точок зору. По-перше, можна ретельно розглянути вимоги до проектування додатків у багатоплатформенний спосіб, наприклад, щодо графічного інтерфейсу та продуктивності – концептуальний погляд. По-друге, можна розглянути технічні вимоги, наприклад, щодо передумов і сполучних елементів – технологічний погляд [17].

Наступні вимоги є важливими для безперешкодного створення додатків для декількох платформ одночасно:

– користувальницький інтерфейс (UI), який може бути побудований з типових елементів, і який забезпечує різні макети;

– можливість забезпечити потік управління і різні форми взаємодії в додатку;

– зовнішній вигляд, який узгоджується з рекомендаціями щодо дизайну та юзабіліті для відповідних платформ;

– можливість визначення типів даних та забезпечення базових операцій створення, отримання, оновлення та видалення (CRUD), як на пристрої, так і в клієнт-серверній моделі;

– відображення полів форм на модель даних, включаючи перевірку введених даних та можливість збереження даних;

– засоби реагування на вхідні дані, події та зміни стану, а також можливість доступу принаймні до найпоширеніших апаратних засобів.

Крім того, фреймворк повинен мати швидку криву навчання, підтримувати основні платформи та надавати своєчасні результати.

Як видно з дослідження PWA, суворою вимогою є достатня довгострокова реалістичність.

Ця оцінка приблизно відповідає передумовам для бізнес-додатків, як це було запропоновано. Базуючись на дослідженні можна виділити декілька технологічних передумов для створення додатків, які охоплюють декілька платформ:

– адекватна підтримка розробки за допомогою середовища розробки та інструментів, а також хороша тестованість додатків;

– достатній ступінь масштабованості;

– продуктивність, близька до нативної;

– хороший рівень супроводжуваності та розширюваності додатків;

– інтерфейси для бізнес-функціональності, такі як підтримка бекенд-систем, і принаймні базовий рівень безпеки, інтегрований у фреймворк.

3 ПОРІВНЯЛЬНИЙ АНАЛІЗ PWA, WEB-ДОДАТКУ ТА WEB-САЙТУ

За для проведення аналізу даних сфери було проведено порівняння PWA, нативного додатку WEB-сайту ‘Планер’. Для порівняння було підібрано один сервіс – https://todoist.com/home:, який має різну реалізацію на різних платформах: було виявлено 9 особливостей, які слід враховувати при порівнянні PWA з нативними додатками.

3.1 Скільки пам'яті пристрою додатки займають і як встановлюються

Найпомітніша відмінність між PWA та нативними додатками – об'єм пам'яті пристрою, який вони займають.

Це досягнуто інтеграцією в веб-сайт у мобільних та десктопних браузерах і не потребують завантаження. Щодо WEB-сайту, відкривати браузер щоразу може бути незручно. Замість цього створена опція "Додати на домашній екран". Тоді він буде виглядати як рідна іконка програми.

Існують випадки, такі як Twitter Lite або інші, коли ви можете встановити PWA з магазину додатків. Однак це актуально лише для Google та Microsoft, тоді як Apple не дозволяє публікувати PWA у своєму магазині додатків.

Це дозволяє використовувати перевагу меншої ваги PWA у порівнянні з нативними додатками. Більшість PWA мають розмір менше 1 МБ, що рідко зустрічається серед нативних додатків.

3.2 Продуктивність і швидкість

При порівнянні PWA з сайтом, PWA – явний переможець. Проте нативний додаток має кращу продуктивність:

Порівняння було проведено за допомогою сервісу browserstack та google page speed.

3.2.1 Функціональне тестування

Для тестування продуктивності сервісів було проведено функціональне тестування за наступним чек лістом.

1. Встановлення.
2. Видалення.
3. Оновлення версій.
   3.1. Вхід в Play маркет.
   3.2. Пошук додатку.
   3.3. Натискання кнопки оновити.
4. Запуск програми (відображення Splash Screen).
5. Працездатність основного функціоналу додатка.
   5.1. Авторизація (за номером телефону/через соц. мережі/e-mail).
   5.2. Реєстрація (за номером телефону/через соц. мережі/e-mail).
   5.3. Валідація обов'язкових полів.
   5.4. Навігація між розділами додатка.
   5.5. Редагування даних у профілі користувача.
   5.6. Тестування фільтрів.
6. Стресове тестування (не має з'єднання з інтернетом).
7. Скрол/свайп елементів.
8. Тестування PUSH повідомлень.
9. Згортання/розгортання сервісу.
10. Різні типи підключень (мобільний інтернет/Wi-Fi).
11. Робота додатка у фоні.
12. Політики конфіденційності та інші посилання на документи.

Результат функціонального тестування наведено у табл. 3.1.

Також результат тестування наведено у вигляді діаграми на рис. 3.1.

Таблиця 3.1 – Результат функціонального тестування

Параметри	Нативний додаток	PWA додаток	Веб-сайт
Встановлення	21 с	7 с	Не вимагає встановлення
Видалення	4 с	5 с	Не вимагає видалення
Оновлення	1хв 2с	Автоматичне	Автоматичне
Запуск програми (відображення Splash Screen)	0.8 с	0.7с	2 с
Авторизація (за номером телефону/через соц. мережі/e-mail)	32 с	33 с	40 с
Реєстрація (за номером телефону/через соц. мережі/e-mail)	45 с	48 с	1хв 4с
Додавання елементу	6 с	6 с	9 с
Валідація обов'язкових полів	Ідентично		
Навігація між розділами додатка	Ідентично		
Редагування даних у профілі користувача	Ідентично		
Тестування фільтрів	Ідентично		
Стресове тестування (не має з'єднання з інтернетом)	Безперебійна робота	Безперебійна робота	Робота неможлива
Скрол/свайп елементів	Ідентично		
Тестування PUSH повідомлень	Надсилання PUSH повідомлень без обмежень	Android, Windows – без обмежень IOS – обмежена	Обмежено
Згортання/розгортання сервісу	1хв 1с	0.9 с	1хв 12с
Різні типи підключень (мобільний інтернет/Wi-Fi)	Безперебійна робота	Безперебійна робота	Робота Wi-Fi швидше на 43% ніж мобільний інтернет
Робота додатка у фоні	Займає 80МБ.	Займає 10МБ.	Займає 40МБ.
Політики конфіденційності та інші посилання на документи	Ідентично		

Рисунок 3.1 – Результат функціонального тестування

3.2.2 Тестування сумісності

Тестування сумісності використовується, щоб переконатися, що застосунок сумісний з іншими версіями ОС, різними оболонками та сторонніми сервісами, а також апаратним забезпеченням пристрою.

Було перевірено сервіс на тестування переривань, а саме:

– вхідний дзвінок;

– смс;

– Push;

– будильник;

– режим "Не турбувати”.

Висновок: сервіси працюють ідентично.

Також перевірено сервіси під час підключення зовнішніх пристроїв:

– сім-карти;

– навушників;

– смарт годинників.

Висновок: сервіси працюють ідентично.

3.2.3 Стресове тестування

Стресове тестування спрямоване на визначення ефективності продуктивності програми в умовах підвищеного навантаження. Стрес-тест у цьому контексті орієнтований тільки на нативний та PWA додаток.

Було перевірено поведінку сервісу при таких умовах:

– високе завантаження центрального процесора;

– брак пам'яті;

– завантаження батареї;

– низька пропускна здатність мережі;

– велика кількість взаємодій користувача з додатком (штучно).

Результат стресового тестування наведено у таблиці 3.2.

Таблиця 3.2 – Результат стресового тестування

Параметри	Нативний додаток	PWA додаток	Веб-сайт
Високе завантаження центрального процесора	Показник подає на 74%	Показник подає на 72%	Безперебійна робота
Брак пам’яті	Показник подає на 80%	Показник подає на 74%	Безперебійна робота
Низький заряд батареї	Показник подає на 87%	Показник подає на 86.3%	Показник подає на 90%
Низька пропускна здатність мережі	Безперебійна робота	Безперебійна робота	Показник подає на 90%
Велика кількість взаємодій користувача з сервісом	Безперебійна робота	Безперебійна робота	Показник подає на 90%

Результат стресового тестування також наведено у вигляді діаграми по кожному з параметрів на рисунку 3.2-3.6.

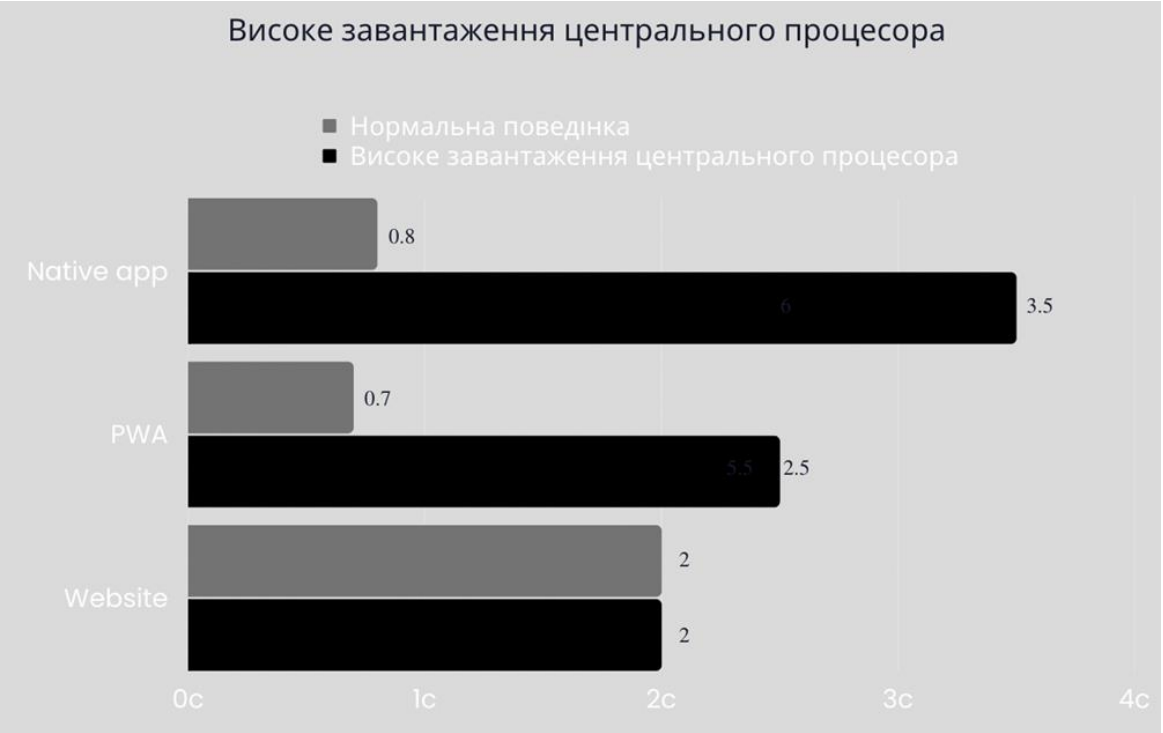

Рисунок 3.2 – Результат тестування на високе завантаження центрального процесора

Рисунок 3.3 – Результат тестування на брак пам’яті

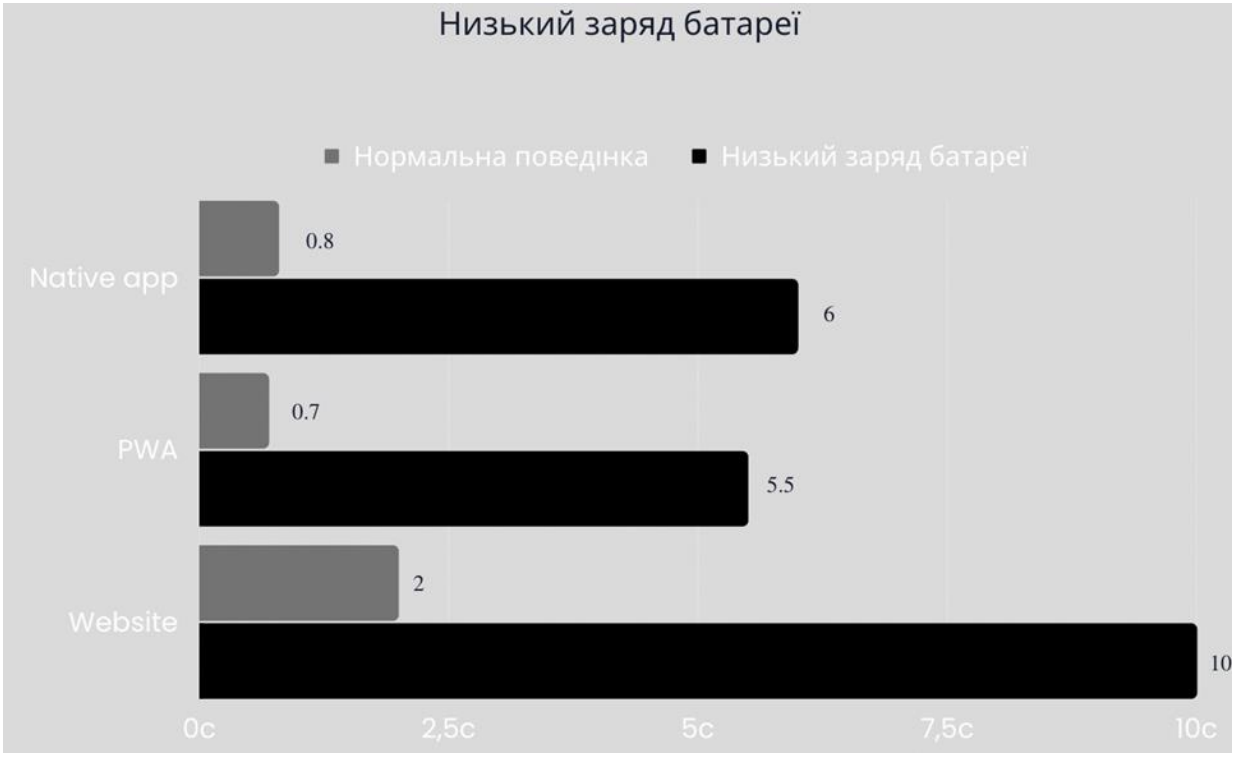

Рисунок 3.4 – Результат тестування на низький заряд батареї

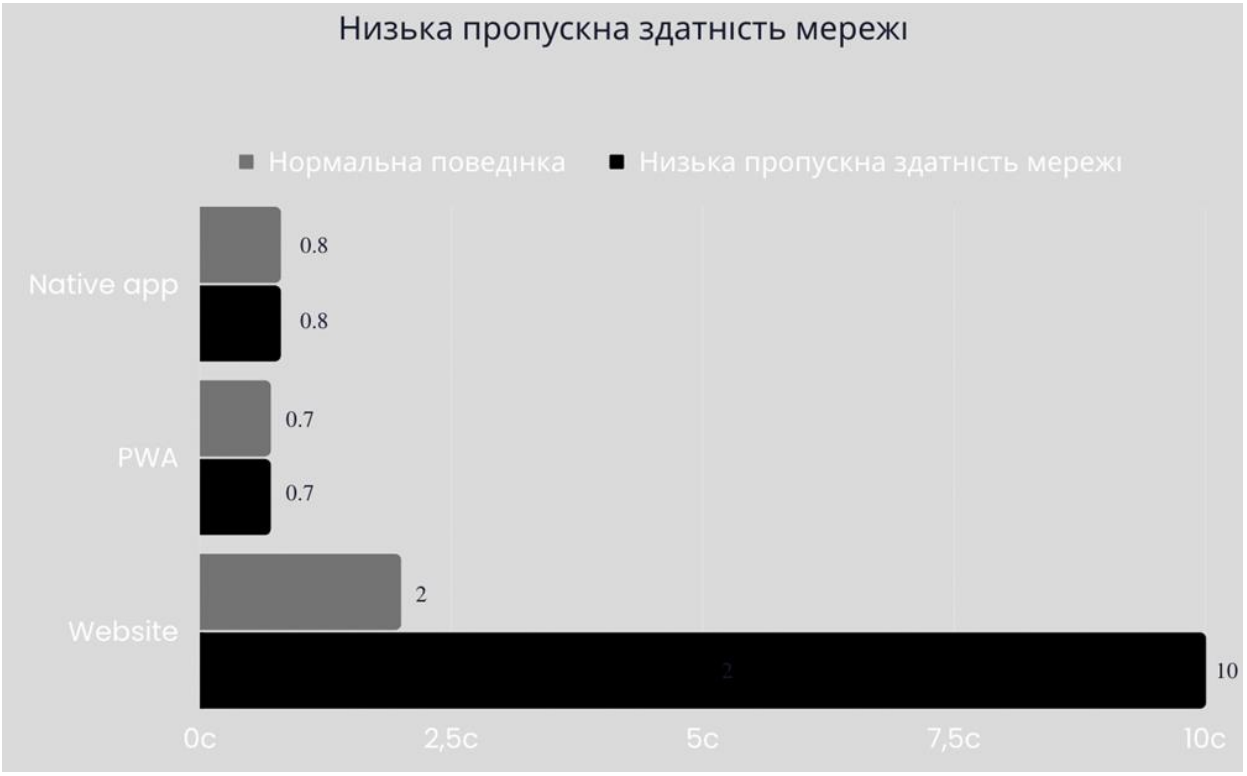

Рисунок 3.5 – Результат тестування на низьку пропускну здатність мережі

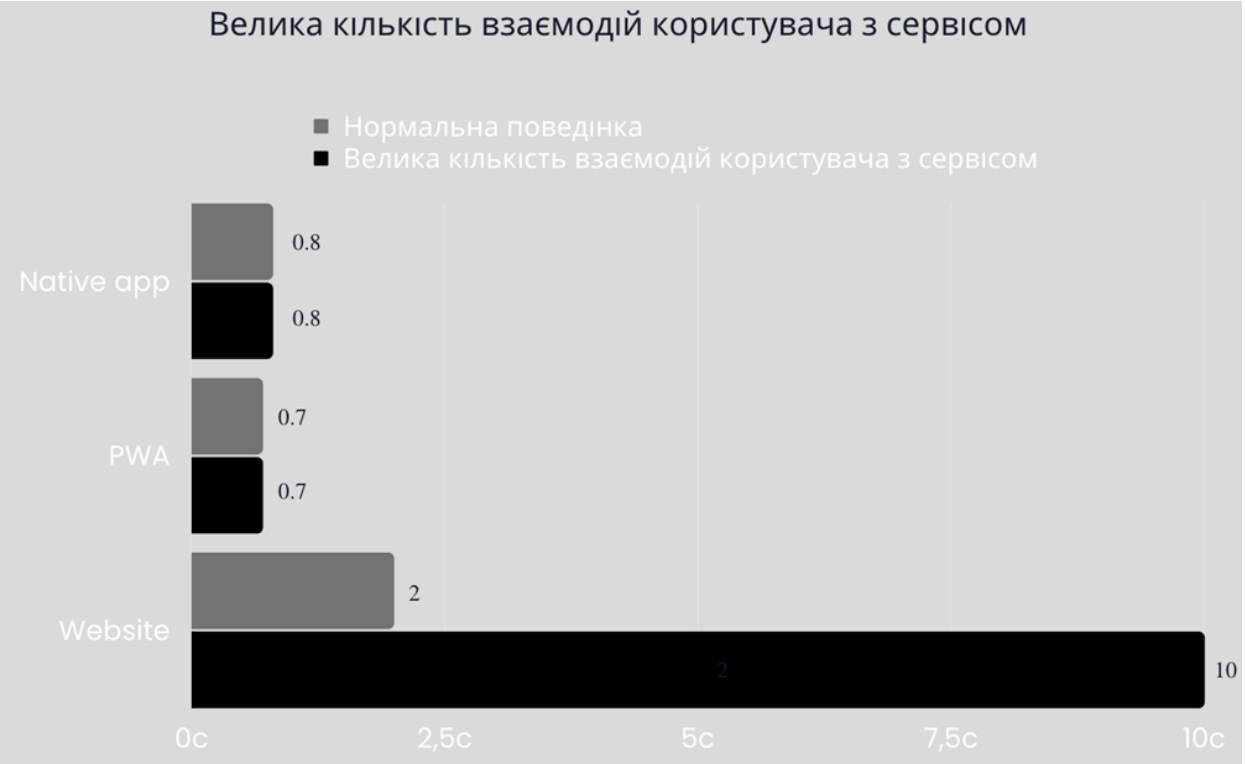

Рисунок 3.6 – Результат тестування на велику кількість взаємодій користувача з сервісом

Результат автоматизованого тестування продуктивності сервісів можно побачити на рисунку 3.7-3.8.

Продуктивність веб-сайту складає лише 22%, що являється критично. межою.

В основі будь-якої PWA лежать сервісні скрипти. Такі скрипти працюють у фоновому режимі і не залежать від веб-сторінки. PWA включають в себе просунуте кешування, GraphQL (сучасний метод API) та інші методи прискорення роботи сайту. Це робить їх такими ж швидкими, як і нативні додатки.

Оскільки PWA – це все ще веб-сайт, швидкість його сторінки залежить від усіх факторів мобільного веб-сайту, таких як час відгуку сервера і рендеринг CSS на критичному шляху, наприклад.

Але, PWA працюють через браузер. Це призводить до більшої затримки та споживання енергії, ніж у нативних додатків. Нативні додатки можуть бути інтегровані в операційну систему, що дозволяє їм отримувати

доступ до апаратного забезпечення пристрою для виконання більшої кількості обчислень і забезпечення кращого користувацького досвіду.

Отже, якщо ми подивимося на швидкість, то нативний додаток виграє, хоча і з невеликою перевагою. Нативний додаток є більш потужним, а нативний код на Kotlin/Swift є швидшим.

У данному випадку така різниця не є критичною.

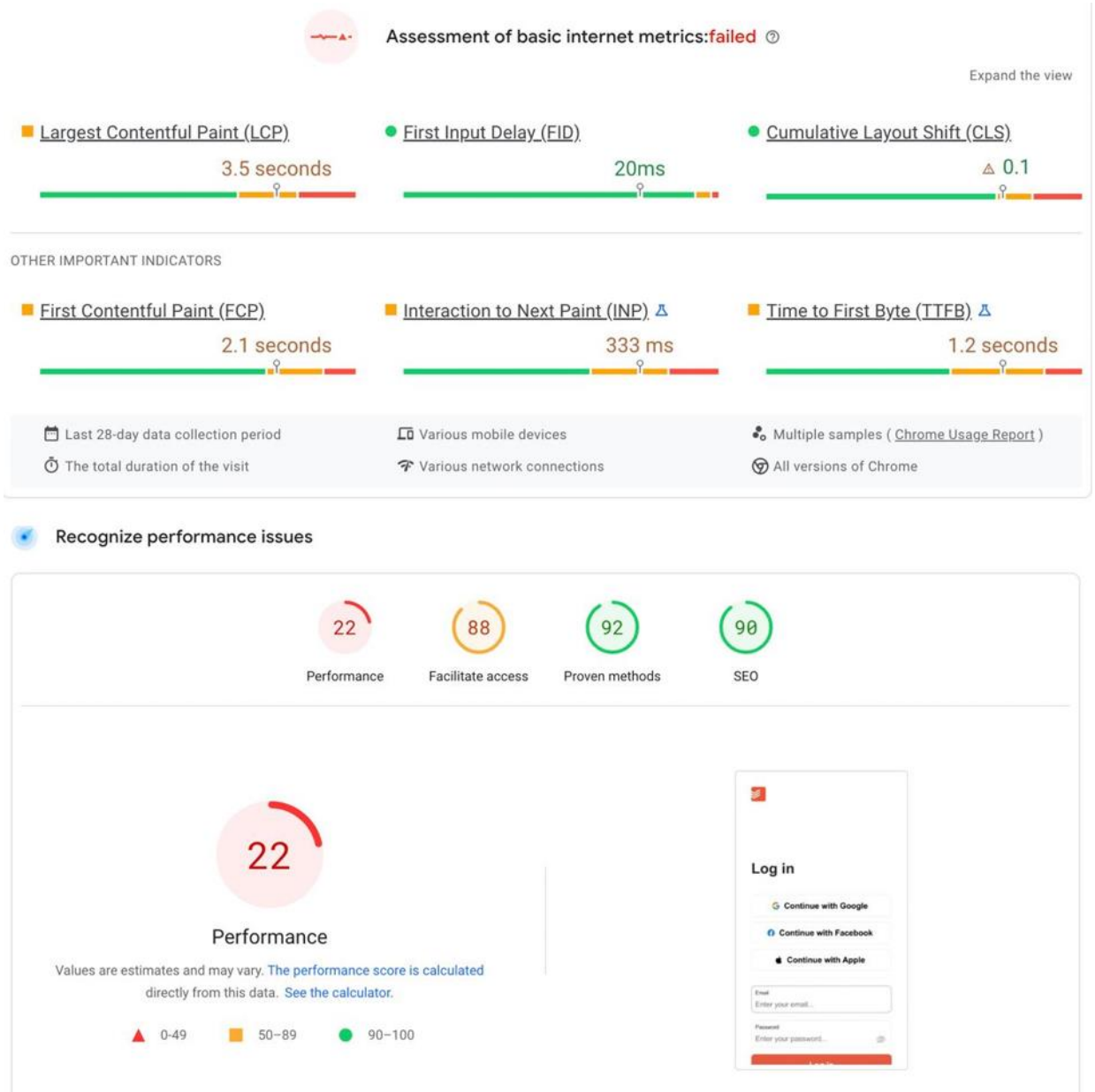

Рисунок 3.7 – Результат проведення тестування продуктивності веб-сайту

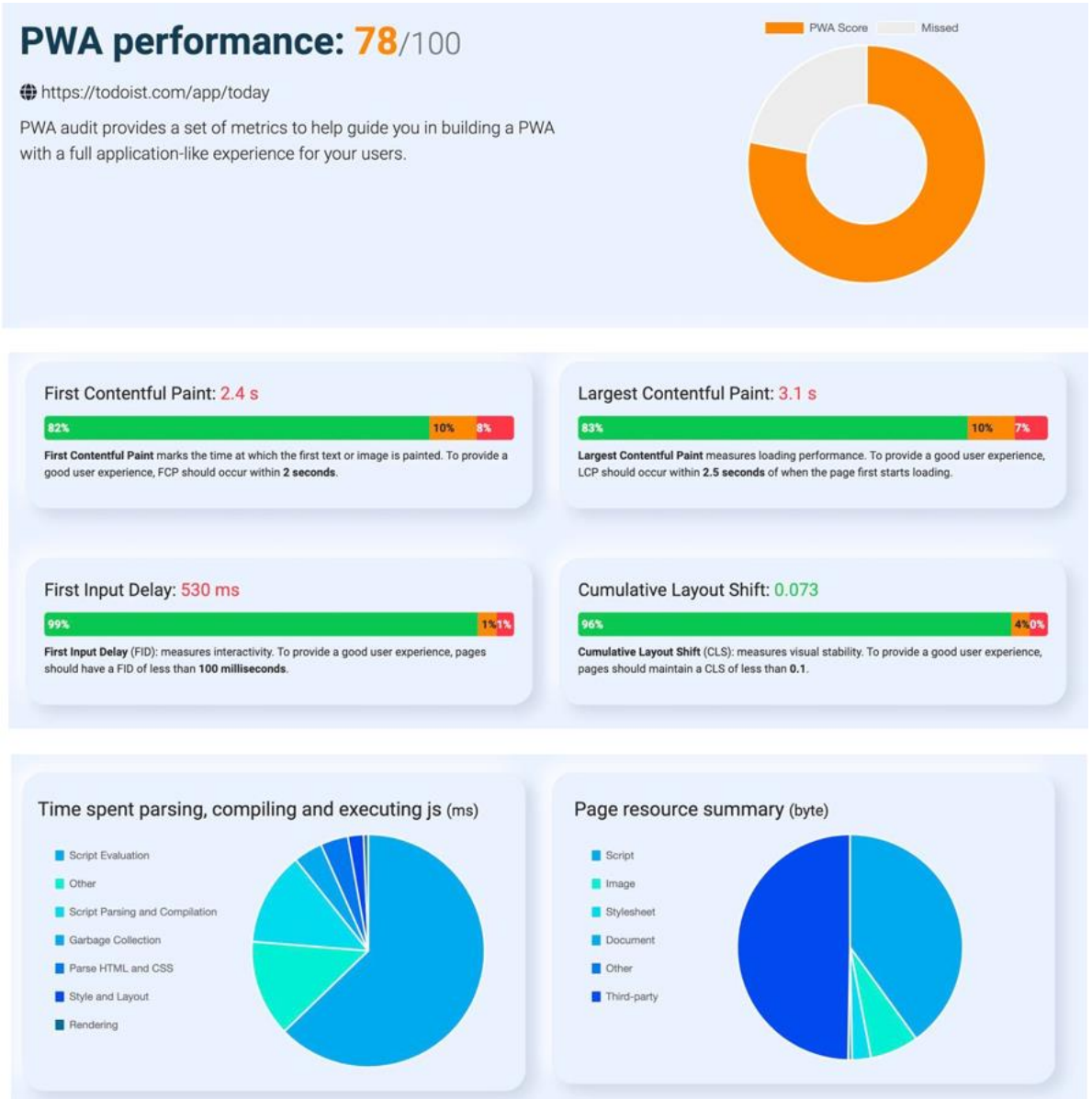

Рисунок 3.8 – Результат проведення автоматизованого тестування продуктивності PWA додатку

3.3 Особливості UI/UX

Чи є нативні додатки ідеальними, коли мова йде про мобільний UX? Вони дозволяють користувачам отримувати максимальну віддачу від своїх пристроїв, оскільки створюються саме для смартфонів, на відміну від веб-сайтів. Але їх створення вимагає ґрунтовних знань рідних мобільних мов або фреймворків, таких як Swift або , щоб спроектувати інтерфейс як мінімум для двох платформ.

З іншого боку, код PWA адаптується до різних платформ, забезпечуючи однакову функціональність та шаблон інтерфейсу для Android, iOS та настільних комп'ютерів.

Тим не менш, прогресивний веб-додаток слідують за нативними додатками з безшовним та інтуїтивно зрозумілим UI/UX, тому тут немає явного переможця з точки зору зручності для користувача.

PWA можуть відстежувати операційну систему на рівні додатків і використовувати різні бібліотеки інтерфейсу. Подібним чином можна відстежувати, чи користувач отримує доступ до програми з мобільного телефону або комп'ютера. Виходячи з цього, інтерфейс може змінюватися, тому ми можемо налаштувати PWA для відображення нативних елементів і підлаштовуватися під ОС.

PWA (ліворуч), веб-сайт та нативний додаток майже ідентичні:

– головне меню знаходиться вверху сторінки;

– навігація інтуїтивно зрозуміла;

– розділ профілю на PWA виглядає навіть більш приємним для ока, ніж у нативному додатку, але це суб'єктивно.

На рис. 3.9-3.11 наведені приклади інтерфейсу.

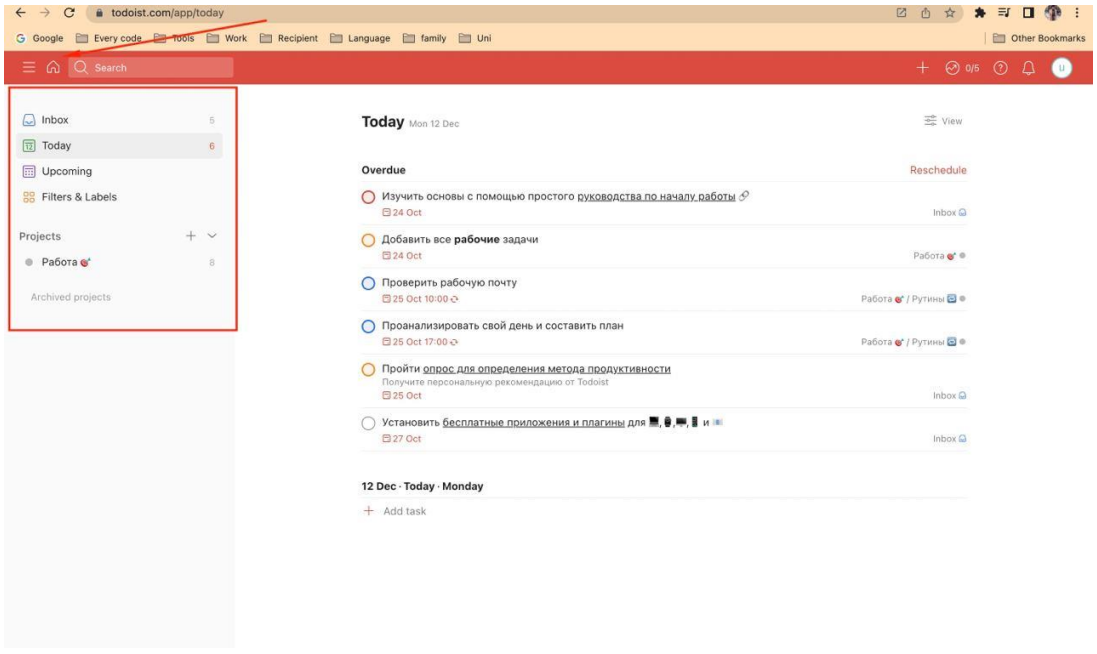

Рисунок 3.9 – Інтерфейс веб-сайту

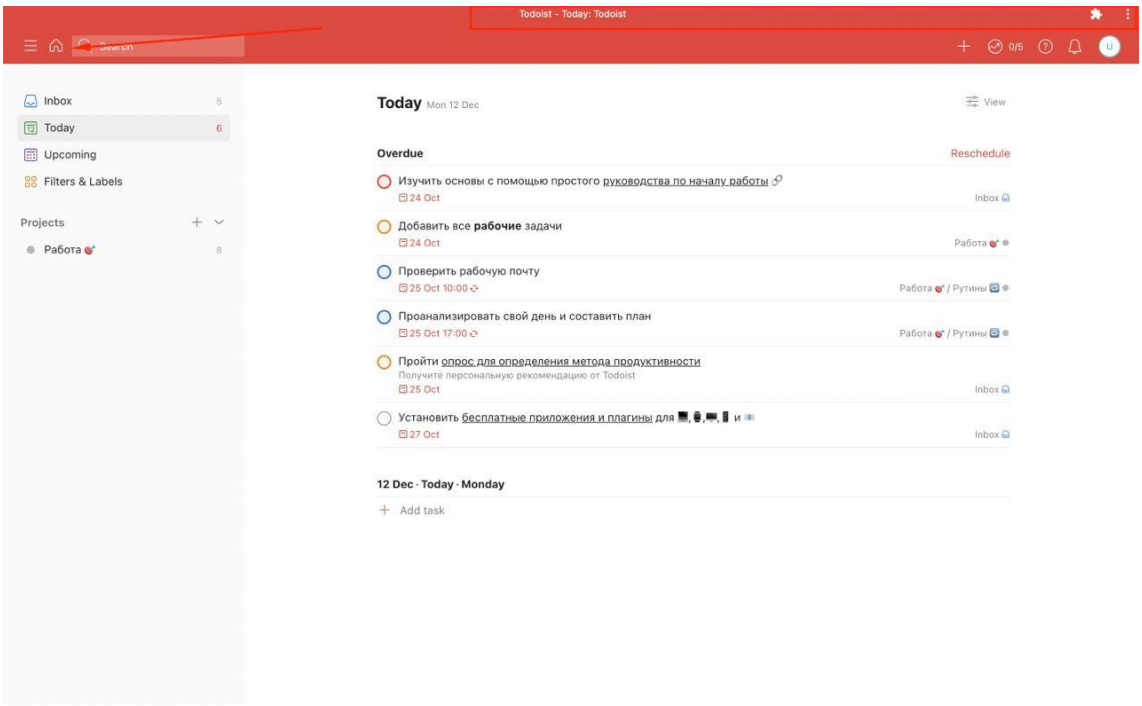

Рисунок 3.10 – Інтерфейс PWA додатку

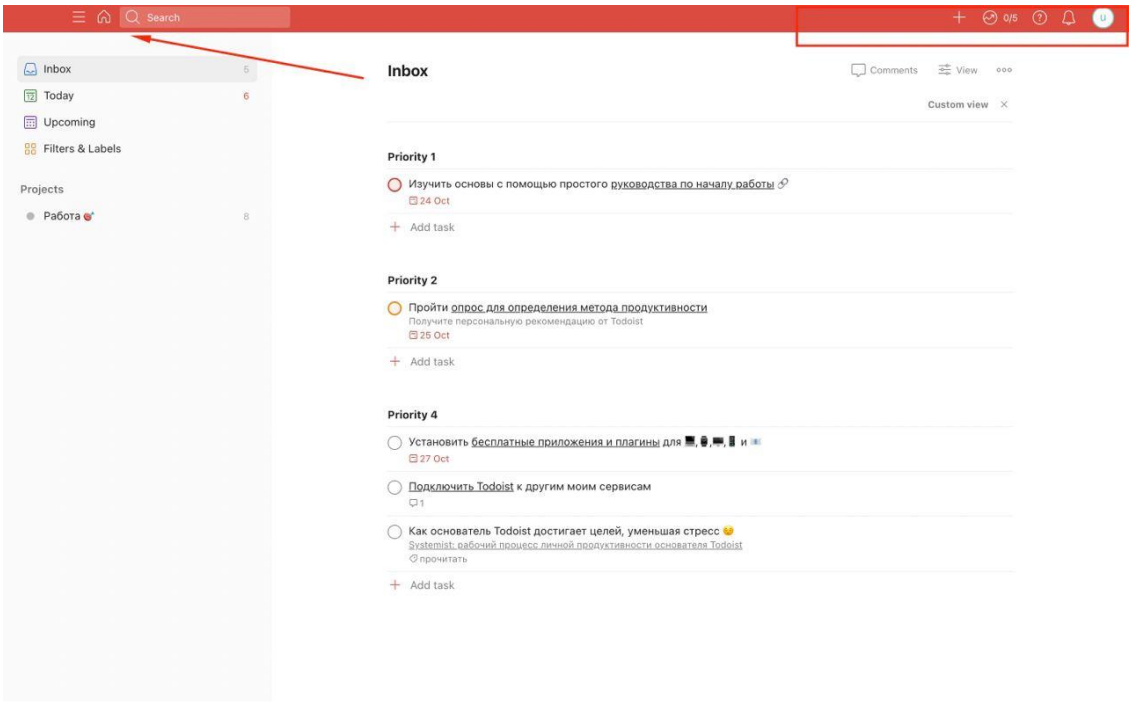

Рисунок 3.11 – Інтерфейс нативного додатку

3.4 Функція push-повідомлень

Push-повідомлення повторно залучають відвідувачів. Можливо створити цю функцію з нуля або інтегрувати її в додаток за допомогою сторонніх

сервісів, таких як PushBots або OneSignal. І вони безперебійно працюють на будь-яких ОС, тобто нативні додатки ставлять галочку на цьому пункті.

PWA дозволяють надсилати пуш-повідомлення завдяки працівникам сервісу. Однак ця функція варіюється в залежності від платформи. Вона доступна в Chrome, Opera, Safari та Mozilla, але з iOS не все так просто. Push-повідомлення на PWA добре працюють на Android, але для користувачів iOS ця функція обмежена.

Ці обмеження будуть подолані в міру розвитку технології, що поставить PWA і нативні додатки в більш рівні умови. Але поки що останні в цьому відношенні лідирують.

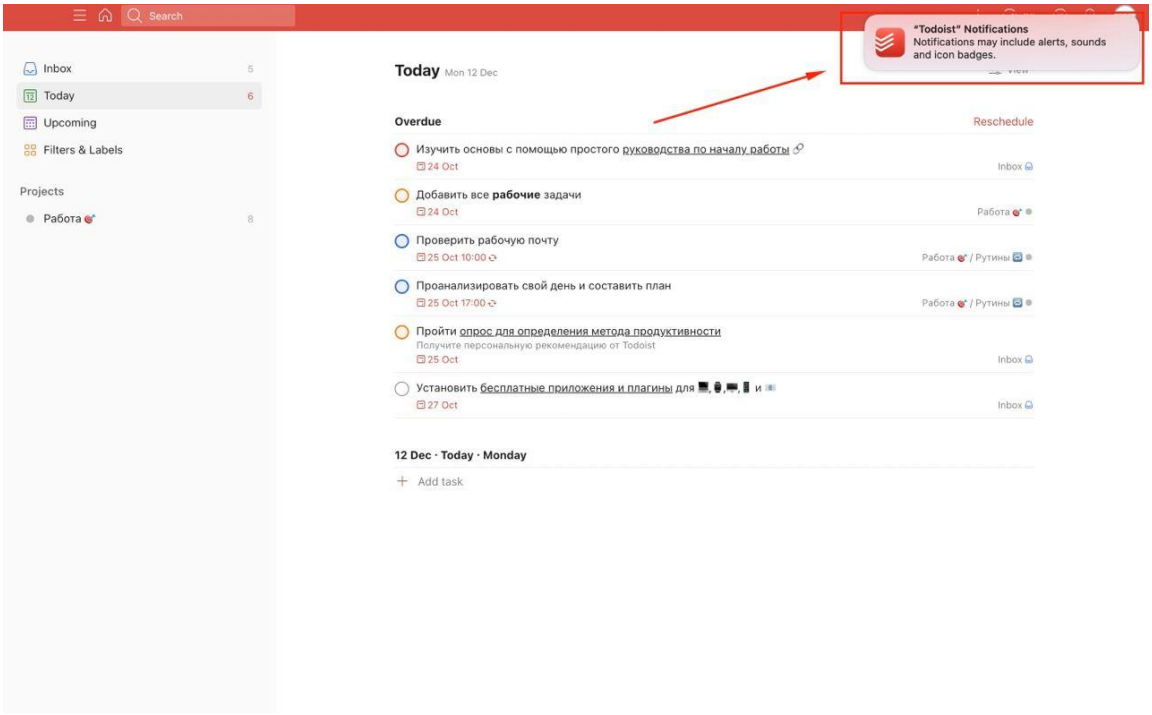

Рисунок 3.12 – Push-повідомлення\

3.5 Офлайн-режим

Обидва рішення PWA та нативний додаток забезпечують офлайн-доступ і працюють при повільному інтернет-з'єднанні.

Сервісні працівники PWA дозволяють управляти запитами в автономному режимі, здійснювати попередню вибірку, кешувати певні

ресурси та синхронізувати дані з віддаленим сервером. Після додавання програми на домашній екран можливо одразу запустити її і користуватися нею в автономному режимі або в умовах низької пропускної здатності.

Сторінки, які були відкриті раніше, повернуть результати. Після відновлення з'єднання інформація буде оновлена. Таким чином, як PWA, так і нативні додатки дозволяють роботу режимі офлайн (незалежно від стабільності інтернет-з’єднання).

3.6 Можливості та обмеження

Нативні додатки призначені для роботи в певному програмному середовищі. Наприклад, нативний додаток для iOS підтримує специфічні функції Apple, такі як Face ID. Додатки для Android та Windows також отримують їх апаратні властивості.

Прогресивні веб-додатки повинні мати справу з деякими обмеженнями при використанні HTML, JS та CSS. Ці обмеження залежать від платформ або систем, на яких працюють додатки. Наприклад, iOS не підтримує PWA push-повідомлення. На противагу цьому, Android має кращу підтримку PWA і покращує свою екосистему швидше, ніж iOS.

Отже, PWA не має доступ до низькорівневих апаратних функцій нативних додатків.

3.7 Доступність для пошукових систем

Пошукові системи не ранжують нативні програми. Отже, не можливо використовувати SEO-стратегії для підвищення видимості бренду серед потенційних клієнтів. Що стосується PWA, то Google індексує їх як веб-сайти. Крім того, швидкість і мобільність PWA також вважаються важливими факторами SEO ранжування.

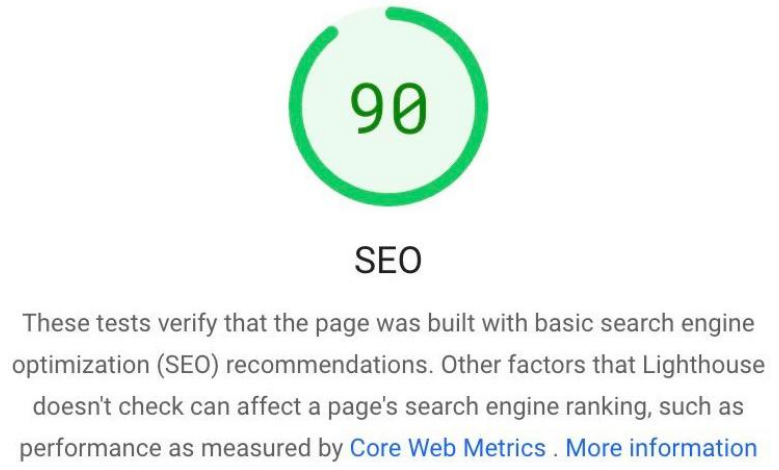

Рисунок 3.13 – SEO складова веб-сайту та PWA додатку

SEO складова веб-сайту та PWA додатку має результат у 90%, коли у нативного додатку взагалі не має SEO складової (рисунок 3.13).

3.8 Безпека

Нативні додатки вважаються більш безпечними з наступних причин:

– вони можуть використовувати багатофакторну аутентифікацію;

– забезпечують більш безпечну комунікацію за допомогою закріплення сертифікатів.

Процедури автентифікації та вимоги до безпеки Google Play та iOS App Store.

Коли нативні додатки отримують доступ до апаратного забезпечення, операційна система надає додатку дозвіл на цей модуль. Як наслідок, нативний додаток повинен бути більш надійним, ніж URL-адреса.

У той же час, PWA не позбавлені вирішального фактору безпеки, оскільки вони повинні працювати через HTTPS і мати такі функції, як маніфест веб-додатків і сервісні працівники. Це гарантує, що передача даних між клієнтом і сервером захищена. Таким чином, вміст і взаємодія настільки безпечні, наскільки це можливо на захищеному веб-сайті.

Безпека PWA покладається на браузер як на віртуального захисника. PWA працює в пісочниці браузера, і його можливості вже обмежені цим суворо обмеженим та ізольованим середовищем. Тому він може отримати доступ тільки до ресурсів браузера і до того, що дозволяє браузер. Це ізолює додаток від апаратного забезпечення смартфона та конфіденційних даних користувача.

3.9 Витрати на розробку

Розробка та підтримка PWA коштує дешевше, ніж нативні додатків. Це не те саме, що витрачати гроші на окремі нативні додатки для Android та iOS, кожен з яких має власний набір інструментів розробки та мови коду. Крім того, PWA сумісні з різними браузерами, тому додаток можна використовувати як на настільних, так і на мобільних пристроях.

Нативні додатки вимагають вивчення мови та створення окремої версії для кожної платформи (щонайменше одна версія для iOS та одна для Android).

Таким чином можна зробити висновок, що переваги PWA полягають у:

– PWA працюють у будь-якому браузері на десктопних та мобільних пристроях;

– додаток швидко завантажується та використовується на будь-якому пристрої та з будь-яким підключенням до Інтернету (і навіть є можливість працювати в офлайн-режимі);

– PWA реалізували найкращі практики UI/UX нативних додатків, щоб залучити користувача за допомогою інтуїтивно зрозумілого та привабливого інтерфейсу;

– вони включають в себе просту функцію "Додати на домашній екран", що дозволяє зберегти легкий PWA на своєму пристрої як ярлик для швидкого доступу;

– за зразком нативних додатків, вони підтримують push-повідомлення, які мають вирішальне значення для утримання клієнтів;

– оскільки прогресивний веб-додаток – це веб-сайт, його сторінки можуть бути виявлені пошуковими системами.

З іншого боку, у PWA є і недоліки:

– існує обмежений набір апаратних функцій, доступних для PWA для iOS. Багато функціональних можливостей все ще заблоковані в підтримці iOS PWA;

– однією з найбільш критичнх відсутніх функцій є можливість надсилання push-повідомлень користувачам iOS;

– PWA споживають більше заряду акумулятора. Оскільки додаток розробляється за допомогою фреймворків, PWA вимагає більше процесорної потужності і виснажує акумулятор.

Переваги нативних додатків полягають у наступному:

– вони завантажуються в пам'ять мобільного пристрою, що дає їм повний доступ до апаратного забезпечення, API, функціональних можливостей і даних користувача. Вони також сумісні з іншими продуктами Google та Apple;

– вони розроблені виключно для мобільного використання для інтеграції з ОС та мають відшліфований UI/UX;

– з огляду на цей набір висновків, PWA краще підходять для простих додатків які не мають великої кількості низькорівневих функцій.

4 ТЕХНОЛОГІЧНІ ПЕРСПЕКТИВИ ТА ОБМЕЖЕННЯ

Значні зусилля, орієнтовані на розробників, були представлені під час Google I/O 2017, де Османі продемонстрував технічну базову лінію для тестування фреймворків JavaScript на відповідність критеріям PWA. Базова лінія, названа HNPWA, має на меті допомогти розробникам вибрати фреймворк JavaScript для створення PWA. До тесту включені численні фреймворки, такі як React, Preact, Svelte, Vue.js та Angular. Всі кодові бази тестів мають відкритий вихідний код, і тести з використанням нових фреймворків та підходів можуть бути додані на веб-сайт HNPWA для вивчення іншими.

Для PWA доступ до API пристроїв і платформ все ще обмежений тими API, які підтримуються браузерами користувачів. Це одне з основних обмежень підходу PWA у порівнянні з нативною або крос-платформною розробкою, де всі або більшість відкритих API пристроїв доступні розробнику через нативні SDK або подібні абстракції. Дещо обнадіює те, що команда Google Chrome додала 215 нових API лише за останній рік, тому прогрес у подоланні цього розриву, безсумнівно, покращується. Тим не менш, з невід'ємним майбутнім впровадженням нових функцій мобільних платформ, доступ PWA до них все ще обмежений до тих пір, поки специфікація HTML/JavaScript не буде формалізована W3C і (або принаймні) не буде впроваджена виробниками браузерів. Фрагментація платформ, версій операційних систем та можливостей браузерів і пристроїв не сприяє руху Мережі до уніфікації. Навіть у сфері мов програмування Інтернет може зазнати подальшої фрагментації. Помітною технологією, що повільно набирає обертів у галузі, є WebAssembly (wasm) – формат і засіб для написання веб-додатків з використанням інших мов, окрім JavaScript. Це може відкрити можливості для портування програмного забезпечення, зокрема ігор, які традиційно працювали лише як нативні настільні додатки.

Для веб-додатків та PWA підтримка декількох мов може залучити нових розробників до платформи, що може призвести до до подальшого прийняття та розповсюдження таких додатків [18].

4.1 Статус Quo прийняття PWA

Під час конференції розробників Google, Google I/O 2017, низка ініціатив щодо прогресивних веб-додатків була представлена як на сцені, так і у вигляді згадок. Великі компанії та ключові гравці мобільного веб-простору вже почали конвертувати свої існуючі веб-додатки в PWA з великим успіхом.

До них відносяться вже згадані вище Twitter та OLA, які є провідними компаніями у своїх галузях. Компанії також оприлюднили статистичні дані про розміри додатків і повідомили про збільшення їх використання після прийняття PWA.

Нативні додатки Twitter для Android та iOS мають розмір відповідно > 23 Мб та > 100 Мб, в той час як їх PWA важить 0,6 Мб (600 Кб), все ще забезпечуючи більшість очікуваних функцій. Вони також бачать більше мільйона запусків на домашньому екрані щодня, і виявили, що їх PWA, Twitter Lite, має велике значення для ринків, що розвиваються. Аналогічно, OLA, найбільший в Індії додаток для виклику таксі, виявив, що його PWA важить 0,20 Мб (200 Кб), що значно менше, ніж їхні рідні додатки для Android та iOS, відповідно, 60 Мб та 100 Мб. OLA класифікує своїх клієнтів за рівнями залежно від їхнього місцезнаходження, де рівень 3 представляє райони з дуже низьким рівнем зв'язку в Індії, а рівень 1 – райони з хорошим або задовільним зв'язком. OLA виявила, що в районах 3-го рівня спостерігалося збільшення мобільного трафіку на 68% з моменту запуску їх PWA, а коефіцієнт конверсії був на 30% вище, ніж у їх рідному додатку. У регіонах 2-го рівня коефіцієнт конверсії з їхньої PWA був таким же, як і в їхньому рідному додатку.

Це ілюструє бізнес-потенціал і важливість PWA для такої компанії, як OLA, яка працює в різних сферах, маючи справу з ситуаціями з низьким рівнем зв'язку [19].

Серед інших відомих компаній, що працюють над PWA, – Forbes, Financial Times, Lyft, Expedia, AliExpress, Tinder, Flipkart та Housing.com. Ці компанії та їхні продукти представляють різні ніші та ринки, ілюструючи як люди з особливими потребами можуть бути корисними будь-де.

Також під час конференції було проведено сім переговорів, спрямованих на розвиток та ентузіазм щодо цього можливого наступного покоління мобільного Інтернету. PWA обговорювалися в контексті мобільного користувацького досвіду (UX), підтримки технічними рамками, тестування продуктивності та міграції. Очевидно, що Google просуває PWA як спробу покращити користувацький досвід мобільного Інтернету.

Мобільна розробка, незалежно від того, чи є вона нативною, крос-платформною або веб-орієнтованою, швидко розвивається завдяки постійним потокам нових технічних фреймворків, підходів і методів. Результатом цього є фрагментація спільнот розробників, підходів до розробки та думок про те, як слід розробляти додатки.

По-перше, з впровадженням Kotlin як першокласної мови програмування на платформі Android, більше розробників, ймовірно, можуть долучитися до розробки під Android. Однак, додатково цікавою є нова розробка компанії Jetbrains, творців Kotlin, під назвою Kotlin/Native. Цей бекенд на основі LLVM допоможе розробникам запускати Kotlin на платформах, що не є (J)VM, таких як iOS. Таким чином, мова Kotlin може бути придатною для крос-платформної розробки в найближчому майбутньому, з підтримкою за замовчуванням на Android. Кросплатформенна підтримка не буде обмежуватися Android та iOS; також повідомляється, що підтримуються такі платформи, як MacOS, Ubuntu та Raspberry Pi. Оскільки Kotlin також може транслюватися в JavaScript, подібно до інших рішень, таких як CoffeeScript та TypeScript, розробники тепер

можуть використовувати Kotlin для розробки програмного забезпечення на різних платформах.

По-друге, ми є свідками поширення технічних фреймворків та підходів до розробки, орієнтованих на мобільні пристрої. Якщо в попередніх дослідженнях часто згадуються крос-платформні фреймворки для додатків, такі як PhoneGap, Titanium Appcelerator, DragonRad і Rhodes, то останнім часом з'явилося безліч нових і більш технічно оновлених фреймворків.

Хоча останні роботи охоплюють такі фреймворки, як React Native, Ionic та Fuse, але інші рішення, включаючи NativeScript, Quasar та Apache Weex ще не вивчені на такому ж рівні, як, наприклад, PhoneGap в академічному контексті. Оскільки PWA набувають все більшої популярності, буде запроваджено ще один напрямок таких фреймворків, від розробки нативних додатків, скоріше, до орієнтації на Web без будь-яких нативних абстракцій, таких як Cordova [20].

4.2 Відкрите питання

У данній роботі було застосовано підхід, який поєднує кілька методологічних аспектів: він поєднує в собі дослідження літератури, теоретичний внесок і позиційний документ. Хоча цей синтез дозволив відповісти на багато питань, відповісти на основне питання дослідження вдалося лише до певної міри. Як наслідок, виникли відкриті питання. Прийняття PWA може впасти за відсутності підтримки iOS.

Хоча на Apple може чинитися технологічний тиск, дуже малоймовірно, що PWA стануть об'єднуючим фактором, якщо пристрої на базі iOS не будуть підтримуватися. Apple показала в минулому, що вона може займати дуже послідовну позицію, якщо вона не впевнена в технології, як це було продемонстровано у випадку з Adobe Flash.

З цією невизначеністю пов'язана проблема прогнозування прийняття індустрією. Технологічний прогрес все ще настільки стрімкий, що навіть

обґрунтовані оцінки можуть легко виявитися помилковими. Такі показники, як технологічна перевага або поточна ринкова влада, не є надійними, як показала історія розвитку крос-платформних додатків до теперішнього часу. Також залишається неясним, чи може бути досягнутий проривний технологічний прогрес у сфері смартфонів і планшетів. Посилення уваги до голосового управління, віртуальної реальності, доповненої реальності і навіть технологій, які все ще мають науково-фантастичну привабливість (наприклад, нейронні інтерфейси), можуть змінити правила гри – або ні.

Носимі пристрої та інші новітні мобільні пристрої, ймовірно, отримають ще більший імпульс у своєму розвитку – і присутності на ринку. Наразі незрозуміло, наскільки добре PWA підходять для настільних пристроїв. Ще більш сумнівним є те, чи мають вони сенс для пристроїв Інтернету речей (IoT), які можуть не мати графічного інтерфейсу користувача. Майбутні дослідження повинні вивчити цей напрямок. Зокрема, було б важливо дізнатися про прогалини, які PWA (та інші крос-платформні підходи) створюють щодо IoT. Враховуючи всі технологічні аспекти, важливим питанням є те, наскільки добре PWA будуть працювати, коли прийдуть наступні технологічні кроки. Таке ж питання, звичайно, потрібно поставити задавати будь-якій можливій об'єднуючій технології.

5 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

У даній роботі проведено експеримент з виявлення найвдалішого веб-додатку, на прикладі додатків для планування. Одним з ефективних методів є метод експертних оцінок, який найкраще підходить, щоб визначити вподобання споживачів.

Методи експертних оцінок – це спосіб прогнозування та оцінки майбутніх результатів дій на основі прогнозів фахівців.

При застосуванні методу експертних оцінок проводиться опитування спеціальної групи експертів з метою визначення певних змінних величин, необхідних для оцінки досліджуваного питання.

Необхідною умовою ефективного застосування методів експертної оцінки є достатня обізнаність експерта з досліджуваної проблеми, високий рівень ерудиції, здатність його давати чіткі вичерпні відповіді. Крім того, експерт не повинен бути зацікавленим в тому чи іншому варіанті вирішення поставленої перед ним проблеми. Експерти підбираються за ознакою їх формального професійного статусу – посади, наукового ступеня, стажу роботи та ін. Такий підбір сприяє тому, що в число експертів потрапляють високопрофесійні, з великим практичним досвідом у даній галузі спеціалісти.

Метод експертних оцінок – це фактично метод прогнозування, основоположним критерієм якого є досягнення згоди серед усіх членів експертної групи.

Для проведення експерименту запропоновано 3 варіанти сервісів для планування. Експертами було обрано людей фахівців в даній сфері, які оцінювали веб-додатки. Для експерименту було підібрано один сервіс на різних платформах та з різною реалізацією (рис. 5.1-5.3):

– нативний додаток;
– PWA додаток;
– веб-сайт.

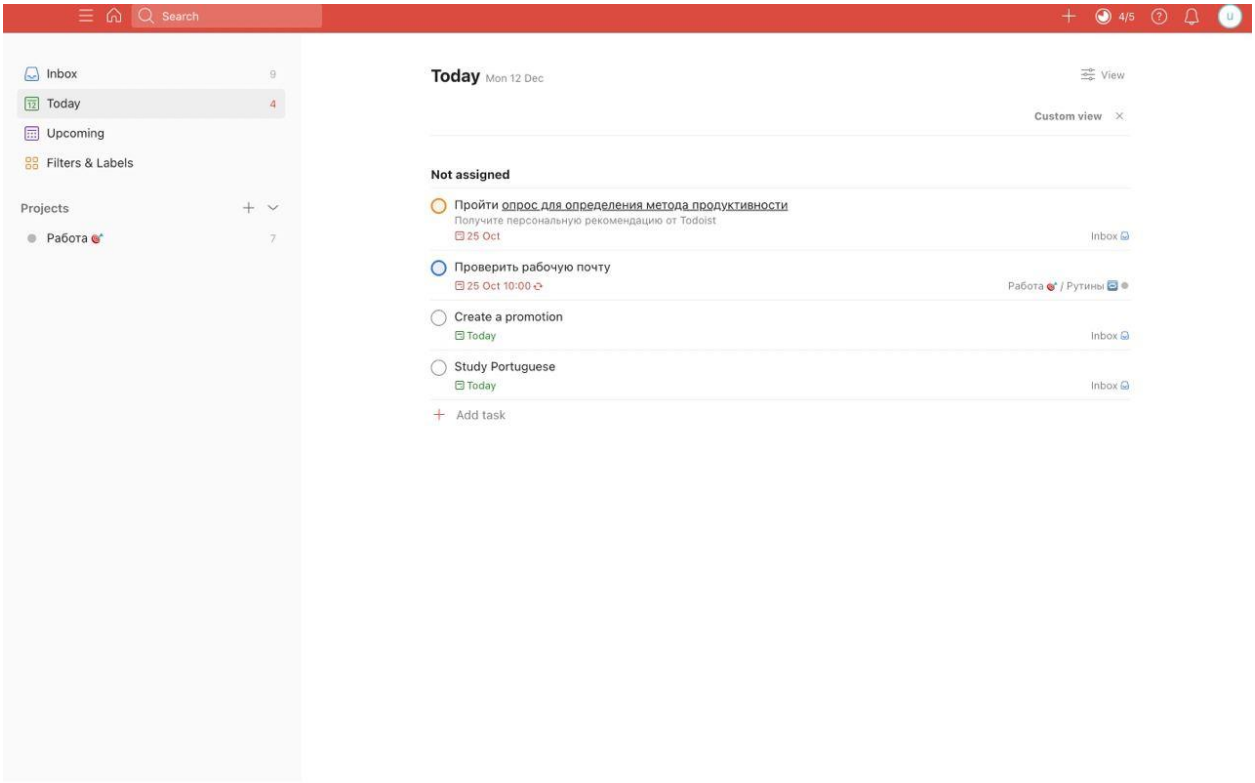

Рисунок 5.1 – Нативний веб-додаток

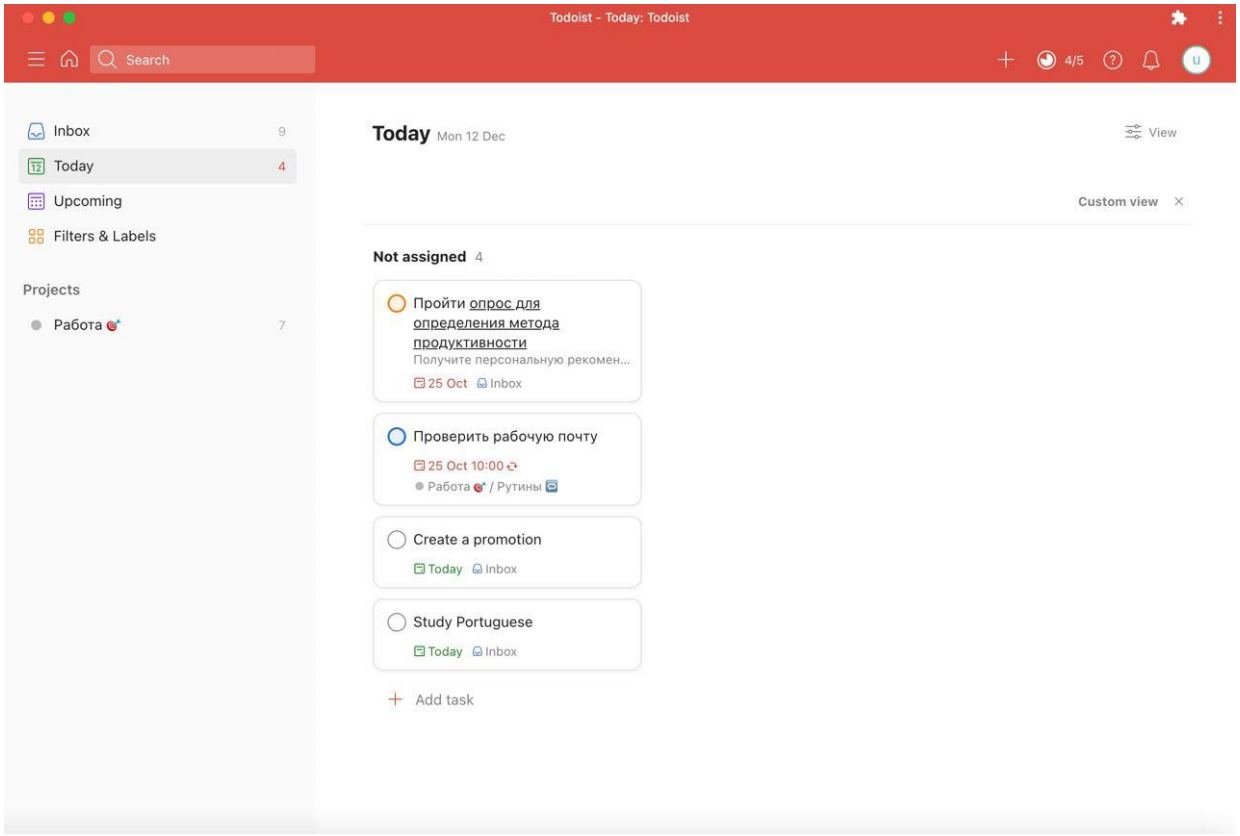

Рисунок 5.2 – PWA додаток

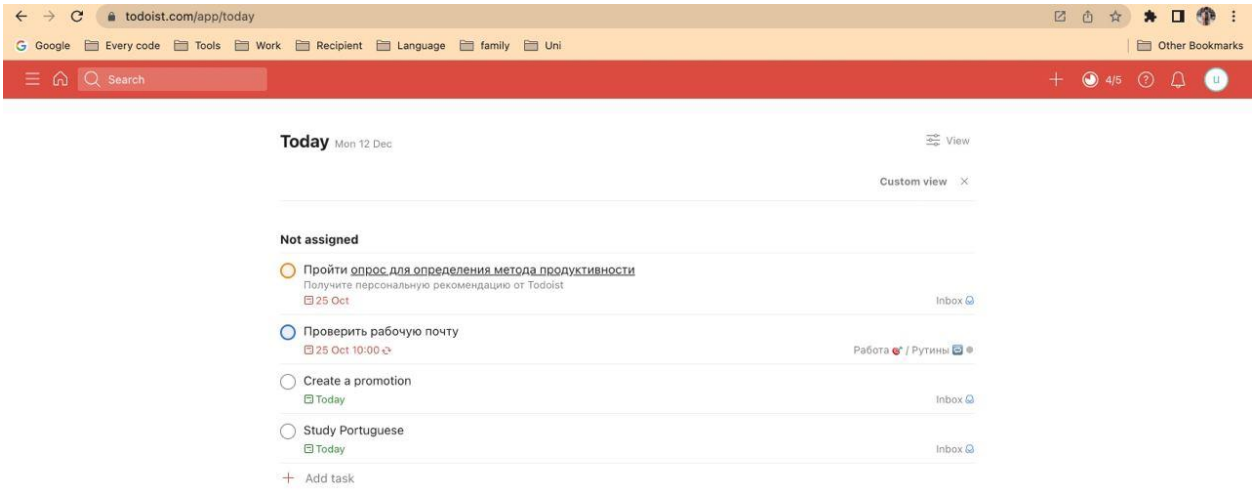

Рисунок 5.3 – Веб-сайт

Для оцінки експертам було запропоновано три критерії у вигляді анкети (Додаток А): установка (на скільки зручною та швидкою є установка), Користувацький досвід (UX), та офлайн доступ (його зручність використання). Після опитування експертів проводиться розрахунок результатів для виявлення найкращої з альтернатив для цього спочатку було складено таблицю 5.1 та розраховано коефіцієнт узгодженості.

Таблиця 5.1 – Критерій оцінки

Критерій	Оцінка експерта					Строкова сума	Вага	Відх. Від ср. Зн.	Квадрит. Відх.
	Ек. 1	Ек. 2	Ек. 3	Ек. 4	Ек. 5				
Установка	2	2	2	1	1	8	0,2	2	4
Користувацький досвід (UX)	4	4	4	3	3	19	0,47	9	81
Офлайн доступ	3	3	3	2	2	13	0,32	3	9
						Σ 40	Σ 1		

Вага кожного з критеріїв розраховується діленням їх строкової суми на загальну суму всіх строкових сум.

Для визначення коефіцієнта конкордації використовується формула:

$$W = \frac{12*S}{n^2(m^3-m)}, \quad (5.1)$$

де n – кількість експертів;

m – кількість критеріїв;

S – сума квадратичного відхилення.

Отже:

$$S = 4 + 81 + 9 = 94,$$

$$W = \frac{12*94}{5^2(3^3-3)} = \frac{1128}{600} = 1{,}88.$$

Виходячи з результату, який отримано, W=1,88, можна казати про достатню узгодженість думок експертів.

Далі представлено розрахунки порівняння альтернатив за оцінками експертів.

5.1 Критерії №1. Установка

Наскільки зручним, швидким та зрозумілим є процес встановлення додатку. Коефіцієнт конкордації обчислюємо за (5.1):

$$S = 4+ 49 + 25 = 78,$$

$$W = \frac{12*78}{5^2(3^3-3)} = \frac{936}{600} = 1{,}6.$$

Виходячи з цього значення можна зробити висновки, що оцінка проводилась досить узгоджено, тобто думки експертів зійшлись.

Таблиця 5.2 – Оцінка установки

		Експерт					Строкова сума	Вага альтернативи	Відхилення від ср.знач.	Квардрат відхилиння від ср.знач.
		1	2	3	4	5				
Сервіс	1	3	3	3	4	2	15	0,35	2	4
	2	4	4	4	4	4	20	0,5	7	49
	3	1	1	1	1	1	5	0,1	5	25
Середнє значення							10			

5.2 Критерії №2. Користувацький досвід (UX)

Наскільки зручним є користування безпосередньо навігацією.

Таблиця 5.3 – Оцінка користувацького досвід (UX)

		Експерт					Строкова сума	Вага альтернативи	Відхилення від ср.знач.	Квардрат відхилиння від ср.знач.
		1	2	3	4	5				
Сервіс	1	3	3	3	3	3	15	0,37	5	25
	2	3	3	4	3	2	14	0,35	4	16
	3	3	3	1	2	2	11	0,2	1	1
Середнє значення							10			

Коефіцієнт конкордації обчислюємо за (3.1).

$$S = 25 + 16 + 1 = 42,$$

$$W = \frac{12*42}{5^2(3^3-3)} = \frac{504}{600} = 0{,}84.$$

Виходячи з цього значення можна зробити висновки, що оцінка проводилась досить узгоджено, тобто думки експертів зійшлись.

5.3 Критерії No3. Офлайн доступ

Наскільки зручним та достовірним є зручність при використанні сервісу в офлайн режимі.

Таблиця 5.4 – Оцінка офлайн доступу

		Експерт					Строкова сума	Вага альтернативи	Відхилення від ср.знач.	Квардрат відхилиння від ср.знач.
		1	2	3	4	5				
Сервіс	1	4	4	3	4	3	18	0,45	8	64
	2	3	3	3	4	4	17	0,4	7	49
	1	1	1	1	1	1	5	0,1	5	25
Середнє значення							10			

Коефіцієнт конкордації обчислюємо за (5.1).

$$S = 64+ 49 + 5 = 118,$$

$$W = \frac{12*118}{5^2(3^3-3)} = \frac{1416}{600} = 2{,}36.$$

Виходячи з цього значення можна зробити висновки, що оцінка проводилась не дуже узгоджено, тобто думки експертів розійшлись.

Для підведення підсумку та остаточного вибору найкращої з запропонованих альтернатив були проведені розрахунки вагових коефіцієнтів:

$$Q_1 = 0{,}2 \times 0{,}35 + 0{,}47 \times 0{,}37 + 0{,}32 \times 0{,}45 = 0,\ 38;$$

$$Q_2 = 0{,}2 \times 0{,}5 + 0{,}47 \times 0{,}35 + 0{,}32 \times 0{,}4 = 0.39;$$

$$Q_3 = 0{,}2 \times 0{,}1 + 0{,}47 \times 0{,}2 + 0{,}32 \times 0{,}1 = 0{,}14.$$

Виходячи з отриманих результатів, можна зробити висновок, що найкращим є PWA додаток. Користування додатком було найбільш зручним та комфортним. Тобто можна зробити висновки що, при використанні сервісу для планування важливими є такі характеристики:

– усі функції мають бути інтуїтивно зрозумілі кожному користувачу. Важливо, щоб людина миттєво знаходила потрібні кнопки, легко орієнтувалася у функціоналі і відразу бачила всі доступні можливості;

– швидкість встановлення програми, необхідно щоб користувач здійснював якнайменше натискань при установці сервісу;

– робота такого виду сервісу вкрай важлива в офлайн режимі важлива, швидка синхронізація з діями користувача та оновлення.

Отже, функціонал програми розробляється як для зручності користувача, а й у вигоди бізнесу. В ідеалі, натискання кнопки має по ланцюжку вести користувача до цільової дії. Додаток повинен містити рівно стільки функцій, скільки необхідно клієнту на шляху до цільової дії.

Навіть самий «наворочений» функціонал виявиться марним, якщо не буде якісно продуманий інтерфейс користувача. Дуже важливо давати розуміти користувачу його наступні кроки коли він використовує сервіс. Виявлено, що можливість використання додатку в офлайн режимі є необхідність.

6 ЕКОНОМІЧНА ЧАСТИНА

6.1 Характеристика науково-дослідної роботи

Метою даного розділу є економічне обґрунтування витрат на проведення науково-дослідної роботи (НДР), в межах якої передбачається дослідження технології PWA для створення веб-додатку “Планер”. Під час такого обґрунтування буде здійснено: розрахунок трудовитрат та заробітної плати працівникам, розрахунок одноразових витрат і прибутку, оцінку результатів НДР. Реалізація НДР передбачає такі етапи:

– аналіз предметної області;
– визначення алгоритму реалізації проекту;
– дослідження технології PWA;
– вибір методів для проведення експерименту;
– порівняння технологій PWA, нативного веб-додатку та веб-сайту;
– доведення дійсності технологій PWA.

6.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

Під час виконання науково-дослідної роботи був проведений огляд технології PWA, на основі аналізу спеціальної літератури розглянуті, дослідження існуючих архітектур розробки мобільних пристроїв, виконання порівняння функцій нативної та архітектури PWA та архітектуру розробки PWA на основі порівнянь. Умовно науково-дослідну роботу (НДР) можна розділити на такі етапи: підготовчий, основний і заключний.

На стадії виконання підготовчого етапу здійснено підбір критеріїв оцінки якості існуючих веб-додатків. Проаналізовано цільову аудиторію.

На етапі виконання основної частини НДР були виконані такі роботи:

– розгляд технологій PWA, нативного веб-додатку та веб-сайту;

– аналіз розглянутих технологій та порівняння за критеріями;

– тестування PWA додатку;

– тестування нативного додатку;

– тестування веб-сайту.

У заключній частині здійснюється оцінка ефективності виконання НДР, складання звіту з НДР, захист звіту.

Найбільш складною та відповідальною частиною при плануванні НДР є розрахунок трудомісткості робіт, тому що трудові витрати часто становлять основну частину вартості науково-дослідних робіт і безпосередньо впливають на строки розробки.

Для виконання роботи було залучено 4 особи, контролював процес керівник роботи, тобто робоча загальна чисельність на виконання НДР склала 5 осіб. До складу групи виконавців увійшли:

– керівник – 1 особа, заробітна плата 15000 грн/міс.;

– тестувальник UX – 1 особа, заробітна плата 16000 грн/міс.;

– тестувальник UI – 1 особа, заробітна плата 16000 грн/міс.;

– спеціаліст в сфері обробки аналітичних даних – 2 особи, заробітна плата 20000 грн/міс.

Проведемо розрахунок трудовитрат і заробітної плати виконавців робіт.

Середньоденна заробітна плата виконавця робіт ($З_{ср.дн.}$) розраховується:

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \quad (6.1)$$

де $З_{ср.міс.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n = 22).

Середньоденна заробітна плата керівника складає:

$$Зср.дн. = \frac{15000}{22} = 681{,}8 \text{ (грн)}.$$

Середньоденна заробітна плата тестувальників складає:

$$Зср.дн. = \frac{16000}{22} = 727,2(грн).$$

Середньоденна заробітна плата спеціаліста в сфері обробки аналітичних даних складає:

$$Зср.дн. = \frac{20000}{22} = 909(грн).$$

Етапи виконання НДР, перелік і зміст робіт, трудомісткість їх виконання, заробітна плата виконавців робіт представлені в табл. 6.1.

Таблиця 6.1 – Розрахунок трудовитрат і заробітної плати виконавців робіт

Перелік робіт	Кіль-кість вико-навців	Посада вико-навця	Трудомісткість робіт, люд.-днів	Середньо-денна заро-бітна плата, грн.	Сума заро-бітної плати, грн.
1	2	3	4	5	6
1. Підготовчий етап					
1.1. Розробка та затвердження ТЗ	1	Керівник	2	681,8	1363,6
1.2 Підготовка довідкових матеріалів та даних для виконання НДР	1	Спеціаліст в сфері обробки аналітичних даних	2	909	1818
2. Основний етап					
2.1 Постановка задачі	1	Керівник	1		681,8
2.2 Розгляд технологій PWA, нативного веб-додатку та веб-сайту	1	Спеціаліст в сфері обробки аналітичних даних	4	909	3636
2.3 Аналіз розглянутих технологій та порівняння за критеріями	1	Спеціаліст в сфері обробки аналітичних даних	5	909	4545

Продовження таблиці 6.1

1	2	3	4	5	6
3 Тестування					
3.1 Тестування PWA додатку	1	тестувальник UX, тестувальник UI	2	1454	2908
3.2 Тестування нативного додатку	2	тестувальник UX, тестувальник UI	2	1454	2908
3.3 Тестування веб-сайту					
3.4 Підтвердження результату	1	Спеціаліст в сфері обробки аналітичних даних, керівник	1		1591
4. Заключний етап					
4.1 Аналіз результатів проведення роботи	1	Спеціаліст в сфері обробки аналітичних даних, керівник	2	1591	3182
4.2 Формування висновків та пропозицій за темою дослідження	1	Спеціаліст в сфері обробки аналітичних даних, керівник	1		1591
4.3 Технічне оформлення звіту виконання НДР	1	Керівник	2	681,8	1363,6
Усього			28		22185

Таким чином, сума витрат на заробітну плату в межах виконання НДР складе 22185 грн.

6.3 Розрахунок одноразових витрат на розробку НДР

Калькуляція собівартості розраховується відповідно до існуючих нормативних актів України. До складу калькуляції входять такі статті витрат:

– матеріальні витрати;

– витрати на оплату праці;

– єдиний соціальний внесок;

– амортизація основних засобів (вартість машинного часу);

– витрати на спожиту електроенергію;

– інші витрати.

До інших витрат відносяться адміністративні витрати (водопостачання, водовідведення, опалення, освітлення) та вартість послуг зв'язку.

Матеріальні витрати визначаються витратами на матеріали, визначені їх потребою для виконання робіт, і цін, що діють на момент складання калькуляції. Матеріальні витрати розраховуються за такою формулою:

$$M = \sum_{j=1}^{n} Q_j \times Ц_j, \tag{6.2}$$

де M – сумарні витрати на матеріали, в тому числі малоцінні предмети, що швидко зношуються (носії, папір, канцелярське приладдя тощо), або на літературу, яка необхідна для проведення роботи, тощо;

Q_j – кількість використаних одиниць j-го виду матеріалів, $j = (1 \div n)$;

$Ц_j$ – ціна одиниці j-го виду матеріалів.

Розрахунок матеріальних витрат представлено в табл. 6.2.

Таблиця 6.2 – Розрахунок матеріальних витрат

Найменування	Од. вим.	Кількість, (Q_j)	Ціна ($Ц_j$), грн	Сумарні витрати на матеріали (М), грн
Ручка	шт.	3	10	30
Маркер	шт.	3	15	45
Блокнот	шт.	3	100	300
Усього				375

Витрати на оплату праці розраховуються, виходячи з необхідного для виконання робіт складу й кількості працівників, а також із середньомісячної заробітної плати.

Відповідно до проведених розрахунків витрати на оплату праці виконавців роботи дорівнюють 22185 грн.

Єдиний внесок на загальнодержавне соціальне страхування (ЄСВ) – консолідований страховий внесок, збір якого здійснюється в систему загально обов'язкового державного соціального страхування в обов'язковому порядку і на регулярній основі з метою забезпечення захисту у випадках, передбачених законодавством, прав застрахованих осіб і членів їх сімей на отримання страхових виплат (послуг) за діючими видами загальнообов'язкового державного соціального страхування.

Ставка єдиного соціального внеску складає 22 % від витрат на оплату праці, тобто розмір ЄСВ дорівнює 4880,7 грн.

Під час виконання НДР застосовувалось наступне обладнання: ноутбук вартістю 10000 грн та сматфон для тестування 4700 грн.

Вищенаведене устаткування є власністю організації виконавця, тому доцільно розрахувати суму амортизаційних відрахувань на період виконання НДР. Амортизація основних засобів розраховується за формулою:

$$AB = \sum_{k=1}^{L} \frac{BO_k}{TE_k} \times T, \tag{6.3}$$

$$AB = \frac{10000 \times 35}{730} + \frac{4700 \times 35}{730} = 479{,}4 + 225{,}3 = 704{,}74 \text{ (грн).}$$

де AB – сума амортизаційних відрахувань, нарахованих під час проведення науково-дослідної роботи;

BO_k – вартість основних засобів k-го виду;

TE_k – термін експлуатації основних засобів k-го виду, днів;

T – термін науково-дослідницької роботи, днів;

L – кількість видів обладнання.

Витрати на використану обладнанням електроенергію ($В_е$):

$$B_e = M \cdot t \cdot T_{кВт}, \quad (5.4)$$

де M – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$T_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

Споживна потужність комп'ютера складає 0,5 кВт за годину. Тариф споживачів за першим класом напруги, тобто 35 кВт та більше), складає 1,99 грн./кВт.годин (без ПДВ). Підставивши значення у (6.4), визначимо величину витрат ($В_е$) на спожиту електроенергію:

$$B_e = 0{,}5 * 280 * 1{,}99 = 278{,}6 \text{ грн.}$$

До інших статей витрат відносяться такі:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20% від витрат на оплату праці;

– вартість оплати послуг зв'язку.

Вартість оплати послуг зв'язку становитиме:

а) Інтернет – 300 грн. на місяць (безлімітний пакет); всього 300 грн. за 22 дні виконання НДР;

б) телефон – 200 грн на місяць; усього 200 грн. за 22 дні.

За період виконання НДР витрати на відрядження, аутсорсинг, інформаційні послуги та маркетингові заходи не мали місця.

Для виконання НДР використовувалося безкоштовне програмне забезпечення.

Результати розрахунку кошторису витрат, тобто одноразових витрат, на виконання НДР, наведені в табл. 6.3.

Таким чином, кошторис витрат на виконання даної НДР відбиває сумарні витрати за статтями і складає 27497 грн.

Таблиця 6.3 − Кошторис витрат на розробку НДР

№ з/п	Стаття витрат	Сума, грн.
1	Заробітна плата	22185
2	Єдиний соціальний внесок (22,0 % від п.1)	4880,7
3	Матеріальні витрати	375
4	Амортизація основних засобів	704,74
5	Витрати на спожиту електроенергію	278,6
6	Інші витрати, у тому числі:	-
6.1	Адміністративні витрати (20,0 % від п.1)	4437
6.2	Вартість послуг зв'язку	500
7	Усього витрати	27497

6.4 Оцінка результатів науково-дослідної роботи

Результат – це наслідок послідовності дій, виконаних під час НДР, виражений якісно або кількісно. В загальному випадку оцінка результатів НДР – це визначення ефективності отриманих рішень порівняно з сучасним науково-технічним рівнем.

Відповідно до теми даного дослідження у якості результату впровадження НДР визначено зменшення часу на досягнення мети користувача (завантаження сервісу, написання та збереження to do листа)

Результат від впровадження НДР визначається за формулою:

$$\Delta P_j = \left| Хб_j - Хн_j \right|, \tag{6.5}$$

де ΔP_j – покращення j-ої характеристики досліджуваного процесу за рахунок впровадження результатів НДР ($j=1,m$);

m – кількість досліджуваних характеристик;

$Хб_j$ – базове значення j-ої характеристики;

$Хн_j$ – нове значення j-ої характеристики після впровадження НДР.

У якості досліджуваної характеристики обрано час, використаний на досягнення мети користувачем (завантаження сервісу, написання та

збереження to do листа). Отримання часу досягнення мети проводилося за допомогою веб-сайту, нативного додатку та PWA додатку. Отримані результати тестування наведені у таблиці 6.4.

Таблиця 6.4 – Час досягнення мети користувачем

Показник	Веб сайт	Нативний додаток	PWA додаток
Час виконання завдання (хв.)	4,2	3,7	3,4

Підставивши відповідні значення до формули (6.5), визначимо результат від впровадження НДР у чисельному вигляді:

$$\Delta P_1 = |4{,}2 - 3{,}4| = 0{,}8 \text{ (хв)}.$$

PWA додаток за результатами тестування виявився більш зручним та зрозумілим у користуванні, що дозволило скоротити час досягнення мети користувачем на 0,8 хвилини.

Далі проведено оцінку економічної ефективності отриманого результату виконаної науково-дослідної роботи.

6.5 Визначення економічної ефективності результатів НДР

Для визначення економічної ефективності результатів НДР необхідно порівняти витрати на розробку НДР з отриманими результатами.

Основним показником економічної ефективності науково-дослідної роботи є коефіцієнт «ефект-витрати», який розраховується за формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \tag{6.6}$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня витрат НДР змінює *j*-ту характеристику досліджуваного процесу.

Підставивши раніше визначені значення до (6.6), розрахуємо чисельне значення коефіцієнту «ефект-витрати»:

$$K_{ев} = \frac{0{,}8}{27497} \times 100\% = 0{,}0029(\%).$$

Таким чином, отриманий результат свідчить про те, що завдяки результату від впровадження НДР веб-додатки, оптимізовані розробленою методикою PWA швидше завантажуються та дозволяють зробити необхідну дію в офлайн режимі, що приваблює користувачів. Роботу в цілому можна враховувати ефективною або такою, що має високий науковий та технічний рівень.

ВИСНОВКИ

Отже, вважаючи на зростаючу популярність прогресивних веб-додатків зараз ідеальний час для вивчення відмінностей між PWA та іншими підходами до створення мобільного користувацького досвіду. Для проведення цього дослідження було зібрано список найважливіших критеріїв мобільних додатків.

PWA наздоганяє функції, які включені за замовчуванням у нативних додатках, створюючи нові стандарти та впроваджуючи їх у браузерах. PWA може замінити нативний додаток, якщо ключові функції додатка можуть бути розроблені за допомогою поточних можливостей PWA. Оскільки прогресивні веб-додатки не забезпечують просунутий графічний рендеринг, а загальна складність розробки є низькою, вони можуть бути ідеальним вибором для певних сценаріїв.

В принципі, підходи для PWA та гібридних додатків дуже схожі, однак, все ще існують деякі важливі відмінності, коли мова йде про їхню архітектуру. Гібридні додатки можуть використовувати апаратні/програмні можливості пристроїв за допомогою плагінів, що надає переваги за кількістю функцій, які вони можуть містити, та обсягом доступних обчислювальних ресурсів.

Як і у випадку з нативними додатками, PWA може бути використаний для заміни гібридного підходу, якщо додаток не залежить від розширеного рендерингу і знаходиться в межах існуючих можливостей PWA. Крім того, важливою перевагою використання підходу PWA є те, що він знижує показник CPA (Cost Per Acquisition), оскільки це веб-додаток і його можна підтримувати за допомогою SEO.

В даній роботі проведено аналіз літератури за темою дослідження та виявлено що, масштабних досліджень на тему прогресивних веб-додатків ще не проводилося.

З певних причин визначення прогресивного веб-додатку не є зрозумілим для багатьох. Визначення варіюються від суворих технічних визначень до тих, що описують нетехнічні аспекти, такі як користувацький досвід. Один з основних висновків це те що PWA можна розглядати як збірник хороших практик, яких слід дотримуватися при створенні веб-додатків. Це найбільш помітно впливає на мобільних користувачів з точки зору продуктивності та доступних функцій. Оскільки PWA це хороший стандарт, якого слід дотримуватися при створенні веб-додатків. Він не вимагає багато роботи і користувач може побачити помітне покращення.

В ході виконання завдання на практику сформовано мету, актуальність та задачі дослідження. Також сформовано гіпотезу дослідження.

Виходячи з результатів дослідження, можна зробити висновки, що під час використання такого додатка як планер найбільш зручно використовувати додаток PWA, оскільки значущими характеристиками є:

– усі функції мають бути інтуїтивно зрозумілі кожному за користувача. Важливо, щоб людина миттєво знаходила потрібні кнопки, легко орієнтувалася у функціоналі і відразу бачила всі доступні можливості;

– швидкість встановлення програми, необхідно щоб користувач здійснював якнайменше натискань при установці сервісу;

– робота такого виду сервісу вкрай важлива в офлайн режимі важлива, швидка синхронізація з діями користувача та оновлення.

Виходячи з вищевикладеного, ми ретельно проаналізували PWA з точки зору їх можливостей в якості об'єднуючої технології для розробки мобільних додатків. На даний момент важко стверджувати, що PWA вже задовольняють багатьом вимогам уніфікованої багатоплатформної розробки. Тим більше, що незабаром, швидше за все, з'являться й інші варті уваги технології.

Однак масовий інтерес з боку практиків спонукає до подальших досліджень. Принаймні, PWA може сприяти отриманню більш багатого досвіду розробки, і – в кінцевому підсумку – створенню кращих додатків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Такур П. Evaluation and Implementation of Progressive Web Application: бакалаврська робота. Гельсінський університет прикладних наук, 2018.

2. Progressive web apps (PWAs). developer.mozilla.org. URL: https:// developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (дата звернення: 04.10.2022).

3. Франкстон Б. Прогресивні веб-додатки. IEEE Consumer Electronics Magazine. 2018. С. 106.

4. Фортунато Д., Бернардіно Х. Прогресивні веб-додатки: Альтернатива нативним мобільним додаткам. 2018 р.

5. Бржоушек П. Оцінка та використання технології Google Progressive Web Apps : бакалаврська робота. Брно. 2017.

6. Франссон Р. Comparing Progressive Web Applications with Native Android Applications : магістерська робота. Vaxjo. 2017.

7. What is PWA? Progressive Web Apps in eCommerce Explained. URL: https://vuestorefront.io/pwa (дата звернення: 09.10.2022).

8. Samsung. PWA Summit. URL: https://developer.samsung.com/internet/ events/en-us/2021/10/06/pwa-summit (дата звернення: 20.10.2022).

9. Progressive Web Apps: Is PWA the Future of Web Design? // brainhub. URL: https://brainhub.eu/library/is-pwa-the-future (дата звернення: 20.10.2022).

10. Why Progressive Web Apps Are The Future of Mobile Web. URL: https://medium.com/ymedialabs-innovation/why-progressive-web-apps-are-the-future-of-mobile-web-5195f381d856 (дата звернення: 21.10.2022).

11. The current state of progressive web apps // stardust-testing. URL: https://www2.stardust-testing.com/en/the-current-state-of-progressive-web-apps (дата звернення: 22.10.2022).

12. Biørn-Hansen A., Grønli T.-M., Majchrzak T.A. Progressive Web Apps: The possible web-native unifier for mobile development // Міжнародна конференція з веб-інформаційних систем та технологій. 2017.

13. Систематичні огляди літератури для освіти. URL: https:// libraryguides. griffith.edu.au/systematic-literature-reviewsfor-education (дата звернення: 22.10.2022).

14. Такур П. Оцінка та впровадження прогресивного веб-додатку: бакалаврська робота. Гельсінський університет прикладних наук. 2018.

15. Вальстрем М. Дослідження прогресивних веб-додатків для охорони здоров'я: магістерська робота. Університет Умео. 2020.

16. Штайнер Т. Аналіз особливостей прогресивних веб-додатків, коли засобом веб-доступу не є веб-браузер. 2021.

17. Франкстон Б. Прогресивні веб-додатки // IEEE Consumer Electronics Magazine. 2022.

18. Франссон Р. Дріагін О. Порівняння прогресивних веб-додатків з нативними додатками Android: бакалаврська робота. Університет Ліннея. 2017.

19. Теплий Я.Б.К. Гібридний мобільний додаток для проекту. Чеський технічний університет. Прага. 2016.

20. Фортунато Д. Бернардіно Х. Progressive web apps: An alternative to the native mobile Apps // Conference on Information Systems and Technologies (CISTI):13th Iberian Conference on IEEE. 2010.