Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет

Механічний факультет

Кафедра комп'ютерних наук і інформаційних систем

ДИПЛОМНА РОБОТА

магістра

РОЗРОБЛЕННЯ ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО СТАНУ АВТОМОБІЛЯ

Завідувачка кафедри канд. техн. наук, доцентка	Ганна ПЛЄХОВА
Нормоконтролер, доктор філософії	Андрій ЛЕБЕДИНСЬКИЙ
Керівник канд. техн. наук, доцент	Олена ШАПОШНІКОВА
Студент гр. МК-61-23	Володимир БОНДАР

Харків – 2024

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

Факультет механічний
Кафедра комп'ютерних наук і інформаційних систем
Освітній рівень другий (магістр)
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувачка кафедри
_____ Г. А. Плєхова
«___» _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бондарю Володимиру Володимировичу

1. Тема роботи Розроблення інтерфейсу мобільного застосунку відстеження технічного стану автомобіля
Керівник роботи Шапошнікова Олена Павлівна, кандидат технічних наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені рішенням Вченої ради механічного факультету «10» жовтня 2024 року, протокол № 136.
2. Строк подання студентом проекту (роботи) «9» грудня 2024 року

3. Вихідні дані до роботи:
3.1 Мета – аналіз особливостей проектування та практичного прототипування мобільних застосунків.
3.2 Об'єкт дослідження – інтерфейс мобільних застосунків.
3.3 Предмет дослідження – інформаційні технології для реалізації інтерфейсу мобільного застосунку.
3.4 Завдання та інструментальні засоби їх виконання:
3.4.1 Основні задачі: дослідити та розробити макет мобільного застосунку
3.4.2 Інструментальні засоби: Figma, Visual Studio Code.
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити):
4.1 Аналітичний огляд.
4.2 Проектування інтерфейсу мобільного застосунку.
4.3 Розробка інтерфейсу мобільного застосунку.
5. Перелік демонстраційного матеріалу (з точним зазначенням обов'язкових слайдів):
1) постановка задачі; 2) визначення цінності пропозиції; 3) аналіз інтерфейсів існуючих мобільних застосунків; 4) структура мобільного застосунку; 5) розробка вайрфеймів; 6) шляхи користувача інформаційної системи; 7) прототип інтерфейсу мобільного застосунку; 8) висновки

6. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 10 жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналітичний огляд	10.10.2024 – 11.10.2024	
2	Проектування інтерфейсу мобільного застосунку	12.10.2024 – 29.10.2024	
3	Розробка інтерфейсу мобільного застосунку	30.10.2024 – 30.11.2024	
4	Формування висновків по роботі. Оформлення дипломної роботи	01.12.2024 – 05.12.2024	
5	Підготовка презентації і доповіді	06.12.2024 – 09.12.2024	

Студент _____ Бондар В. В.

Керівник роботи _____ Шапошнікова О. П.

РЕФЕРАТ

Дипломна робота складається з 102 сторінок, містить 27 рисунків, 2 таблиці та 17 посилань.

МОБІЛЬНИЙ ЗАСТОСУНОК, ДИЗАЙН ІНТЕРФЕЙСУ СИСТЕМИ, UI UX DESIGN, WIREFRAME, PROTOTYPE, ШАБЛОН.

Мета роботи – аналіз особливостей проектування та прототипування інтерфейсу мобільного застосунку для відстеження технічного стану автомобіля.

Об'єкт дослідження – інтерфейс мобільних застосунків для моніторингу транспортних засобів.

Предмет дослідження – інформаційні технології для реалізації інтерфейсу мобільного застосунку, призначеного для контролю технічного стану автомобіля.

Бажані результати - розробити інтерфейс мобільного застосунку, який дозволить користувачам відстежувати технічний стан автомобіля. Застосунок наддасть можливість реєстрації витрат на паливо, щоденно пройдених кілометрів, доданого пального, дат обслуговування та вартості запчастин.

За підсумками дослідження автором опубліковано статтю у збірнику матеріалів міжнародної науково-практичної конференції, що підтверджує наукову цінність та практичну значущість виконаної роботи.

ЗМІСТ

ВСТУП

У сучасному світі користувачі прагнуть отримати найкращий досвід від цифрових продуктів, мінімізуючи витрати часу та зусиль. Однак привабливий дизайн та цікавий контент самі по собі не забезпечують успіху продукту, будь то мобільний застосунок, веб-сайт чи інша цифрова платформа. Основною метою розробки будь-якого продукту є створення рішення, яке буде корисним і зручним у використанні.

У зв'язку з різноманіттям доступних опцій, успіх продукту значною мірою залежить від належного проектування інтерфейсу користувача з урахуванням принципів UI/UX. Ці методи сприяють кращому розумінню потреб цільової аудиторії через проведення досліджень, підвищують задоволеність користувачів завдяки вдосконаленню зручності використання, знижують витрати на залучення нових клієнтів та сприяють підвищенню рівня утримання вже наявних.

Аналіз результатів подібних досліджень дозволяє виявити завдання та потреби потенційних користувачів, що сприяє покращенню не тільки інтерфейсу, а й функціональності інформаційної системи. Наприклад, якщо дослідження вказують на необхідність впровадження нової функції, розробники можуть реалізувати її, забезпечуючи повніше задоволення потреб користувачів.

Процес проектування інтерфейсу мобільного застосунку для відстеження технічного стану автомобіля повинен здійснюватися поетапно. Це включає збір вимог, розробку концепції, проектування інтерфейсу, розробку і тестування, щоб забезпечити оптимальний досвід користувача.

Загалом, успішна комбінація привабливого дизайну, зручної навігації та функціональності, підкріплена дослідженнями та аналізом потреб користувачів, є ключовим фактором у створенні ефективного мобільного застосунку для моніторингу автомобіля.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі моніторингу технічного стану автомобілів

Моніторинг технічного стану автомобілів — це процес постійного відстеження різних параметрів транспортного засобу з метою виявлення можливих несправностей або необхідності технічного обслуговування. З розвитком технологій такі системи стають все більш доступними для звичайних автомобілістів через мобільні додатки, що робить їх важливими як для власників автомобілів, так і для бізнесів у сфері автомобільного обслуговування.

Потреба ринку:

Зниження витрат на обслуговування та ремонт: Багато автомобілістів не мають можливості або часу регулярно звертатися до сервісних центрів для перевірки стану авто. Відстеження основних показників технічного стану автомобіля в режимі реального часу дозволяє виявляти можливі проблеми на ранніх етапах і уникати дорогих поломок. Це економить як час, так і гроші.

Покращення безпеки на дорогах: Технічні несправності є однією з основних причин ДТП. Мобільні додатки, що моніторять стан авто, можуть попереджати про проблеми, які потребують негайної уваги, наприклад, про знос гальмівних колодок або зниження рівня масла. Це дозволяє власникам автомобілів підтримувати свій транспортний засіб у належному стані та знижувати ризик аварійних ситуацій.

Простота управління автомобілем: Багато водіїв мають кілька транспортних засобів або просто не пам'ятають про всі деталі, пов'язані з техобслуговуванням. Мобільні додатки спрощують процес управління технічним обслуговуванням, нагадуючи про необхідність змінити масло, пройти техогляд або замінити деталі, що зношуються.

Екологічний аспект: Технічні проблеми автомобіля, такі як поганий стан двигуна або вихлопної системи, можуть спричиняти підвищений викид шкідливих речовин в атмосферу. Завдяки вчасному моніторингу та обслуговуванню автомобілі можуть працювати ефективніше, споживаючи менше пального і виділяючи менше забруднюючих речовин.

Кому це потрібно?

Приватні водії: Вони шукають способи знизити витрати на обслуговування автомобіля, підвищити безпеку та покращити його надійність. Мобільний застосунок для моніторингу технічного стану допоможе зручно стежити за потребами автомобіля та вчасно отримувати попередження про необхідність ремонту або сервісу.

Компанії з автопарками (логістичні компанії, служби таксі тощо): Для таких компаній критично важливо, щоб всі автомобілі були в справному стані, адже це впливає на бізнес-процеси, безпеку водіїв і клієнтів, а також репутацію. Система моніторингу допомагає централізовано відстежувати стан автопарку та своєчасно реагувати на виникнення несправностей.

СТО та сервісні центри: Для цих компаній застосунок може стати засобом підвищення лояльності клієнтів. Сервісні центри можуть пропонувати інтеграцію з додатками для моніторингу, що дозволяє отримувати інформацію про технічний стан автомобіля ще до його візиту на СТО.

Задача користувача

Основна задача користувача — це зручний і простий інструмент для моніторингу технічного стану автомобіля в режимі реального часу. Водії хочуть уникнути непередбачених поломок та витрат, мати контроль над витратами на пальне та обслуговування, а також підтримувати безпеку на дорозі. Користувачі очікують, що застосунок буде інтуїтивно зрозумілим, забезпечуватиме своєчасні сповіщення та дозволить зручно керувати інформацією про автомобіль.

Виходячи з цієї потреби ринку та задач користувачів, ми можемо аналізувати існуючі рішення та визначити, які функції та можливості повинні бути в мобільному додатку для моніторингу технічного стану авто.

1.2 Дослідження цільової аудиторії

При розробці користувацького інтерфейсу, дослідження цільової аудиторії є ключовим аспектом, що допомагає зрозуміти реальні потреби, болі та очікування користувачів. Робота виключно на основі власних припущень може призвести до серйозних помилок, оскільки розробники часто перебувають у «пастці експерта». Для них продукт здається інтуїтивно зрозумілим і простим у використанні, адже вони безпосередньо працюють над його створенням. [3] Проте користувачі, особливо нові, можуть не мати такої обізнаності і можуть зіткнутися зі складнощами, які розробник не врахував.

Завдяки глибокому аналізу цільової аудиторії на ранніх стадіях можна значно покращити користувацький досвід і створити інтерфейс, який буде зрозумілим та зручним для кінцевого споживача. Важливо не тільки враховувати думки ключової цільової аудиторії, яка найактивніше користуватиметься продуктом, але й звертати увагу на вторинну аудиторію, яка може не використовувати продукт постійно або мати інші вимоги до його функціоналу.

Розподіл користувачів на сегменти допоможе краще зрозуміти їхні завдання, потреби та мотивації. Основна аудиторія — це користувачі, для яких продукт вирішує конкретні проблеми або задовольняє важливі потреби. Дотична або вторинна аудиторія може використовувати продукт випадково або епізодично, що також впливає на дизайн, але їхній досвід буде менш критичним для загального успіху продукту.

На основі цього дослідження можна впровадити інструменти на зразок Value Proposition Canvas (Рис. 1.1). Він дозволяє структурувати процес розуміння цінностей, які користувачі прагнуть отримати, а також проблем, які вони хочуть

вирішити за допомогою продукту. Канва складається з двох основних блоків: профілю клієнта, де аналізуються завдання, болі та вигоди користувачів, та ціннісної пропозиції компанії, яка пояснює, як саме продукт вирішує ці проблеми та приносить користь. [4]

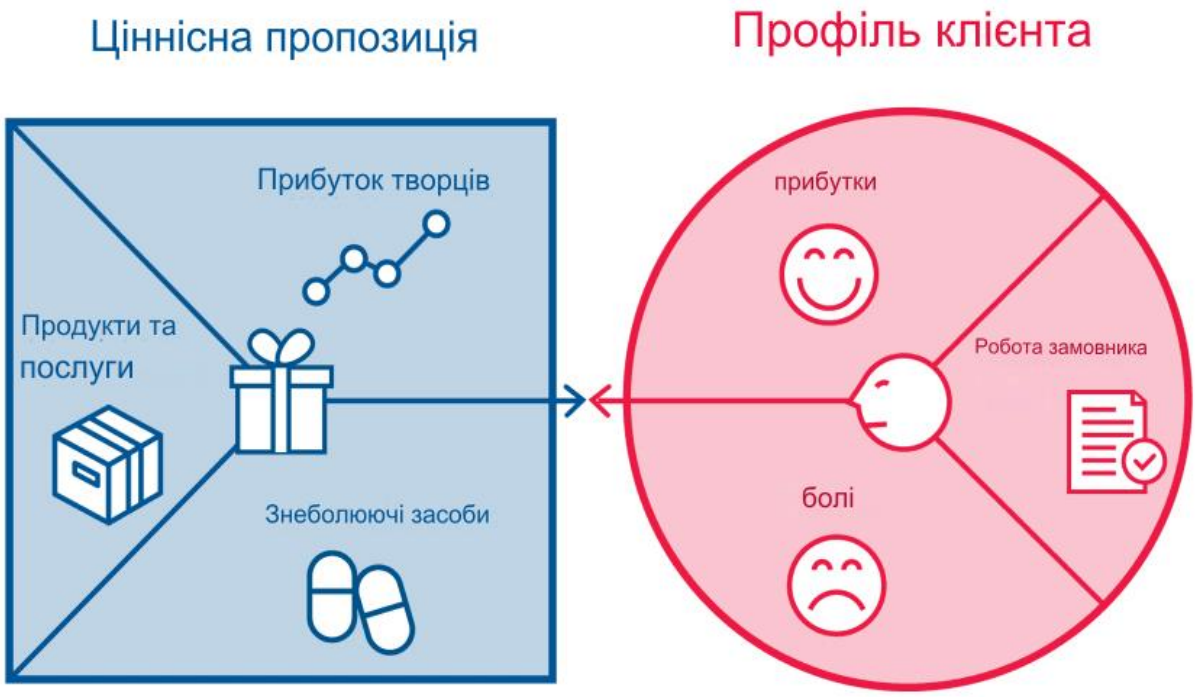


Рисунок 1.1 - Інструмент Value Proposition Canvas

Профіль клієнта включає три ключові елементи:

Завдання — це ті функціональні, соціальні чи емоційні завдання, які користувачі намагаються виконати, проблеми, які вони хочуть вирішити, або потреби, які вони бажають задовольнити.

Болі — це негативні аспекти або труднощі, з якими користувачі стикаються під час виконання завдань, включаючи ризики та проблеми, які виникають під час використання продукту.

Вигоди — це ті результати або відчуття, які користувачі прагнуть отримати, зокрема поліпшення продуктивності, зниження витрат чи просто задоволення.

У свою чергу, Карта цінностей описує, як продукт може допомогти:

Творці вигід — це способи, якими продукт або послуга сприяють отриманню користувачем вигід.

Знеболювальні — це функції, які зменшують або усувають проблеми користувачів.

Продукти та послуги — це конкретні рішення, що допомагають користувачам досягти їхніх цілей.

Після цього важливо знайти «відповідність» між пропозицією цінності та профілем клієнта, щоб переконатися, що продукт максимально вирішує основні проблеми користувачів та приносить найбільшу користь. Тільки після цього можна перейти до тестування та валідації пропозиції через відгуки користувачів, що допоможе вдосконалювати продукт на основі реальних даних.

Постійний процес дослідження та отримання зворотного зв’язку від цільової аудиторії забезпечує гнучкість у розробці продукту. Це дозволяє адаптувати інтерфейс до реальних потреб користувачів, знижувати ризики та підвищувати шанси на успішний запуск продукту на ринку. Постійний діалог із користувачами дозволяє вчасно виявляти проблеми, які потребують вирішення, та ефективно реагувати на нові вимоги або зміни в їхніх очікуваннях.

Для досягнення гармонії між ціннісною пропозицією та профілем клієнта важливо класифікувати всі елементи, такі як продукти, послуги, знеболювальні і творці вигід, від категорії «приємно мати» до «необхідної» з точки зору клієнтської цінності. Відповідність досягається тоді, коли запропоновані рішення спрямовані на задоволення найважливіших потреб клієнтів та розв'язання їх ключових проблем.

Наступним кроком є аналіз конкурентних переваг компанії в цих сферах, щоб переконатися, що пропозиція цінності є не лише привабливою для клієнтів, але й унікальною на ринку. Це допоможе забезпечити стійкість продукту серед конкурентів та закріпити його на ринку як найкраще рішення для певного сегменту аудиторії.

Використання матриці для побудови кожного аспекту ціннісної пропозиції дозволяє наочно оцінити важливість різних компонентів для клієнта та оцінити конкурентну перевагу, яку компанія має, пропонуючи цей продукт або послугу. Ця матриця допомагає структурувати комунікацію та вибрати ті повідомлення, які мають найбільше значення для клієнтів, виділяючи ключові аспекти продукту. Результати застосування інструменту Value Proposition Canvas для аналізу цільової аудиторії для розробки мобільного застосунку моніторингу технічного стану автомобіля наведено зведено у таблицю 1.1.

Таблиця 1.1 - Результати застосування інструменту Value Proposition Canvas

Customer Profile (Профіль клієнта)	**Value Map (Карта ціннісної пропозиції)**
Jobs (Завдання клієнта)	**Products & Services (Продукти та сервіси)**
- Стежити за технічним станом авто.	- Мобільний додаток для моніторингу технічного стану авто.
- Контролювати витрати на пальне та ремонти.	- Системи введення даних про пробіг, витрати на обслуговування та пальне.
- Отримувати нагадування про технічне обслуговування.	- Автоматизовані нагадування про необхідність технічного обслуговування.
- Мати доступ до всієї історії технічного обслуговування авто.	- Аналітика щодо витрат на технічне обслуговування та пальне.
- Можливість роботи в офлайн-режимі.	
Pains (Болі клієнта)	**Pain Relievers (Те, що зменшує біль)**
- Високі витрати на ремонти через несвоєчасне обслуговування.	- Вчасні сповіщення про необхідність обслуговування.
- Забуття про техогляди та заміну витратних матеріалів.	- Централізоване зберігання всіх даних для зручного доступу.
- Ризик несподіваних поломок під час поїздок.	- Виявлення можливих несправностей до серйозних поломок.
- Відсутність єдиного інструменту для моніторингу авто.	- Систематизація всієї інформації в одному місці.
Gains (Вигоди)	**Gain Creators (Те, що створює вигоди)**
- Зниження ризику поломок авто.	- Підвищення ефективності та безпеки автомобіля.
- Вчасне обслуговування для зменшення витрат на ремонти.	- Прогнозування витрат на обслуговування.
- Зручність в управлінні технічними даними.	- Оптимізація витрат на обслуговування та пальне.

Результати аналізу дають змогу визначити основні повідомлення, які мають найбільший потенціал привернути увагу широкої аудиторії. Ці повідомлення повинні бути основними у маркетингових кампаніях і відображатися у всіх рекламних матеріалах. Також аналіз виявляє потенційні напрями для вдосконалення продукту, такі як оптимізація його функціональних можливостей, покращення зручності користування або усунення слабких місць у пропозиції компанії. Крім того, дослідження ринку допоможе з'ясувати, де компанія може виграти за рахунок удосконалень у порівнянні з конкурентами, та підвищити привабливість свого продукту для споживачів.

Таке систематичне та постійне вдосконалення не тільки допомагає підтримувати актуальність продукту, але й створює передумови для довгострокової конкурентоспроможності та стійкого зростання компанії.

Крім того для розробки мобільного застосунку, що дозволить користувачам моніторити технічний стан автомобіля, з урахуванням UI/UX, слід звернути увагу на кілька важливих аспектів.

По-перше, застосунок повинен мати простий та інтуїтивний інтерфейс, що полегшить його використання навіть для тих, хто не має великого досвіду у користуванні подібними системами. Використання графічних елементів і зручних іконок сприятиме швидкому доступу до функцій застосунку.

По-друге, застосунок має включати широку базу даних з функціями обліку витрат на паливо, пробіг, технічне обслуговування та ремонт.

Користувачі повинні мати можливість легко знаходити інформацію про свої витрати, отримувати нагадування про майбутнє обслуговування та мати доступ до історії витрат.

Це дозволить створити продукт, який відповідатиме сучасним вимогам і забезпечить користувачів усіма необхідними інструментами для зручного моніторингу свого автомобіля.

1.3 Обґрунтування важливості UI та UX дизайну

UI-дизайн і UX-дизайн - взаємозалежні елементи в процесі проектування користувацького досвіду. Ці два аспекти працюють разом, щоб створити інтерфейс, який не тільки привабливий з погляду візуального дизайну, а й забезпечує зручний, інтуїтивний і ефективний користувацький досвід (Рис.1.2). Це ключовий елемент успіху будь-якого цифрового продукту, включно з мобільними застосунками. [1-2]

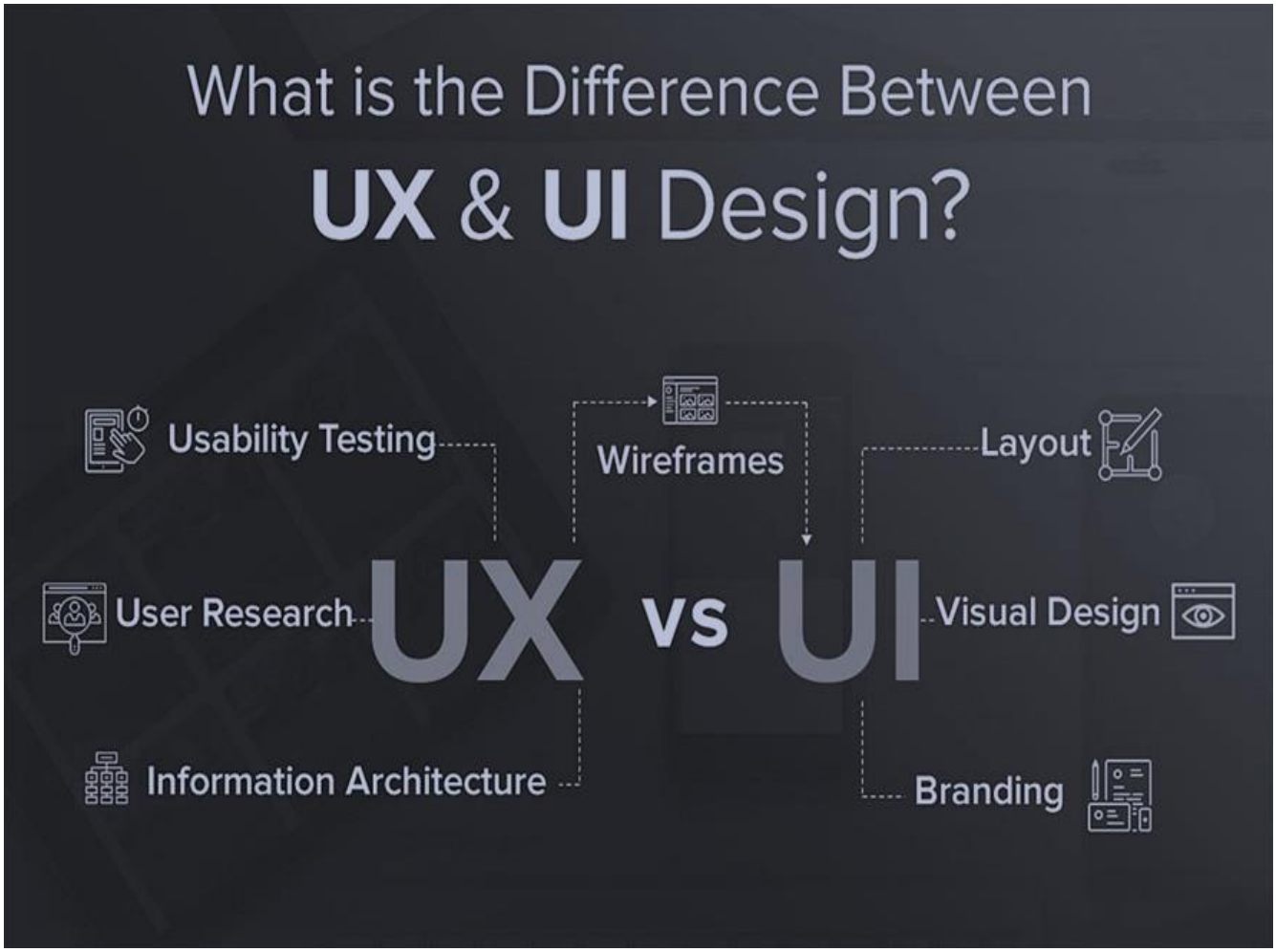


Рисунок 1.2 - UI-дизайн і UX-дизайн

UI-дизайн (призначений для користувача інтерфейс) фокусується на візуальних та інтерактивних елементах інтерфейсу застосунку. Його основне

завдання - створити привабливий інтерфейс, що дає змогу користувачам легко орієнтуватися в застосунку і взаємодіяти з його функціоналом. Дизайнери призначеного для користувача інтерфейсу розробляють макети екранів, визначають форму і розмір кнопок, обирають колірні схеми і типографіку, створюють анімацію і візуальні ефекти. Успішний дизайн користувацького інтерфейсу покликаний полегшити користувачам розуміння роботи застосунку та забезпечити приємні візуальні відчуття.

У контексті застосунку для моніторингу стану автомобіля дизайн користувацького інтерфейсу має охоплювати такі аспекти, як вибір графічних елементів для відображення стану різних компонентів автомобіля (наприклад, піктограми паливної системи, двигуна, гальм тощо), зрозумілі та доступні кнопки для введення даних про технічне обслуговування та витрати. Такі аспекти, як створення легкодоступних кнопок для введення даних про технічне обслуговування і витрати. Крім того, дизайн призначеного для користувача інтерфейсу має враховувати потребу користувача у швидкому доступі до необхідної інформації, наприклад, за допомогою приладових панелей та інтерактивних графіків, які наочно відстежують технічний стан автомобіля.

UX-дизайн (користувацький досвід) фокусується на тому, як користувачі взаємодіють із застосунком і яке враження вони отримують від цієї взаємодії. Він стосується структури застосунку, логіки роботи та загальної простоти використання: основне завдання UX-дизайнера - створити інтуїтивно зрозумілу та логічну навігацію, яка дасть змогу користувачеві швидко знаходити потрібні функції та виконувати завдання без зайвих зусиль.

У випадку з мобільним додатком для моніторингу технічного обслуговування автомобілів UX-дизайн визначає, як користувачі взаємодіють із функціями додатка, такими як реєстрація даних про обслуговування, відстеження витрат на пальне, отримання попереджень про обслуговування та перегляд історії витрат UX-дизайнери слідкують за тим, щоб усі ці процеси були щоб усі ці процеси були максимально простими та зручними для користувачів. Наприклад,

введення даних про пробіг і витрати має бути інтуїтивно зрозумілим, а система сповіщень має своєчасно інформувати користувачів про необхідність технічного обслуговування.

Взаємодія між UI- та UX-дизайн нероздільні для успішних цифрових продуктів Хоча UI і UX зосереджені на різних аспектах дизайну, вони повинні працювати разом, щоб забезпечити приємний користувацький досвід UI-дизайн формує естетичну складову застосунку, тоді як UX-дизайн гарантує, що користувачі отримають максимальну віддачу від продукту, не відчуваючи дискомфорту або труднощів.

Наприклад, якщо мобільний застосунок для моніторингу технічного стану автомобіля має дуже привабливий візуальний дизайн, але складний у використанні, користувачі не зможуть ефективно використовувати його функції, що призведе до зниження задоволеності й утримання. З іншого боку, навіть простий і логічний інтерфейс може знизити залученість користувачів, якщо він не відповідає сучасним візуальним стандартам або є непривабливим.

Для успішної розробки мобільних застосунків для моніторингу технічного стану автомобіля важливо забезпечити гармонійне поєднання UI і UX дизайну. Це означає, що інтерфейс має бути візуально привабливим, простим у використанні та інтуїтивно зрозумілим для кінцевого користувача.

Вплив UI/UX на користувацький досвід

Користувацький досвід багато в чому залежить від хорошого дизайну UI і UX. У той час як дизайн користувацького інтерфейсу спонукає користувачів взаємодіяти з додатком за допомогою візуальних елементів, таких як кольори, форми і анімація, дизайн UX забезпечує логічність і зручність цієї взаємодії.

У випадку з мобільними застосунками для моніторингу автомобілів добре продуманий UX дає змогу користувачам швидко знаходити інформацію про технічний стан свого автомобіля, отримувати важливі сповіщення та легко додавати нові дані; UI забезпечує приємний візуальний досвід, який стимулює повторне використання додатка; а UX-дизайн гарантує, що користувачі можуть

легко знаходити інформацію про технічний стан свого автомобіля, отримувати важливі сповіщення та додавати нові дані.

Таким чином, успішний цифровий продукт - це продукт, який поєднує в собі естетичну привабливість дизайну призначеного для користувача інтерфейсу з функціональністю та логікою дизайну UX, даючи змогу користувачам відчувати задоволення від роботи з додатком.

1.4 Аналіз існуючих рішень інтерфейсів мобільних застосунків

Для того, щоб проєктувати інтерфейс застосунку, було розглянуто відомі продукти, що допомагають власникам автомобілів контролювати стан транспортного засобу, витрати на його обслуговування та інші показники.

Перший застосунок — це "My Car". Він доступний для Android і надає спектр функцій для водіїв, що допомагають у моніторингу технічного стану автомобіля та контролі витрат (Рис. 1.3).

"My Car" дозволяє користувачам зберігати інформацію про заправки, пробіг, витрати на технічне обслуговування та ремонти автомобіля [5]. Він також містить функцію нагадувань про необхідні роботи, що допомагає уникати затримок з проведенням планових техобслуговувань. Це дозволяє забезпечувати справний стан транспортного засобу.

Крім того, "MyCar" має функцію запису витрат, що допомагає аналізувати загальні витрати на автомобіль за різні періоди часу. Застосунок також дозволяє створювати звіти на основі введених даних, що може бути корисно для фінансового планування та контролю.

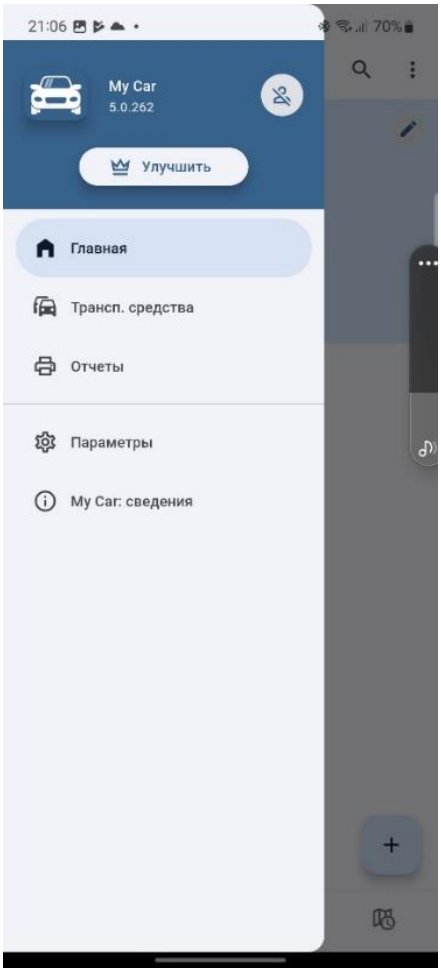


Рисунок 1.3 — Інтерфейс "MyCar"

Опис інтерфейсу:

Головний екран: Після запуску застосунку "MyCar" користувач потрапляє на головний екран, де відображаються основні функції і показники автомобіля. Тут є інформація про загальний пробіг, залишок пального (якщо були введені дані), а також нагадування про майбутні техобслуговування. Користувач також може швидко перейти до функцій введення нових записів про заправки, техобслуговування чи перегляду історії.

Журнал витрат: На головному екрані користувач бачить список останніх витрат. Можна додати нові записи про заправку, ремонт або обслуговування. Ці записи містять детальну інформацію, таку як дата, вид витрати, сума та пробіг автомобіля на момент події.

Контроль технічного стану: Застосунок надає можливість створювати нагадування про майбутні техобслуговування, базуючись на пробігу автомобіля або часових інтервалах. Наприклад, після заміни масла або гальмівних колодок користувач може ввести дату наступної заміни відповідно до рекомендацій виробника.

Статистика та звіти: У розділі статистики користувач може переглянути діаграми та звіти, що показують витрати на автомобіль у різних категоріях, наприклад, заправки, ремонти або техобслуговування. Це дозволяє аналізувати динаміку витрат та планувати бюджет на майбутнє.

Налаштування: У налаштуваннях можна обрати мову застосунку, змінити одиниці виміру (кілометри чи милі), налаштувати сповіщення про необхідність проведення технічних робіт або додати декілька автомобілів для моніторингу.

Інтерфейс "MyCar" розроблений таким чином, щоб бути зручним для користувачів. Він має простий та інтуїтивно зрозумілий дизайн, що дозволяє швидко вводити необхідні дані та отримувати доступ до важливої інформації про стан автомобіля.

Другий застосунок це "Fuelio". Він доступний для Android та iOS і надає широкий спектр функцій для водіїв, що допомагають у відстеженні витрат на паливо, контролі технічного стану автомобіля та інших важливих аспектів (Рис. 1.4).

"Fuelio" дозволяє користувачам зберігати інформацію про заправки, пробіг, витрати на технічне обслуговування та ремонти автомобіля. Він також має функцію нагадувань про необхідні технічні роботи, що допомагає уникати затримок із проведенням планових обслуговувань. Це дозволяє забезпечувати справний стан транспортного засобу. [6-7]

Крім того, "Fuelio" має функцію ведення обліку витрат на паливо та технічне обслуговування, що допомагає аналізувати загальні витрати на автомобіль за різні періоди часу. Застосунок також дозволяє створювати звіти на

основі введених даних, що може бути корисно для фінансового планування та контролю.

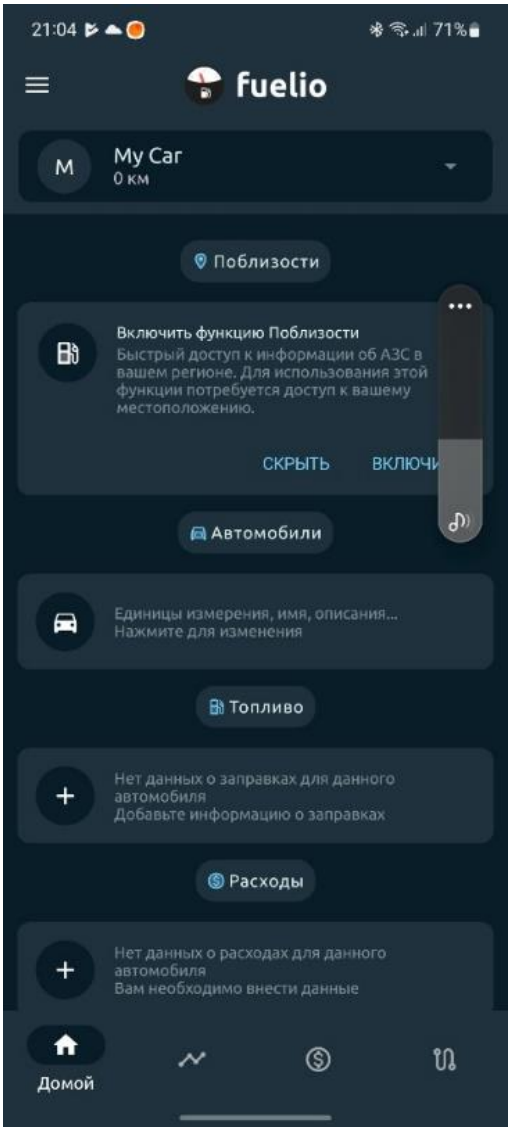


Рисунок 1.4 — Інтерфейс "Fuelio"

Опис інтерфейсу:

Головний екран: Після запуску застосунку "Fuelio" користувач потрапляє на головний екран, де відображаються основні функції та показники автомобіля. Тут є інформація про загальний пробіг, середню витрату пального, останню заправку та витрати на паливо. Користувач також може швидко перейти до функцій введення нових записів про заправки чи перегляду історії витрат.

Журнал витрат: На головному екрані користувач бачить список останніх витрат на заправки. Можна додати нові записи про заправку, ремонт або обслуговування автомобіля. Ці записи містять детальну інформацію, таку як дата, кількість палива, сума витрат і пробіг на момент події.

Контроль технічного стану: "Fuelio" надає можливість створювати нагадування про майбутні техобслуговування, які базуються на пробігу автомобіля або часових інтервалах. Наприклад, після заміни масла чи гальмівних колодок користувач може ввести дату наступного технічного обслуговування відповідно до рекомендацій виробника.

Статистика та звіти: У розділі статистики користувач може переглянути діаграми та звіти, що показують витрати на автомобіль у різних категоріях, таких як заправки, ремонти або технічне обслуговування. Це дозволяє аналізувати динаміку витрат та планувати бюджет на майбутнє.

Налаштування: У налаштуваннях можна обрати мову застосунку, змінити одиниці виміру (кілометри чи милі), налаштувати сповіщення про необхідність технічного обслуговування або додати декілька автомобілів для відстеження.

Інтерфейс "Fuelio" розроблений таким чином, щоб бути зручним для користувачів. Він має простий та інтуїтивно зрозумілий дизайн, що дозволяє швидко вводити необхідні дані та отримувати доступ до важливої інформації про стан автомобіля.

Третій застосунок "Car Expenses" — це мобільний застосунок, розроблений для Android, який допомагає автомобілістам стежити за витратами на обслуговування та експлуатацію їхніх транспортних засобів (Рис. 1.5). Цей застосунок пропонує інструменти для зручного управління фінансовими та технічними аспектами автомобіля.

«Car Expenses» дозволяє користувачам фіксувати витрати на заправку, обслуговування, ремонти та інші пов’язані витрати. Застосунок пропонує функцію аналізу даних, що допомагає оцінювати загальні витрати на транспортний засіб за різні періоди часу, що є корисним для планування бюджету.

Користувачі також отримують можливість створювати нагадування про важливі дати, наприклад, заміну масла чи проходження техогляду, що допомагає підтримувати автомобіль у справному стані.

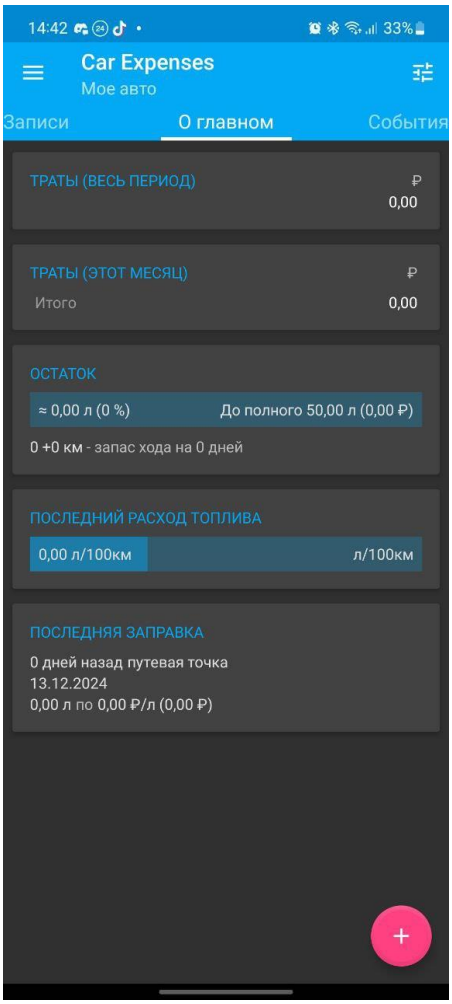


Рисунок 1.5 — Інтерфейс "Car Expenses"

Опис інтерфейсу:

Головний екран: Після відкриття "Car Expenses" користувач потрапляє на головний екран, де відображаються основні фінансові показники автомобіля. Тут можна знайти інформацію про останні записи витрат, зокрема суми заправок, ремонти та техобслуговування. На головному екрані також є швидкий доступ до функцій додавання нових витрат або перегляду детальної статистики.

Журнал витрат: У цьому розділі користувачі можуть переглядати список своїх витрат, відсортований за датою. Кожен запис містить деталі про дату, категорію витрат (заправка, ремонт тощо), суму та інші параметри. Також доступна функція редагування або видалення записів для точного обліку.

Нагадування: Користувачі можуть створювати нагадування про майбутні події, такі як планові техобслуговування, заміна деталей чи інші важливі завдання. Застосунок попереджає про ці події заздалегідь, щоб уникнути можливих проблем із транспортним засобом.

Статистика: У цьому розділі представлено графіки та аналітику, що дозволяють користувачам оцінювати витрати на автомобіль у різних категоріях за вибраний період. Функція фільтрації дозволяє аналізувати витрати за конкретними параметрами, наприклад, лише заправки або ремонти.

Налаштування: У меню налаштувань користувачі можуть вибрати мову застосунку, змінити одиниці вимірювання (літри чи галони, кілометри чи милі), а також налаштувати нагадування. Крім того, є можливість додати кілька автомобілів для моніторингу, що зручно для власників автопарків або кількох транспортних засобів.

Інтерфейс "Car Expenses" створений з урахуванням потреб користувачів, забезпечуючи простоту та зручність використання. Його інтуїтивний дизайн дозволяє швидко знаходити потрібні функції, вводити дані та отримувати доступ до необхідної інформації. Завдяки цьому застосунок стає незамінним помічником для водіїв, які прагнуть контролювати витрати та підтримувати свої автомобілі в належному стані.

Для аналізу функціональних можливостей мобільних застосунків для моніторингу витрат на автомобіль були розглянуті три популярні програми: Car Expenses, Fuelio та MyCar. Кожен із цих застосунків має свої переваги та недоліки, які впливають на зручність використання та можливості. Таблиця 1.2 містить стислий огляд їхніх ключових характеристик.

Таблиця 1.2 - Стислий огляд існуючих застосунків

Застосунок	**Переваги**	**Недоліки**
Car Expenses	1. Інтуїтивно зрозумілий і простий у використанні інтерфейс.	1. Обмежені можливості аналізу витрат у вигляді графіків або звітів.
	2. Детальний облік витрат на ремонт, заправку та технічне обслуговування.	2. Відсутність синхронізації з хмарними сервісами.
	3. Можливість додавання нагадувань про технічні роботи та події.	3. Немає підтримки інтеграції з іншими пристроями чи системами.
Fuelio	1. Простий інтерфейс для обліку витрат на пальне.	1. Не підтримує облік витрат на ремонт або заміну деталей.
	2. Легкість експорту даних для зовнішнього аналізу.	2. Відсутність функцій нагадувань про майбутні технічні обслуговування.
	3. Підтримка кількох транспортних засобів для обліку витрат.	3. Недостатньо інструментів для комплексного аналізу витрат.
MyCar	1. Комплексний функціонал: облік витрат, створення нагадувань, генерація звітів.	1. Може бути складним для початківців через широкий набір функцій.
	2. Зручний облік декількох транспортних засобів у рамках одного акаунту.	2. Інтерфейс може здаватися перевантаженим для користувачів з мінімальними потребами.
	3. Аналітичні інструменти для візуалізації витрат у вигляді графіків і діаграм.	

1.5 Висновок до першого розділу

Аналізуючи ринок мобільних застосунків для моніторингу технічного стану автомобілів, було виявлено декілька наявних рішень, проте деякі з них мають обмежені можливості. Для створення нового застосунку важливо враховувати конкурентні переваги і недоліки цих аналогів, щоб запропонувати унікальний та поліпшений користувацький досвід.

Розробка користувацького інтерфейсу (UI) та користувацького досвіду (UX) є ключовими і взаємопов'язаними аспектами, що відіграють важливу роль у створенні привабливого та зручного застосунку. Поєднання сучасного естетичного дизайну з інтуїтивною навігацією допомагає користувачам легко

орієнтуватися в програмі, забезпечуючи безперешкодну та приємну взаємодію. Наприклад, можливість швидкого доступу до інформації про технічний стан автомобіля, таких як рівень масла, тиск у шинах або дата останнього техобслуговування, сприяє швидкому вирішенню повсякденних задач водія.

Одним з ключових етапів проектування інтерфейсу є проведення досліджень цільової аудиторії. Використання таких даних дозволяє розробникам глибше зрозуміти реальні потреби та очікування користувачів, усунути особисті припущення та переконатися, що застосунок відповідає практичним потребам своїх користувачів. Зокрема, для водіїв може бути важливо мати можливість персоналізувати налаштування програми відповідно до особливостей їхнього автомобіля, а також отримувати нагадування про регулярні технічні перевірки або ремонти.

У підсумку, успіх мобільного застосунку для відстеження технічного стану автомобіля залежить від якісно спроектованого інтерфейсу та зручного користувацького досвіду. Проектування, засноване на дослідженні цільової аудиторії, аналізі конкурентів і поєднанні UI/UX, є важливими кроками для створення продукту, який буде зручним, корисним та привабливим для користувачів.

2 ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Принципи проектування інтерфейсу

Проектування інтерфейсу – це творчий процес, який базується на впровадженні концепцій та ідей, що забезпечують високий рівень функціональності й ефективної взаємодії користувачів із системою. Основною метою цього процесу є визначення ключових принципів, які сприяють створенню зручного, інтуїтивно зрозумілого та ефективного інтерфейсу. [8]

Ключові принципи проектування:

1. Орієнтованість на користувача

Користувачі є центральною складовою будь-якої системи. Тому їхні потреби, очікування та комфорт взаємодії стають першочерговими завданнями під час розробки інтерфейсу. Важливо створити такий інтерфейс, який буде:

- Легким у використанні: мінімізація складних дій та інтуїтивно зрозумілий доступ до основних функцій.
- Інтуїтивно зрозумілим: елементи інтерфейсу повинні бути розташовані логічно, щоб користувач без зусиль міг зрозуміти їх призначення.
- Функціонально зручним: забезпечення простоти навігації та швидкого виконання основних завдань.

2. Простота та мінімалізм

Одним із найважливіших принципів є прагнення до мінімалізму. Простота дизайну сприяє підвищенню ефективності роботи користувачів, оскільки знижує ймовірність помилок і покращує загальний досвід взаємодії. Основні аспекти цього принципу:

- Чистота дизайну: уникайте перевантаження інтерфейсу зайвими елементами.

- Зрозумілість: використовуйте прості візуальні рішення, які полегшують сприйняття.
- Легкість у використанні: мінімалізм допомагає швидко орієнтуватися в інтерфейсі навіть новим користувачам.

3. Задоволеність користувачів

Задоволеність користувачів є ключовим фактором успіху будь-якого продукту. Цей показник залежить від кількох важливих аспектів:

- Функціональність: інтерфейс має підтримувати всі необхідні функції, які відповідають потребам користувача.
- Швидкість реакції: система повинна миттєво реагувати на дії користувача, щоб уникнути затримок.
- Надійність: стабільність роботи продукту гарантує довіру користувачів.
- Інтеграція: легкість об’єднання продукту з іншими платформами чи системами.
- Інформаційна підтримка: доступність підказок, навчальних матеріалів і повідомлень про помилки.

Збалансування цих факторів дозволяє створити інтерфейс, який буде зручним, зрозумілим і функціональним.

Простий і мінімалістичний дизайн не лише робить інтерфейс привабливим, але й сприяє підвищенню ефективності використання системи. Користувачам не потрібно витрачати час на вивчення складних елементів або пошук потрібних функцій. Це значно знижує рівень помилок і фрустрації під час роботи з продуктом.

Досвід користувачів (UX) формується на основі їхньої взаємодії з системою. Чим легше і приємніше ця взаємодія, тим вища задоволеність продуктом.

Успішний UX базується на таких складових:

- Зрозумілий інтерфейс.
- Логічна структура.

- Швидка реакція системи.
- Передбачуваність дій.

Фактори, що впливають на проектування інтерфейсу користувача, відіграють вирішальну роль у забезпеченні його зручності, функціональності та ефективності. Інтерфейс має чітко відображати основні функції системи, щоб користувачі могли легко зрозуміти, як з нею взаємодіяти. Швидка реакція системи є ще одним важливим аспектом: затримки у виконанні завдань знижують задоволеність і продуктивність користувачів.

Надійність інтерфейсу забезпечує безперебійну роботу системи в рамках встановлених параметрів. Це дозволяє економити час, уникати виникнення непередбачених помилок та створювати довіру користувачів до продукту.

Пристосованість до інтеграції також є важливою складовою. Вона забезпечує зручність взаємодії користувачів із платформою, дозволяючи легко інтегрувати продукт із іншими системами чи додатками.

Інформаційна підтримка користувачів допомагає їм швидко освоювати нові функції системи, орієнтуватися в інтерфейсі та вирішувати можливі труднощі. Вона включає доступ до підказок, довідкових матеріалів або навчальних інструкцій.

Після запуску продукту технічне обслуговування інтерфейсу стає важливим процесом. Його мета — покращення та оптимізація інтерфейсу для забезпечення високої якості роботи й відповідності новим вимогам користувачів.

Крім того, узгодженість, продуктивність та інтеграція є додатковими факторами, які впливають на задоволеність користувачів. Узгодженість у дизайні сприяє інтуїтивному розумінню, продуктивність системи забезпечує швидке виконання завдань, а гнучка інтеграція розширює функціональні можливості продукту.

Однією з поширених проблем є те, що на початкових етапах проектування користувачі часто не можуть чітко сформулювати свої вимоги до системи. Це призводить до труднощів у створенні інтерфейсу, який би відповідав їхнім

очікуванням. Лише після ознайомлення з функціональними можливостями продукту користувачі починають виявляти недоліки або вносити нові пропозиції.

Щоб уникнути проблем, необхідно активно залучати користувачів на ранніх етапах проектування. Це включає:

- Проведення досліджень їхніх потреб, очікувань та уподобань.
- Регулярне тестування прототипів із представниками цільової аудиторії.
- Використання ітеративного підходу, що дозволяє поступово вдосконалювати інтерфейс, враховуючи зворотний зв’язок.

На етапі проектування важливо конкретизувати й формалізувати вимоги до продуктивності, узгодженості та інтеграції інтерфейсу. Відсутність чіткого визначення цих характеристик може призвести до нерозуміння між замовниками та розробниками. Для цього слід:

- Використовувати вимірювані показники ефективності.
- Застосовувати тестові сценарії, які дозволяють оцінити, наскільки продукт відповідає вимогам користувачів.

Дизайнери повинні виявляти приховані потреби користувачів, які не завжди очевидні. Це допомагає уникати помилок у проектуванні та створювати продукт, який буде не лише функціональним, а й приємним у використанні.

Основні принципи проектування інтерфейсу, орієнтованого на користувача.

Розробка інтерфейсу, який буде зручним і ефективним для користувачів, вимагає дотримання ряду важливих правил. Ці правила забезпечують інтуїтивність, функціональність і привабливість системи для різних категорій користувачів.

1. Простота та зрозумілість

Інтерфейс має бути простим у використанні та зрозумілим для користувачів з різним рівнем технічної підготовки. Важливо уникати використання складних термінів і надмірно заплутаних елементів, які можуть ускладнити взаємодію.

Зрозумілий і лаконічний дизайн допомагає користувачам швидко орієнтуватися в системі.

2. Консистентність

Усі частини інтерфейсу повинні дотримуватися єдиного стилю. Використання спільних шаблонів, стандартних елементів і єдиних підходів до взаємодії полегшує навігацію для користувачів, знижуючи потребу в додатковому навчанні. Консистентність створює відчуття передбачуваності та довіри до системи.

3. Відповідність контексту

Інтерфейс має враховувати специфіку використання системи та потреби цільової аудиторії. Наприклад, мобільні застосунки потребують адаптації до невеликих екранів і обмеженого часу взаємодії користувачів. Особливості завдань, які вирішує система, також впливають на вибір дизайну.

4. Видимість та зрозумілість стану

Користувачі завжди повинні знати, що відбувається в системі. Для цього потрібно забезпечити індикатори стану, такі як:

- Сповіщення про успішні дії.
- Попередження про помилки.
- Відображення процесу завантаження чи обробки даних.

Ці елементи допомагають уникнути плутанини та покращують досвід користувачів.

5. Ефективність та швидкодія

Інтерфейс повинен забезпечувати швидке виконання завдань. Зменшення кількості дій для досягнення результату, оптимізація швидкості завантаження та реакції системи сприяють ефективності роботи користувачів і підвищують їх задоволеність.

Дизайн інтерфейсу має відповідати потребам і очікуванням користувачів. Платформа повинна бути гнучкою та налаштованою під замовника. Три основні принципи розробки користувацького інтерфейсу:

- Простота та зрозумілість: Забезпечення легкості використання інтерфейсу, мінімізації складних дій.
- Зниження навантаження на пам'ять користувача: Використання знайомих шаблонів, які спрощують навігацію та взаємодію.
- Логічна послідовність: Організація інтерфейсу у відповідності до логіки виконання завдань.

Ці принципи залежать від апаратного забезпечення, операційної системи, компонентів інтерфейсу та завдань, які виконує система.

Навігація в інтерфейсі

Користувачі повинні мати чіткі орієнтири для переміщення в системі. Елементи, які спрощують навігацію:

- Заголовки вікон: Вказують, де перебуває користувач у системі.
- Навігаційні карти: Забезпечують візуальне уявлення про структуру системи.
- Стійкі конструкції: Повторювані елементи, які забезпечують передбачуваність.

Миттєвий динамічний відгук інтерфейсу допомагає користувачам розуміти своє місцезнаходження та наступні дії. Важливо уникати ситуацій, коли користувач змушений змінювати умови роботи чи адаптуватися до різних стилів введення.

Розробка інтерфейсів для мобільних пристроїв має враховувати обмеження, пов’язані з розміром екрану. Особливості мобільного інтерфейсу:

- Одне вікно на екрані: Мобільні користувачі здебільшого бачать лише одну функціональну область одночасно. Це вимагає чіткого фокусування на одному завданні.
- Простота навігації: Основні елементи мають бути легко доступними для взаємодії однією рукою.
- Ефективне використання простору: Важливо адаптувати дизайн до невеликих екранів, забезпечуючи зручність і зчитуваність.

2.2 Структура мобільного застосунку

Ефективна структура мобільного застосунку є важливим аспектом, що визначає зручність його використання, якість розробки та можливість індексації пошуковими системами. Продумана структура вирішує багато завдань, забезпечуючи зручну навігацію, логічність взаємозв'язків і зручність для кінцевого користувача.

Структура мобільного застосунку — це схема організації екранів, категорій і функцій, яка демонструє зв'язки між компонентами. Вона є основою дизайну та розробки застосунку, що визначає, які функції будуть включені та як вони взаємодіятимуть між собою. Навігація в мобільному застосунку представлена набором URL або маршрутів, які об'єднують усі елементи системи в єдине ціле. [9]

Залежно від завдань, застосунок має дві ключові складові структури:

- Зовнішня структура: відображає розташування блоків на екранах і визначає візуальну організацію інтерфейсу.
- Внутрішня структура: охоплює ієрархію категорій, логічний зв'язок між сторінками та організацію інформаційного наповнення.

Тип структури залежить від мети застосунку та його функціоналу. Основні типи структур включають:

- Ієрархічна структура: Контент організований у вигляді дерева, де головний екран веде до підекранів і підкатегорій. Цей тип підходить для застосунків із великою кількістю сторінок або функцій, таких як інтернет-магазини чи освітні платформи.
- Пласка структура: Забезпечує прямий доступ до всіх ключових функцій без складної ієрархії. Вона ідеально підходить для невеликих застосунків, наприклад, нотатників або калькуляторів.

- Структура на основі вкладок: Використовує вкладки для розділення функцій або категорій. Цей тип забезпечує легкий доступ до різних частин застосунку та дозволяє швидко перемикатися між ними.
- Структура на основі карт: Забезпечує візуалізацію зв’язків між сторінками за допомогою карт або інтерактивного меню. Вона підходить для складних систем із великою кількістю інформації, наприклад, навігаційних або туристичних застосунків.

Щоб структура мобільного застосунку була ефективною, необхідно дотримуватися наступних рекомендацій:

- Зрозуміла навігація: Користувачі повинні легко знаходити потрібні функції та розділи.
- Логічність: Усі екрани й сторінки мають бути організовані так, щоб користувач міг швидко орієнтуватися в системі.
- Простота використання: Інтерфейс має бути доступним навіть для новачків, без потреби в додатковому навчанні.

Для демонстрації логіки навігації та зв’язків між елементами застосунку було створено деревоподібну структуру, що забезпечує доступ до всіх необхідних екранів.

Рисунок 2.1 ілюструє схему екранів інформаційної системи, включаючи головний екран, підекрани та функціональні блоки.

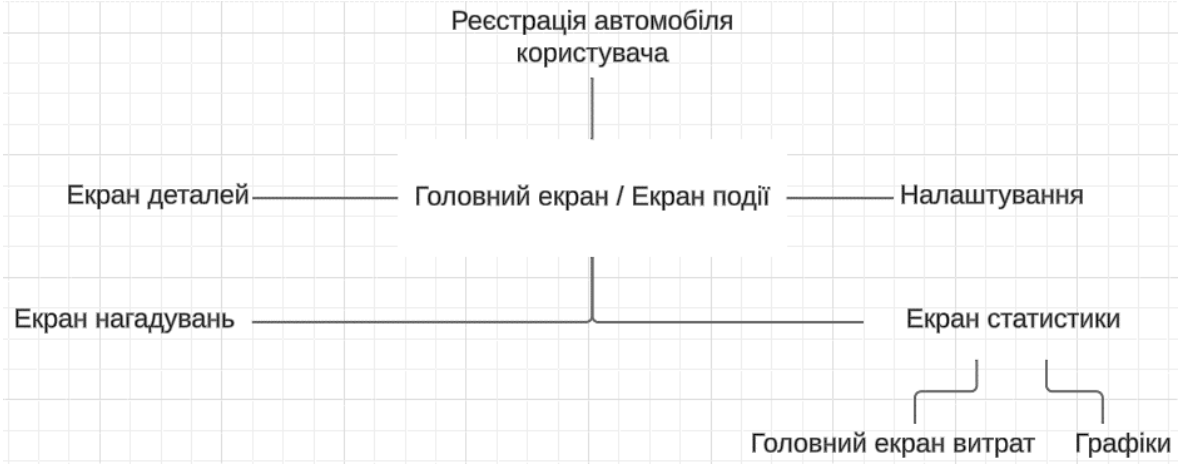


Рисунок 2.1 – Структура екранів інформаційної системи.

Мобільний застосунок буде мати такі екрани:

- Реєстрація автомобіля користувача;
- Екран подій / Головний екран;
- Екран деталей;
- Екран нагадувань;
- Екран налаштування;
- Екран статистики;
- Головний екран витрат;
- Графіки;

Дана структура буде зрозумілою та інтуїтивною у використанні, адже має звичні сторінки для користувача.

2.3 Розробка вайрфеймів для інтерфейсу мобільного додатка

Процес створення вайрфреймів забезпечує створення базової логічної і функціональної структури майбутнього продукту, тому на важливому етапі розробки сучасних інтерфейсів вайрфрейми служать спрощеним графічним зображенням, що показує базову структуру, інформаційну ієрархію і взаємодію користувача з системою. Вони являють собою своєрідну візуалізацію дизайнерської концепції, яка буде використовуватися як орієнтир для подальшого розвитку, включаючи всі ключові компоненти майбутнього продукту.

Основною метою створення каркасів є створення інтуїтивно зрозумілого і зручного для користувача інтерфейсу, який відповідає потребам користувача і забезпечує логіку взаємодії з продуктом завдяки цьому інструменту дизайнери можуть визначити базову структуру інтерфейсу, сформувати набір дій користувача і забезпечити узгодженість всіх елементів. Крім того, каркаси стають платформою для тестування прототипів і виявлення деталей майбутнього дизайну.

Створення каркасів засноване на декількох важливих принципах, які допомагають зберегти ясність, функціональність і гнучкість. Один з головних принципів-простота. Це означає, що навіть користувачі, які не мають технічного досвіду, повинні розуміти вайрфейми і передавати основну ідею структури і функціональності продукту без зайвих деталей. Ясність-ще один важливий принцип, оскільки він дозволяє побачити ієрархію інформації, логіку розташування компонентів і взаємозв'язку між ними. Такий підхід полегшує розпізнавання інтерфейсу і забезпечує його узгодженість.

Функція каркаса полягає у відображенні основних функцій системи. Це допоможе визначити, які елементи повинні бути присутніми, їх розташування в загальній структурі та їх взаємодія з іншими компонентами. У свою чергу, узгодженість сприяє єдності дизайну між різними екранами або сторінками, забезпечуючи безперервність взаємодії з користувачем. Ще одним важливим аспектом є гнучкість, яка дозволяє швидко змінювати дизайн і адаптувати його до нових потреб і тестів. Крім того, каркаси орієнтовані на взаємодію користувача з системою і вказують логіку дій за допомогою кнопок, полів введення або елементів навігації. [10]

Процес створення вейрфреймів є ітеративним і має на увазі поетапне поліпшення на основі відгуків команди або користувачів. На початковому етапі визначаються Головна сторінка і функції, після чого для розміщення елементів використовуються прості геометричні форми. Щоб не відволікати увагу від функціональності, каркаси зазвичай виготовляються в нейтральних кольорах, таких як чорний, білий і сірий. Такий підхід дозволяє зосередитися на функціональності, логіці взаємодії і розстановці пріоритетів контенту.

Використання каркасів має багато переваг. Це дозволяє впорядкувати функціональність продукту, визначити найбільш важливі елементи інтерфейсу і протестувати логіку роботи системи на ранніх етапах проектування. Крім того, каркаси сприяють поліпшенню взаємодії з користувачем, забезпечуючи інтуїтивне і логічне взаємодія. Це також може допомогти вам визначити

пріоритетність важливого вмісту відповідно до потреб вашої аудиторії та створити основу для подальшої розробки дизайну.

На заключному етапі в якості основи для розробки візуального дизайну використовуються каркаси. Це дозволяє деталізувати зовнішній вигляд продукту без зміни його функціональності і логіки. Потім створюється інтерактивний прототип, який використовується для тестування та подальших удосконалень. Таким чином, вайрфейм відіграє важливу роль у дизайні інтерфейсу, забезпечуючи гармонійне поєднання функціональності, зручності та візуальної привабливості. [10]

Під час проектування користувацького інтерфейсу, використовуємо отриману в результаті дослідження користувачів інформацію для розробки вайрфреймів основних екранів. Цей підхід дозволив визначити, який контент має високий пріоритет і визначити основні функції для ключових сторінок майбутньої продукції:

На рисунку 2.2 зображено процес реєстрації автомобіля користувачем у системі. Інтерфейс пропонує користувачеві ввести основну інформацію про автомобіль, включаючи марку та модель, тип пального, а також тип коробки передач.

На рисунку 2.3 зображено екран подій, який є головним екраном інформаційної системи, призначений для надання користувачеві всебічного огляду фінансових операцій, пов'язаних із його автомобілем. У центрі екрану розташоване функціональне зображення автомобіля, яке може використовуватися для візуалізації або взаємодії з даними, що стосуються транспортного засобу.

На рисунку 2.4 зображено екран деталей автомобіля надає користувачеві доступ до детальної інформації про стан і облік основних компонентів автомобіля. Цей екран відкривається після натискання на функціональну зону автомобіля на екрані подій.

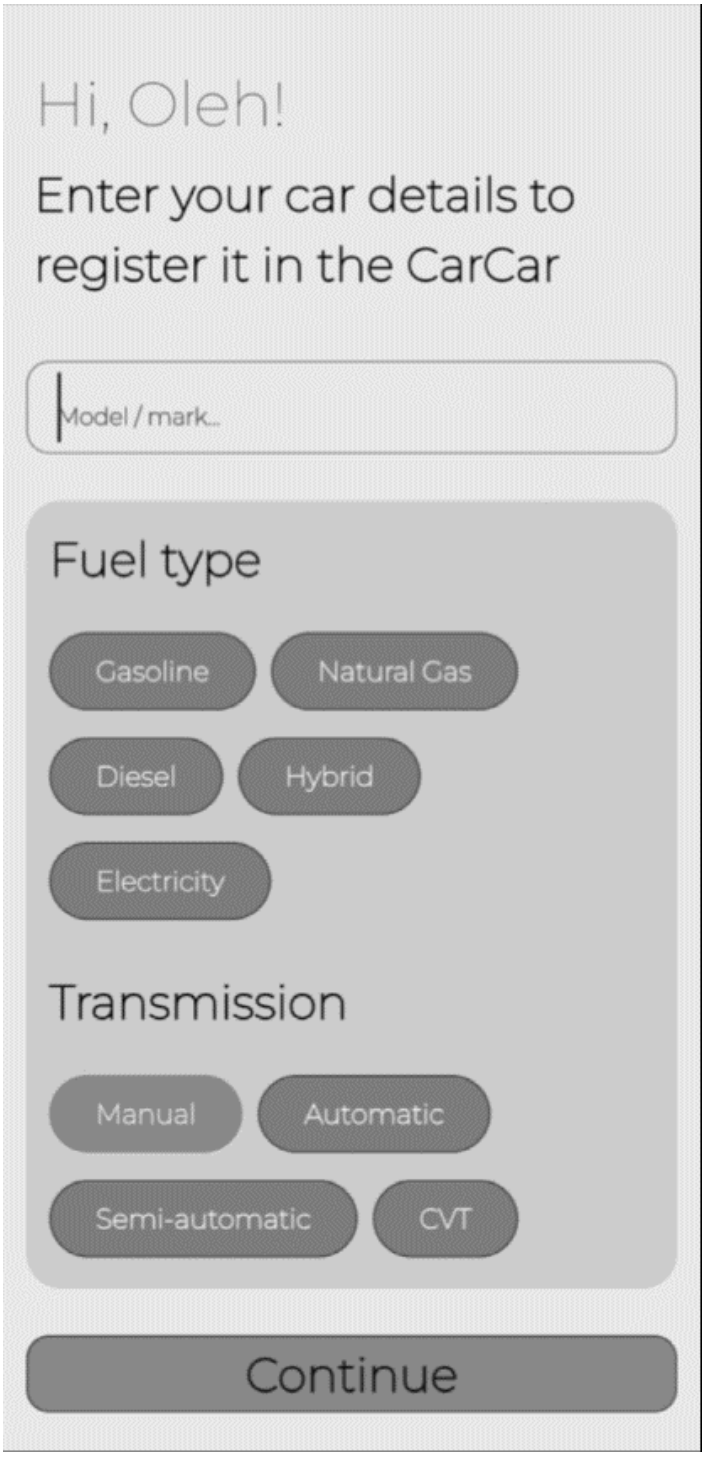


Рисунок 2.2 – Варфрейм реєстрація автомобіля.

На рисунку 2.5 зображено екран нагадувань забезпечує користувача інструментами для зручного управління нагадуваннями, пов'язаними з автомобілем. Він містить список уже створених ремайнерів і забезпечує простий доступ до функції додавання нових нагадувань.

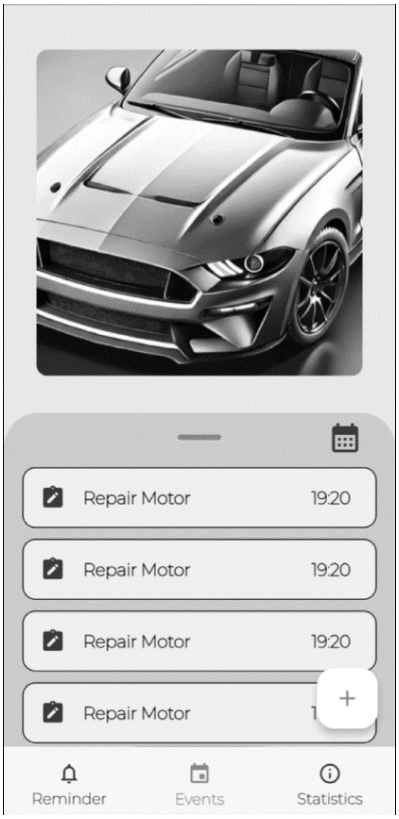


Рисунок 2.3 – Варфрейм подій.

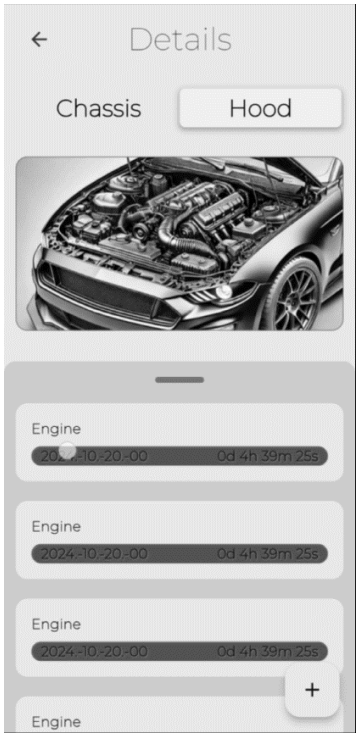


Рисунок 2.4 – Варфрейм деталей.

На рисунку 2.6 зображено екран статистики витрат на авто представляє детальний аналіз витрат, допомагаючи користувачеві контролювати фінансові витрати на обслуговування автомобіля. Цей екран є третім пунктом нижнього меню навігації та пропонує кілька функціональних елементів для роботи з даними.

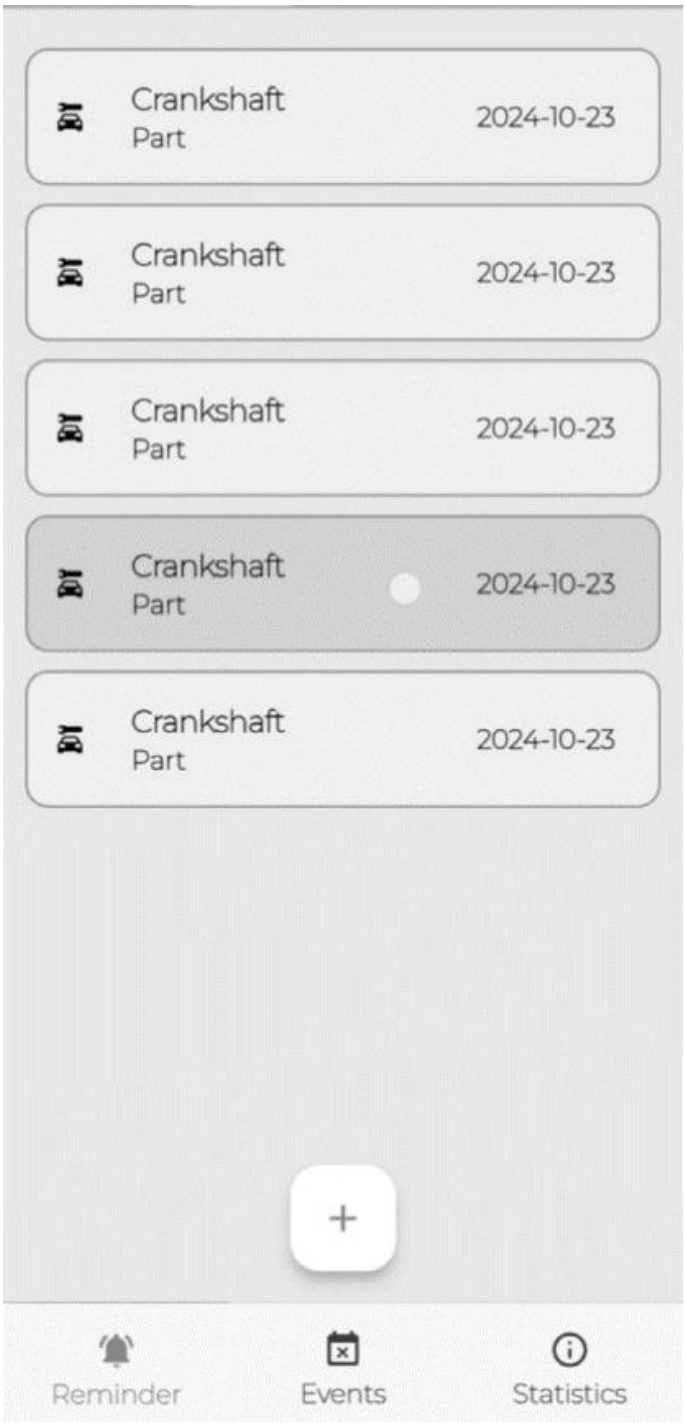


Рисунок 2.5 – Варфрейм нагадувань.

а) Лінійний графік б) Кругова діаграма

Рисунок 2.6 – Варфрейми статистики.

2.4 Визначення шляхів користувача інформаційної системи (User flow)

User Flow – це ключовий інструмент у створенні ефективного інтерфейсу, який відображає послідовність дій користувача в системі або застосунку [11]. Він дозволяє зрозуміти, як користувач взаємодіє з продуктом, щоб досягти своїх цілей. Це візуалізація всіх можливих етапів і шляхів, які користувач проходить, починаючи з початкової точки і до отримання бажаного результату.

Створення User Flow є невіддільною частиною процесу розробки, оскільки воно допомагає вибудувати логічну структуру продукту та забезпечити зрозумілість для користувачів. Зазвичай User Flow оформлюється у вигляді діаграм або описових схем, які детально демонструють усі кроки, переходи та взаємодії.

У процесі проектування User Flow враховуються різні аспекти, такі як основні завдання користувача, його потреби та контекст використання. Наприклад, якщо користувач шукає спосіб здійснити покупку через застосунок,

User Flow покаже кожен етап цього процесу: від реєстрації або входу в систему до перегляду товарів, вибору, оплати та підтвердження замовлення.

Головною метою створення User Flow є забезпечення зручності та логічності всіх дій користувача. Такий підхід дозволяє користувачам швидко орієнтуватися в системі, мінімізує кількість зайвих кроків і складних моментів, які можуть викликати роздратування або нерозуміння. Завдяки цьому значно покращується загальний користувацький досвід.

User Flow також допомагає виявляти можливі проблеми ще на етапі розробки. Наприклад, якщо певна дія вимагає надто багато кроків, це може сповільнити користувача або навіть змусити його відмовитися від використання продукту. Аналізуючи подібні ситуації, розробники можуть вносити зміни, оптимізуючи маршрут і зменшуючи кількість дій.

Цей інструмент також дозволяє враховувати різні сценарії використання продукту. Користувачі можуть мати різні цілі та потреби, тому важливо передбачити кілька шляхів, які приведуть до бажаного результату. Наприклад, для тих, хто лише знайомиться із застосунком, маршрут може бути простішим і включати підказки, тоді як для досвідчених користувачів передбачаються скорочені шляхи виконання дій.

Розробка User Flow – це також спосіб забезпечити тісний зв’язок між різними відділами команди розробників. Дизайнери, програмісти, аналітики та менеджери можуть разом працювати над створенням оптимального маршруту, враховуючи як технічні обмеження, так і потреби цільової аудиторії.

Зрештою, User Flow є важливим етапом проектування інтерфейсу, який дозволяє створити продукт, що відповідає очікуванням користувачів. Він забезпечує не лише зручність і логічність, але й допомагає мінімізувати помилки та складнощі, які можуть виникнути в процесі використання. Створення якісного User Flow є основою для розробки сучасного інтуїтивного та функціонального інтерфейсу.

У сучасному проектуванні цифрових продуктів існує кілька основних видів user flow, які допомагають визначити шляхи взаємодії користувачів із системою. Ці види розрізняються за способом візуалізації, обсягом охоплення процесів та деталізацією, що дозволяє адаптувати їх до потреб і складності продукту.

Види User Flow за способом візуалізації

- Діаграма потоку користувача. Цей тип представляє послідовність дій користувача у вигляді графічного зображення. Використовуються стрілки, лінії чи блоки для відображення переходів між сторінками або екранами. Такий підхід є зрозумілим і наочним, дозволяє швидко оцінити загальну логіку системи та виявити проблемні місця.
- Прототип із функціональними переходами. Інтерактивна форма user flow, яка дозволяє взаємодіяти з прототипом продукту. Користувачі можуть переходити між екранами, відтворюючи реальні сценарії використання. Цей метод особливо корисний для тестування функціональності й оцінки взаємодії користувачів з інтерфейсом.

Види User Flow за обсягом

- Макро-юзер-флоу. Охоплює весь процес взаємодії користувача з продуктом, починаючи від початкового етапу (наприклад, реєстрація) і до досягнення кінцевої мети (наприклад, успішне замовлення). Такий підхід допомагає зрозуміти загальну структуру системи та побачити всі основні етапи.
- Мікро-юзер-флоу. Зосереджується на деталях конкретного завдання чи функції. Наприклад, це може бути послідовність дій для заповнення форми або оплати замовлення. Такий підхід дає змогу ретельно опрацювати деталі взаємодії з елементами інтерфейсу.

Обсяг і деталізація user flow залежить від складності системи та специфічних потреб користувачів.

Ефективність user flow залежить від чіткого розуміння потреб користувачів.

Це складне завдання, яке включає відповіді на ключові питання:

- Які цілі користувач прагне досягти, використовуючи продукт?

- Що мотивує користувача діяти?
- Які особливості програми допомагають досягти цієї мети?
- Що може завадити використанню програми?
- Яка інформація або функціонал є найважливішими для користувача?
- Які емоційні стимули впливають на дії користувачів?

Ці питання дозволяють глибше зрозуміти мотивацію користувачів і створити user flow, що враховує всі аспекти їхньої взаємодії з продуктом.

Після створення основних вайрфреймів user flow використовується для оптимізації навігаційної системи. Він структурує переходи між сторінками, створюючи чітку та логічну архітектуру. У складних багаторівневих системах user flow є ключовим елементом, що забезпечує зручність використання.

Розробка user flow дозволяє побачити слабкі місця в системі, усунути зайві кроки та зробити взаємодію користувача максимально простою. Це також важливий етап, який об’єднує всі аспекти проекту перед переходом до дизайну інтерфейсу. [11]

Після завершення всіх етапів проектування, наступним кроком є розробка дизайну інтерфейсу, який відображатиме зовнішній вигляд інформаційної системи. Останньою стадією є відтворення макету.

На рисунку 2.7 показано послідовність кроків, які користувач здійснює при використанні застосунка.

Рисунок 2.7 – User flow мобільного застосунку.

2.5 Висновок до другого розділу

У процесі розроблення інтерфейсу мобільного застосунку було сформульовано ключові принципи, які лягли в основу створення зручного та інтуїтивно зрозумілого дизайну. На основі цих принципів було розроблено структуру застосунку, що забезпечує логічну організацію елементів і оптимальний доступ до ключових функцій.

Як візуальну основу для дизайну було створено каркасні схеми. Це спростило процес проектування і забезпечило чітке розуміння розташування і функцій кожного елемента інтерфейсу. Крім того, було створено докладні шляхи користувачів в інформаційній системі. Це дало змогу врахувати всі сценарії використання застосунку, підвищивши зручність та ефективність для кінцевого користувача.

Загалом ця робота стала основою для подальшого розвитку інтерфейсу відповідно до сучасних стандартів UX/UI дизайну.

3 РОЗРОБКА ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Програмні засоби для проектування інтерфейсу

Створення дизайну інтерфейсу програми-один з найважливіших етапів розробки програмного забезпечення. Цей процес передує створенню програмного коду і визначає структуру і функціональність майбутніх продуктів. Помилки на цьому етапі ускладнюють процес розробки, оскільки вимагають багато зусиль та ресурсів для виправлення. Непроектований інтерфейс призводить до необхідності перепрограмування, затримки випуску продукту та збільшення вартості.

На початковому етапі проектування зазвичай створюються прототипи 2-х типів: чорнові і остаточні. Приблизний прототип - це основна схема екрану, яка показує загальну структуру програми, ключові області та їх взаємне розташування. Це дає лише загальне уявлення про такі функції, як розташування списків новин та заголовків сайту, але не містить деталей. На цьому етапі ви можете оцінити кількість інформації, яку потрібно розмістити на кожному екрані, і спланувати кількість кліків, щоб перейти на потрібну сторінку.

Остаточний прототип-це наступний крок у процесі проектування. Він містить більш детальну інформацію про розташування кнопок, тексту, форм, прапорців та інших елементів. Остаточний прототип ще не включає графічний дизайн (кольори, іконки, зображення), але ви можете спланувати функціональність та взаємодію між елементами. На цьому етапі закладається фундамент для зручного і інтуїтивно зрозумілого взаємодії з користувачем в майбутньому.

Розробка користувальницького інтерфейсу тісно пов'язана з останніми тенденціями в UX/UI дизайні і постійно змінюється під впливом технологій. Інструменти для створення користувальницьких інтерфейсів також постійно вдосконалюються, надаючи дизайнерам можливість реалізувати свої найсміливіші

ідеї. Сьогодні існує безліч програм, які дозволяють спроектувати інтерфейс як веб-сайту, так і мобільного додатку. [12]

Одним з найпопулярніших інструментів є Figma. Це графічний редактор і платформа для створення прототипів, яка в основному працює в Інтернеті. Figma підтримує співпрацю в режимі реального часу, що є ключовою перевагою командного проектування. Наприклад, розробники, копірайтери та дизайнери можуть одночасно працювати над проектами, обговорювати ідеї та вносити зміни. Крім того, Figma має мобільний додаток для перегляду прототипів на Android та iOS, що дозволяє користувачам бачити, як виглядає дизайн на різних пристроях. Його функціональність спрямована на оптимізацію розробки користувальницького інтерфейсу і спільної роботи. [13]

Ще одним популярним інструментом є Adobe XD. Програмне забезпечення надає повний набір функцій для створення цифрового дизайну, включаючи прототипування, макети та взаємодію між елементами. Adobe XD простий у використанні та сумісний як з настільними, так і з мобільними платформами. Можливість спільного використання прототипів-це універсальне рішення для команд, що працюють над веб-дизайном і мобільними додатками. [14]

Для розробників з обмеженим бюджетом GIMP є ефективним інструментом. В першу чергу він позиціонується як графічний редактор, але його функціональність дозволяє створювати такі елементи інтерфейсу, як кнопки, меню, значки і поля введення. За допомогою GIMP ви можете використовувати такі функції, як: за допомогою цього інструменту ви можете налаштувати кольори та стилі тексту, а також використовувати шари для створення складних елементів.1. Однією з переваг GIMP є його привабливість для невеликих команд та індивідуальних дизайнерів. [15]

При розробці дизайну інтерфейсу також слід враховувати деталі проекту. Наприклад, для мобільних додатків важливо забезпечити простоту використання на маленькому екрані, для веб-сайтів - адаптивність і швидкість завантаження.

При проектуванні інтерфейсу особлива увага приділяється структурі сторінки, логіці переходів між сторінками і доступності для користувачів різних категорій.

Одним з найважливіших аспектів розробки UX/UI є тестування прототипу. Тестування допомагає виявити конструктивні і функціональні недоліки ще до початку розробки програмного коду. Наприклад, ви можете використовувати інтерактивні прототипи для перевірки зручності навігації та візуалізації інформації або для перевірки відгуків потенційних користувачів.

Таким чином, створення дизайну інтерфейсу є важливим кроком, який вплине на успіх програмного забезпечення в майбутньому. Продумана структура, продуманий функціонал і естетично привабливий інтерфейс забезпечують користувачам позитивний досвід і сприяють успішному виконанню бізнес-завдань. Сучасні інструменти, такі як Figma, Adobe XD та GIMP, спрощують процес проектування та забезпечують високу якість кінцевого продукту. Висока якість кінцевого продукту гарантована.

3.2 Розробка дизайну інтерфейсу мобільного застосунку

Спочатку були обрані концепції, які найкращим чином відображають ідеологію мобільних додатків і актуальні сьогодні.

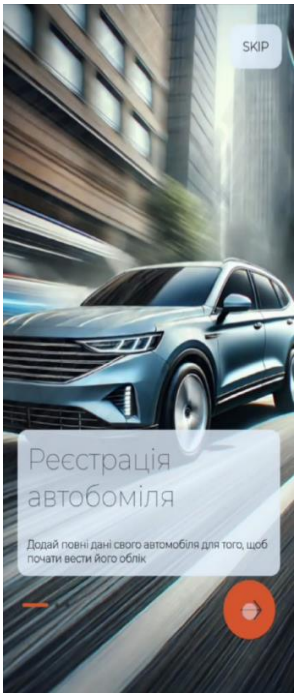
На рисунках 3.1 та 3.2 Показані вибрані шрифти та кольори, що використовуються в інтерфейсі.

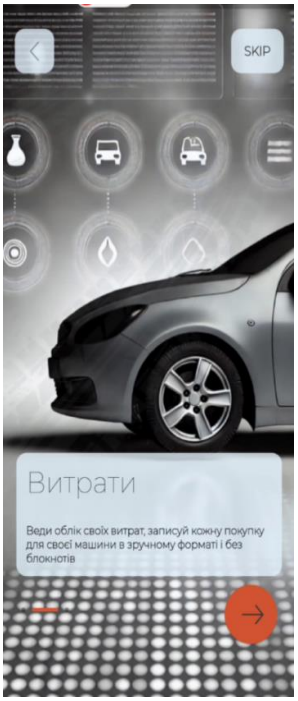
Рисунок 3.1 – Кольори інформаційної системи

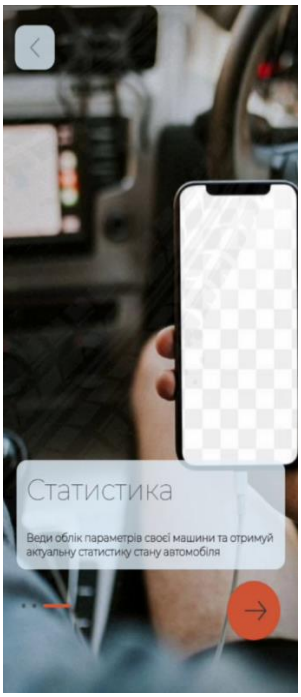
Рисунок 3.2 – Типографіка інформаційної системи

Після встановлення кольорової палітри та типографічних рішень, наступним кроком було розроблений інтерфейс мобільного застосунку.

Було створено вступний екран, котрий зображено на рисунку 3.3. У цьому екрані зображено вступні екрани, які містять короткий зміст о застосунку.







а)Реєстрація б)Витрати в) Статистика

Рисунок 3.3 – Вступні інформативні екрани

Наступний екран зображено на рисунку 3.4 екран реєстраціїї автомобіля. На цьому екрані буде зображено поле з водом назви автомобіля, також вибор палива на якому пересувається транспорт та тип перемикання коробки передач.

На рисунку 3.5 та 3.6 зображено екран з записами події та додаваня події автомобіля. На цьому екран буде можливість переглядати за датую які події відбулися на приклад ремонт двигателя або затрати на палива. Також на цьому еркані у нижньому правому углі є кнопка плюс. При натискані якої з’являється можливість додати такі події як:

- Одометр. Суточний пробіг автомобіля за день
- Примітка.
- Сервіс.
- Витрати.
- Заправка.

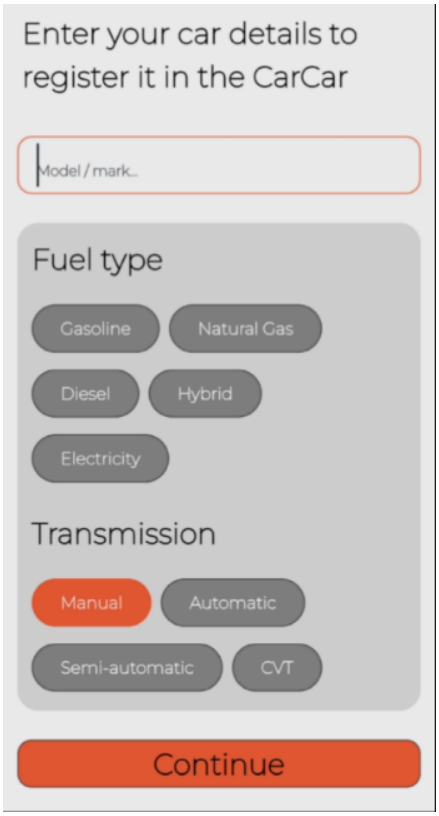


Рисунок 3.4 – Екран реєстрації автомобіля

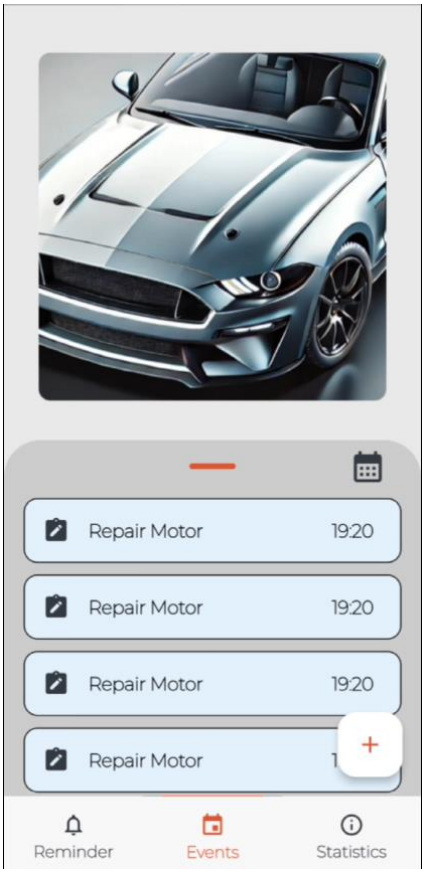


Рисунок 3.5 – Екран з записами події автомобіля

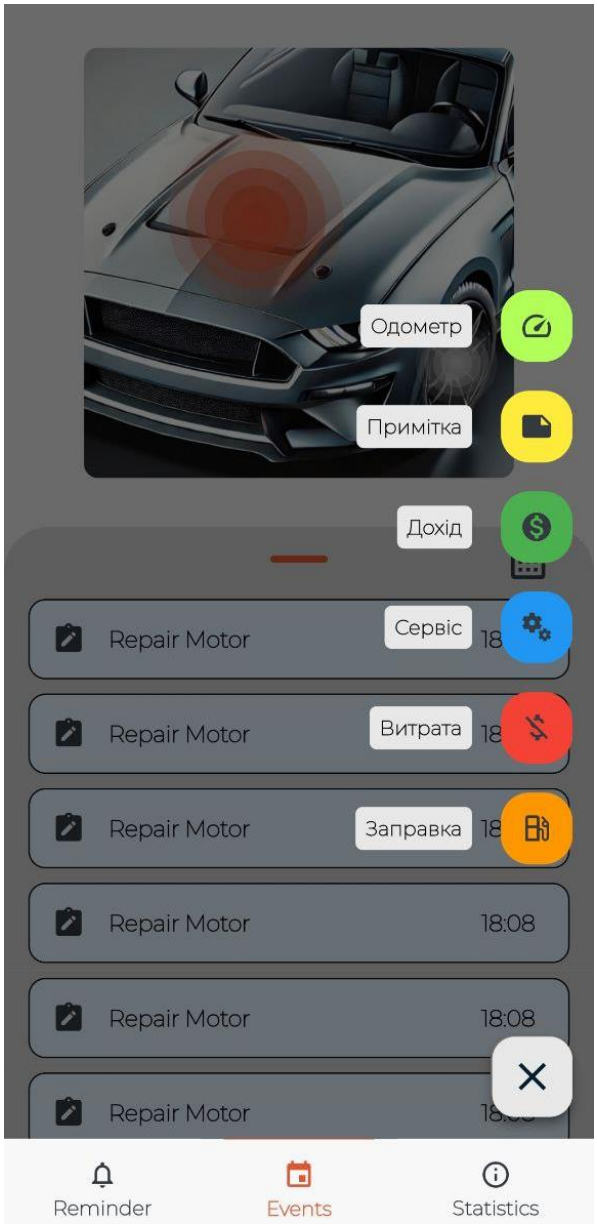


Рисунок 3.6 - Екран з додаванням інформації

На рисунку 3.7 представлений екран для додавання витрат на пальне. Він включає кілька полів:

- Поле для введення суми витрат.
- Вибір типу пального серед шести варіантів (бензин, природний газ, дизель, гібрид, електроенергія, або інше).
- Поле для текстового опису, яке дозволяє деталізувати запис, наприклад: «Бензин 10 л – 700 грн, Газ 15 л – 557 грн».

На рисунку 3.8 зображено приклад заповнення цього екрану, де користувач ввів суму витрат, обрав типи пального (бензин і природний газ) та додав опис. Завершення заповнення форми здійснюється натисканням кнопки «Створити замітку».

Рисунок 3.7 - Екран додавання витрат на паливо

Рисунок 3.8 – Екран з заповненими даними

Додані записи відображаються в загальному списку подій на основному екрані, що забезпечує користувачу зручний доступ до інформації про історію експлуатації автомобіля.

Було створено екран з нагадуванням на рисунку 3.9. На цьому екрані зображено події з нагадування які будуть у майбутньому. Також на низу екрана є кнопка плюс у котрій можна буде створити майбутнє нагадування.

На рисунку 3.10 екран з витратами на місяць. На цьому екрані зображено витрати на поточний місяць у двох типах а) діаграмна та б) кругова.

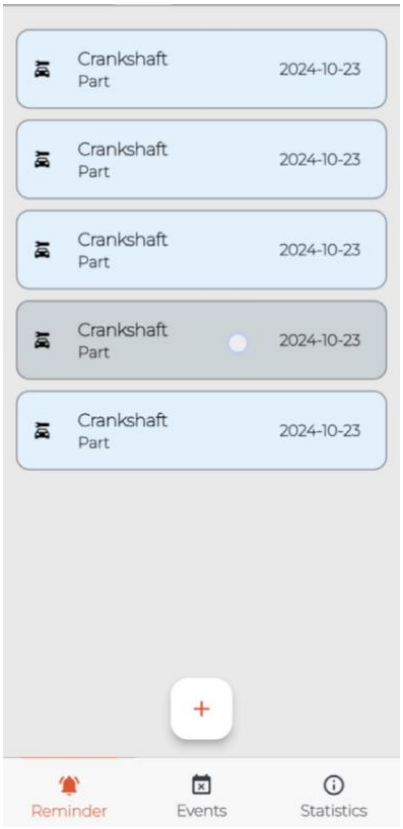
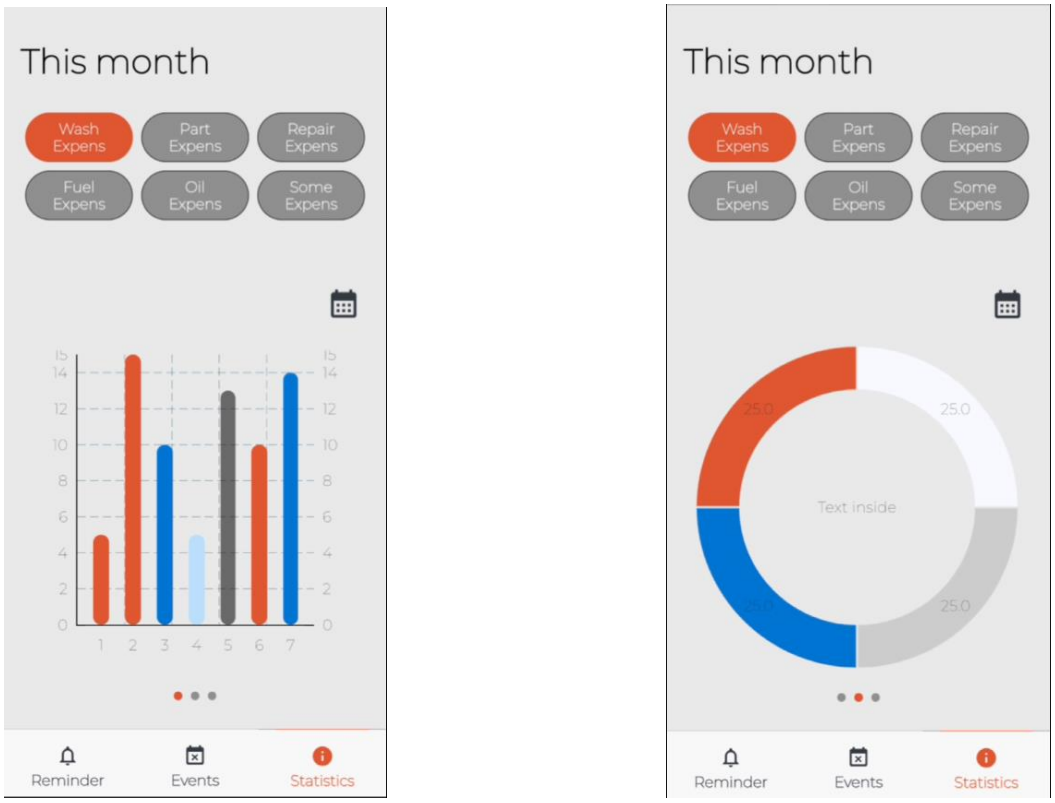


Рисунок 3.9 – Екран нагадування



а) Діаграмна б)Кругова

Рисунок 3.10 - Екран з затратами за поточний місяць

В результаті були розроблені концепції та компоненти інформаційної системи соціальної взаємодії, а екрани додатків Figma стали прототипом

3.3 Опис реалізації мобільного інтерфейсу

Flutter - це сучасний фреймворк для розробки мобільних додатків, який дозволяє створювати інтерфейси, використовуючи лише код. Він розроблений спеціально для оптимізації мобільних додатків і заснований на мові програмування Dart, яка є основною мовою в екосистемі Flutter. Центральною концепцією Flutter є" віджети", які є основними будівельними блоками для створення елементів інтерфейсу користувача. [16]

Віджети Flutter - це основні елементи інтерфейсу, які представляють різні компоненти інтерфейсу користувача, від простих кнопок і текстових полів до складних списків та анімації. Завдяки віджету ви можете створити будь-який інтерфейс, комбінуючи елементи і задаючи властивості. Flutter пропонує розробникам широкий вибір готових віджетів, які ви можете використовувати в проектах або створювати свої власні призначені для користувача елементи.

Flutter використовує декларативний підхід до опису інтерфейсу, в якому кожен компонент інтерфейсу описується як ієрархія віджетів. Цей підхід, який називається "конфігурація віджетів", дозволяє створювати складні інтерфейси, використовуючи основні елементи як будівельні блоки. Наприклад, ви можете розміщувати віджети, групувати їх у контейнери або використовувати макети для їх упорядкування.

Популярні класи віджетів у Flutter:

– Контейнер-це базовий елемент компонування та оформлення інших віджетів.

– Текст - текстовий елемент.

– Кнопки-різні типи кнопок.

– ListView-створює список.

– Image-для маніпулювання графічними елементами.

– Rows і columns-впорядковує елементи в рядки або стовпці.

Система дозволяє писати інтерфейси з використанням структурованого коду, забезпечуючи гнучкість в налаштуванні елементів.

Усі інтерфейси Flutter створюються за допомогою коду Dart. Синтаксис мови схожий з сучасними мовами програмування, такими як JavaScript і Java, що полегшує розуміння його досвідченими розробниками. Dart дозволяє:

– Описує властивості віджета.

– Організовує взаємодію між елементами.

– Налаштування зовнішнього вигляду і поведінки користувальницького інтерфейсу.

Це рішення забезпечує більшу гнучкість у розробці та дозволяє створювати складні інтерфейси з урахуванням усіх потреб вашого проекту.

Переваги підходу Flutter до створення інтерфейсів:

1. Простота розробки: використання готових віджетів значно спрощує процес створення інтерфейсу. Flutter також має можливість створювати власні віджети для унікальних завдань.
2. Гнучкість: завдяки конфігурації віджета ви можете налаштувати інтерфейс відповідно до будь-яких вимог.
3. Швидкість роботи: вбудований інструмент гарячої перезавантаження дозволяє миттєво побачити зміни в інтерфейсі без необхідності повного перезавантаження програми.
4. Масштабованість: ієрархічний підхід дозволяє легко змінювати структуру інтерфейсу та адаптувати її до ваших потреб.

Visual Studio Code, легке безкоштовне середовище розробки, часто використовується для розробки інтерфейсів у Flutter. Вона підтримує:

– Плагіни для Flutter і Dart.

– Зручне редагування і налагодження коду.

– Інтеграцію з інструментами тестування користувальницького інтерфейсу і настройки.

Visual Studio Code забезпечує високу продуктивність і простоту роботи зі складними проектами.

Flutter пропонує унікальний підхід до створення інтерфейсів за допомогою віджетів та декларативних стилів програмування. Dart, широкий вибір вбудованих елементів та інтуїтивно зрозуміле середовище Visual Studio Code роблять розробку ефективною та доступною. Це робить Flutter потужним інструментом для створення сучасних мобільних додатків з гнучким і масштабованим інтерфейсом.

На рисунку 3.11 приведено структуру проекту, котра відповідає розробленим екранам.

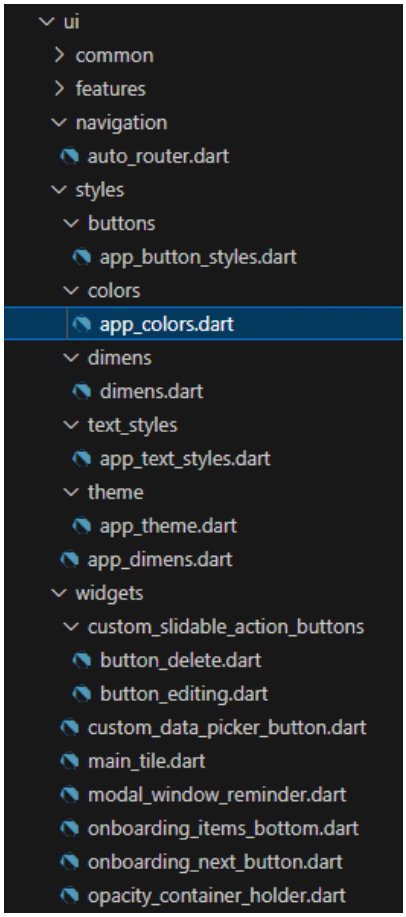


Рисунок 3.11 – Структура проекту

У подальшому кожен елемент цієї структури буде детально описано, зокрема наведено фрагменти коду, що реалізують функціонал окремих екранів та компонентів. Це дозволить продемонструвати взаємодію між модулями та логіку роботи додатку, а також пояснити ключові аспекти програмної реалізації.

Клас NotesManager реалізує механізм обробки валідації та надання зворотного зв'язку користувачеві під час взаємодії з мобільним додатком (Рис. 3.12). Основний метод цього класу, checkValidation, відіграє ключову роль у забезпеченні зручності та зрозумілості інтерфейсу. Він відповідає за аналіз стану введених даних і динамічне інформування користувача про результати перевірки.

```dart
import 'package:car_car/ui/styles/colors/app_colors.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

class NotesManager {
  static checkValidation(
      bool? isError, bool? isValidate, BuildContext context, String title) {
    if (isError != null && isError == true) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            backgroundColor: Theme.of(context).focusColor,
            content: Text('Сталась якась помилка',
                style: Theme.of(context)
                    .textTheme
                    .displayMedium
                    ?.copyWith(fontWeight: FontWeight.bold))),
      );
    } else if (isValidate != null && isValidate == false) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          backgroundColor: Theme.of(context).focusColor,
          content: Text('Ви не заповнили усі данні',
              style: Theme.of(context)
                  .textTheme
                  .displayMedium
                  ?.copyWith(fontWeight: FontWeight.bold))));
    } else if (isValidate == true) {
      Fluttertoast.showToast(
          msg: title,
          toastLength: Toast.LENGTH_SHORT,
          gravity: ToastGravity.CENTER,
          timeInSecForIosWeb: 1,
          backgroundColor: AppColors.primary,
          textColor: Theme.of(context).shadowColor,
          fontSize: 16.0);

      ScaffoldMessenger.of(context).clearSnackBars();
      Navigator.pop(context);
    }
  }
}
```

Рисунок 3.12 – Клас NotesManager

У разі виникнення помилки користувач отримує повідомлення про несподівану ситуацію через вбудований елемент інтерфейсу SnackBar. Це дозволяє своєчасно сигналізувати про проблему, надаючи текстову інформацію у відповідному стилі. Якщо ж виявляється, що деякі дані не заповнені, відображається інше повідомлення, яке закликає завершити введення інформації.

Успішна перевірка супроводжується відображенням toast-повідомлення, яке з'являється у центрі екрана. Це повідомлення, реалізоване за допомогою бібліотеки FlutterToast, надає користувачеві підтвердження про виконання операції. У цей момент додаток також очищує попередні сповіщення SnackBar і здійснює перехід до попереднього екрана через виклик функції навігації.

З точки зору візуального оформлення, повідомлення у додатку дотримуються загального стилю, визначеного темою програми. Це створює відчуття цілісності дизайну й допомагає користувачеві легше сприймати інтерфейс.

Завдяки такій архітектурі клас NotesManager забезпечує не лише функціональність, але й високий рівень зручності, що є важливим для користувацького досвіду. Це рішення може бути використане у багатьох сценаріях, таких як введення даних у формах, обробка подій чи підтвердження успішного виконання завдань.

Клас AppButtonStyles забезпечує уніфіковані стилі для кнопок у мобільному додатку, що дозволяє зберігати послідовність дизайну та швидко змінювати зовнішній вигляд кнопок за потреби. Завдяки цьому підходу можна досягти зручності в управлінні стилями та підвищити загальну естетику інтерфейсу (Рис. 3.13).

```dart
import 'package:flutter/material.dart';
import '../colors/app_colors.dart';

class AppButtonStyles {
  static ButtonStyle activeToggleStyle(BuildContext context) =>
      ElevatedButton.styleFrom(
          shape:
              RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
          elevation: 0,
          backgroundColor: AppColors.primary,
          side: BorderSide(
            color: AppColors.primary,
            width: 1.8,
          ));

  static ButtonStyle inactiveToggleStyle(BuildContext context) =>
      ElevatedButton.styleFrom(
          shape:
              RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
          elevation: 0,
          backgroundColor: Theme.of(context).disabledColor,
          side: const BorderSide(
            color: AppColors.grayDark,
            width: 0.8,
          ));

  static ButtonStyle mainButton(BuildContext context) => ElevatedButton.styleFrom(
    elevation: 0,
    backgroundColor: AppColors.primary,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(12),
      side: const BorderSide(width: 0.8, color: AppColors.grayDark),
    ),
  );
}
```

Рисунок 3.13 – Клас AppButtonStyles

Основні стилі визначені для трьох типів кнопок: активної та неактивної кнопки-перемикача (toggle button) і основної кнопки для взаємодії користувача. Кожен із цих стилів створюється за допомогою функції ElevatedButton.styleFrom, яка дозволяє налаштовувати такі параметри, як форма, колір, товщина межі та інші властивості.

Активна кнопка-перемикач виділяється насиченим кольором AppColors.primary і має товсту межу, що підкреслює її готовність до дії. У той же час неактивна кнопка має більш приглушений вигляд, використовуючи колір, визначений темою додатка для стану відключення (disabledColor), та тоншу темно-сіру межу.

Основна кнопка використовується для ключових взаємодій у додатку. Її дизайн включає помітне заокруглення, насичений фоновий колір і тонку темну межу, що забезпечує чіткість і візуальну привабливість.

Цей клас надає централізований спосіб управління стилями кнопок, що спрощує внесення змін у майбутньому та гарантує, що всі елементи інтерфейсу залишатимуться візуально узгодженими. Такий підхід підкреслює увагу до деталей у дизайні та забезпечує комфорт користувачів при взаємодії з додатком.

Клас AppColors представляє набір кольорових констант, які використовуються для стилізації інтерфейсу мобільного додатку (Рис. 3.14). Завдяки централізованому управлінню кольорами, цей підхід забезпечує узгодженість дизайну, спрощує підтримку коду та полегшує адаптацію інтерфейсу до змін у дизайні.

```dart
import 'package:flutter/material.dart';

class AppColors {
  static const bgDark = Color.fromRGBO(17, 16, 16, 1);
  static const bgLight = Color.fromRGBO(232, 232, 232, 1);

  static const secondaryColor = Color.fromRGBO(0, 123, 255, 1);
  static const accentColor = Color.fromRGBO(255, 193, 7, 1);
  static const successColor = Color.fromRGBO(40, 167, 69, 1);
  static const warningColor = Color.fromRGBO(255, 193, 7, 1);
  static const dangerColor = Color.fromRGBO(220, 53, 69, 1);

  static const primary = Color.fromRGBO(220, 87, 51, 1);
  static const primaryWithOpacity =
      Color.fromRGBO(220, 87, 51, 0.20784313725490197);
  static const primaryDark = Color.fromRGBO(196, 33, 12, 1);

  static const grayLight = Color.fromRGBO(248, 248, 248, 1);
  static const gray = Color.fromRGBO(108, 117, 125, 1);
  static const grayDark = Color.fromRGBO(52, 58, 64, 1);
}
```

Рисунок 3.14 – Клас AppColors

Кольорові константи охоплюють різні аспекти стилізації, зокрема фонові кольори, основні й акцентні кольори, а також кольори, пов’язані з певними

станами (успіх, попередження, небезпека). Наприклад, фонові кольори bgDark і bgLight використовуються для створення темного та світлого інтерфейсів, відповідно, що дозволяє реалізувати підтримку різних тем додатку.

Основний акцент у дизайні досягається за допомогою кольорів, таких як primary, який визначає ключовий візуальний стиль програми. Його темніша версія (primaryDark) і колір із прозорістю (primaryWithOpacity) забезпечують додаткову гнучкість для різних контекстів. Для інформування про стан або надання зворотного зв’язку застосовуються кольори, як-от successColor для успішних операцій, warningColor для попереджень і dangerColor для критичних помилок.

Крім того, сірі відтінки (grayLight, gray, grayDark) використовуються для допоміжних елементів, таких як межі, текст чи іконки, що забезпечує візуальну ієрархію і зручність сприйняття.

Централізований підхід до визначення кольорів у AppColors підвищує модульність коду і полегшує його масштабування. Завдяки цьому додаток може легко адаптуватися до змін у дизайні, зберігаючи цілісність і гармонійність інтерфейсу.

Клас AppTextStyles забезпечує визначення текстових стилів, що використовуються в мобільному додатку, для підтримки узгодженого та естетичного оформлення текстових елементів у різних темах (Рис. 3.15). Завдяки цьому підходу можна легко адаптувати вигляд тексту до світлого або темного режиму додатку, що підвищує зручність користувацького досвіду.

У класі реалізовані дві основні текстові теми: lightTextTheme і darkTextTheme. Вони містять стилі для різних типів тексту, таких як заголовки та дисплейні тексти. Кожен стиль налаштовується відповідно до певних параметрів: розміру шрифту, ваги (насиченості), кольору та інших характеристик.

lightTextTheme: Призначена для світлої теми інтерфейсу, де основним кольором тексту виступає чорний (Colors.black). Заголовки, такі як titleLarge і titleMedium, мають більший розмір шрифту та середню вагу (fontWeight: FontWeight.w600), що підкреслює їхню важливість. Дисплейні тексти

(displayLarge, displayMedium, displaySmall) мають менший розмір шрифту та жирний стиль (fontWeight: FontWeight.bold), що забезпечує акцентованість допоміжних елементів.

darkTextTheme: Ця тема розроблена для темної версії інтерфейсу, де основним кольором тексту є білий (Colors.white). Як і в світлій темі, заголовки й дисплейні тексти відповідають тій самій ієрархії, але адаптовані для гармонійного сприйняття на темному фоні.

```dart
import 'package:flutter/material.dart';

class AppTextStyles {
  static const lightTextTheme = TextTheme(
    titleLarge: TextStyle(
        fontSize: 32, fontWeight: FontWeight.w600, color: Colors.black),
    titleMedium: TextStyle(
        fontSize: 24, fontWeight: FontWeight.w600, color: Colors.black),
    titleSmall: TextStyle(
        fontSize: 18, fontWeight: FontWeight.w600, color: Colors.black),
    displayLarge: TextStyle(
        fontSize: 16, fontWeight: FontWeight.bold, color: Colors.black),
    displayMedium: TextStyle(
        fontSize: 14, fontWeight: FontWeight.bold, color: Colors.black),
    displaySmall: TextStyle(
        fontSize: 12, fontWeight: FontWeight.bold, color: Colors.black),
  );

  static const darkTextTheme = TextTheme(
    titleLarge: TextStyle(
        fontSize: 32, fontWeight: FontWeight.w600, color: Colors.white),
    titleMedium: TextStyle(
        fontSize: 24, fontWeight: FontWeight.w600, color: Colors.white),
    titleSmall: TextStyle(
        fontSize: 18, fontWeight: FontWeight.w600, color: Colors.white),
    displayLarge: TextStyle(
        fontSize: 16, fontWeight: FontWeight.bold, color: Colors.white),
    displayMedium: TextStyle(
        fontSize: 14, fontWeight: FontWeight.bold, color: Colors.white),
    displaySmall: TextStyle(
        fontSize: 12, fontWeight: FontWeight.bold, color: Colors.white),
  );
}
```

Рисунок 3.15 – Клас AppTextStyles

Таке централізоване управління текстовими стилями значно спрощує підтримку коду. Зміни в дизайні можна легко впровадити шляхом редагування

одного класу, що гарантує узгодженість текстових елементів у всьому додатку. Крім того, використання окремих текстових тем сприяє гнучкому налаштуванню інтерфейсу, що відповідає сучасним вимогам до дизайну мобільних додатків.

Залишковий код реалізації інтерфейсу із застосуванням платформи Flutter приведено у Додатку А.

3.4 Висновок до третього розділу

Під час розроблення інтерфейсу мобільного застосунку було обрано та використано сучасні програмні інструменти, що забезпечують високу якість дизайну та простоту використання. Для створення дизайну застосунку було обрано інструменти, які могли б інтегрувати найкращі практики UI/UX-дизайну та відповідати вимогам мобільної платформи.

Дизайн інтерфейсу був ретельно опрацьований, щоб створити привабливу, інтуїтивно зрозумілу і функціональну систему. Усі елементи були гармонійно інтегровані, щоб забезпечити простоту використання застосунку та ефективність взаємодії з користувачем. Завдяки продуманій роботі дизайн має чітку структуру зі зручним доступом до функцій.

Реалізація мобільного інтерфейсу також включала в себе практичне втілення рішень, розроблених у середовищі розробки. Цей процес включав у себе технічну інтеграцію елементів дизайну, налаштування анімації та перевірку коректної роботи всіх елементів інтерфейсу. У результаті було створено повноцінний мобільний додаток, здатний задовольнити потреби цільових користувачів.

ВИСНОВКИ

Розробка сучасних мобільних застосунків починається з вивчення переваг і потреб користувачів. Моніторинг допомагає визначити поточні вимоги до мобільних рішень.

Особлива увага приділяється вивченню цільової аудиторії, щоб врахувати звички, цілі та очікування від програми. На цьому етапі також слід визначити найкращий підхід до дизайну інтерфейсу користувача та UX, щоб забезпечити простоту використання.

Аналіз існуючих рішень для мобільного інтерфейсу допоміг визначити ефективні способи інтеграції в нові продукти.

Етап проектування передбачає використання принципів, які забезпечують функціональність, естетичність та інтуїтивність інтерфейсу. Основою роботи стала розробка структури мобільного застосунку, що включає модулі, екрани та взаємодію між ними. Створення вайрфреймів дозволяє візуалізувати макет інтерфейсу, включаючи розташування елементів і основні сценарії використання. Крім того, розроблено детальні шляхи користувача (User flow), щоб забезпечити логічний і комфортний досвід навігації.

Для розробки інтерфейсів були використані сучасні програмні засоби, що дозволяють створювати естетичні та функціональні рішення. Особливу увагу було приділено дизайну, що враховує як стиль, так і адаптацію до різних пристроїв.

Реалізація інтерфейсу включала створення інтерактивних елементів, анімації та динамічних компонентів для покращення взаємодії з користувачем.

Таким чином, процес створення мобільного інтерфейсу включає в себе детальний аналіз, ретельний дизайн і технічну реалізацію. Комплексний підхід дозволяє нам розробляти продукти, що відповідають потребам користувачів і стандартам сучасних технологій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке UI та UX дизайн?. *IT курси у Тернополі з працевлаштуванням | Академія IT STEP*. URL: https://te.itstep.org/blog/ui-and-ux-design.
2. REDSTONE :: Що таке UX / UI-дизайн. *REDSTONE*. URL: https://redstone.agency/blog/shcho-take-ux-ui-dysayn/.
3. What is the Value Proposition Canvas? - B2B International. *B2B International*. URL: https://www.b2binternational.com/research/methods/faq/what-is-the-value-proposition-canvas/.
4. Value Proposition Canvas. *Grow your Brand with RWDC | Delivering Purposeful Design*. URL: https://rwdc.sg/2020/04/27/value-proposition-canvas/.
5. Maltsev A. V. MyCar - Apps on Google Play. *Android Apps on Google Play*. URL: https://play.google.com/store/apps/details?id=com.mobile_solutions.mycar&hl=ua.
6. Sygic. Fuelio: gas log & gas prices - Apps on Google Play. *Android Apps on Google Play*. URL: https://play.google.com/store/apps/details?id=com.kajda.fuelio&hl=ua
7. Sygic a. s. Fuelio - fuel log, MPG. *AppÂ Store*. URL: https://apps.apple.com/ua/app/fuelio-fuel-log-mpg/id1487753318?l=ua
8. DataLife Engine > Версия для печати > Проектування інтерфейсів: поняття, основні принципи, особливості графічного інтерфейсу, етапи, розвиток та поради фахівців. *Новини високих технологій*. URL: https://hi-news.pp.ua/kompyuteri/print:page,1,13441-proektuvannya-nterfeysv-ponyattya-osnovn-principi-osoblivost-grafchnogo-nterfeysu-etapi-rozvitok-ta-poradi-fahvcv.html.
9. Проектування мобільних застосунків: етапи та підводні камені | Wezom. *IT-компания полного цикла разработки программных продуктов*

WEZOM - Киев, Украина. URL: https://wezom.com.ua/ua/blog/proektirovanie-mobilnogo-prilozheniya.

10. Не соромно запитати: що таке вайрфрейм, мокап і прототип. *SKVOT / СКВОТ – онлайн-курси про рекламу, кіно та мистецтво | SKVOT*. URL: https://skvot.io/uk/blog/ne-soromno-zapitati-shcho-take-vayrfreym-mokap-i-prototip.
11. Bohdana Muzyka. User Flows. Як ця техніка допомагає в роботі над проєктами. *dou.ua*. URL: https://dou.ua/lenta/columns/user-flows/#:~:text=User%20Flow%20–%20це%20шлях,%20який,рахунку,%20замовлення%20доставки%20тощо.
12. Як покращити користувацький досвід за допомогою UX/UI дизайну. *InProject - IT компанія, яка створює неймовірні digital продукти*. URL: https://u.to/IyURIQ.
13. Figma: The Collaborative Interface Design Tool. *Figma*. URL: https://www.figma.com/.
14. Design the incredible URL: https://www.adobe.com/ua/products/xd.html.
15. Учасники проектів Вікімедіа. GIMP – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/GIMP.
16. Contributors to Wikimedia projects. Flutter (software) - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Flutter_(software).
17. Шапошнікова О., Бондар В., Савенко О. Застосунок для моніторингу автомобіля: сучасне рішення для контролю витрат, обслуговування та аналітики. *«Процеси цифровізації екосистем»: зб. Наукових праць за матеріалами міжнародної наукової конференції. Харків, ХНАДУ.* 2004. С. 11–14.

ДОДАТОК А.
ПРОГРАМНИЙ КОД

app_button_styles.dart

```
class AppButtonStyles {
  static ButtonStyle activeToggleStyle(BuildContext context) =>
      ElevatedButton.styleFrom(
          shape:
              RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
          elevation: 0,
          backgroundColor: AppColors.primary,
          side: BorderSide(
            color: AppColors.primary,
            width: 1.8,
          ));

  static ButtonStyle inactiveToggleStyle(BuildContext context) =>
      ElevatedButton.styleFrom(
          shape:
              RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
          elevation: 0,
          backgroundColor: Theme.of(context).disabledColor,
          side: const BorderSide(
            color: AppColors.grayDark,
            width: 0.8,
          ));

  static ButtonStyle mainButton(BuildContext context) => ElevatedButton.styleFrom(
```

```
    elevation: 0,
    backgroundColor: AppColors.primary,
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(12),
      side: const BorderSide(width: 0.8, color: AppColors.grayDark),
    ),
  );
}
```

app_colors.dart

```
import 'package:flutter/material.dart';
class AppColors {
  static const bgDark = Color.fromRGBO(17, 16, 16, 1);
  static const bgLight = Color.fromRGBO(232, 232, 232, 1);

  static const secondaryColor = Color.fromRGBO(0, 123, 255, 1);
  static const accentColor = Color.fromRGBO(255, 193, 7, 1);
  static const successColor = Color.fromRGBO(40, 167, 69, 1);
  static const warningColor = Color.fromRGBO(255, 193, 7, 1);
  static const dangerColor = Color.fromRGBO(220, 53, 69, 1);
  static const primary = Color.fromRGBO(220, 87, 51, 1);
  static const primaryWithOpacity =
      Color.fromRGBO(220, 87, 51, 0.20784313725490197);
  static const primaryDark = Color.fromRGBO(196, 33, 12, 1);
  static const grayLight = Color.fromRGBO(248, 248, 248, 1);
  static const gray = Color.fromRGBO(108, 117, 125, 1);
  static const grayDark = Color.fromRGBO(52, 58, 64, 1);
}
```

dimens.dart

```
class AppDimens {
  static const rippleSize = 24.0;
  static const appBarHeight = 54.0;
  static const iconSize = 22.0;
}
```

app_text_styles.dart

```
import 'package:flutter/material.dart';
class AppTextStyles {
  static const lightTextTheme = TextTheme(
    titleLarge: TextStyle(
        fontSize: 32, fontWeight: FontWeight.w600, color: Colors.black),
    titleMedium: TextStyle(
        fontSize: 24, fontWeight: FontWeight.w600, color: Colors.black),
    titleSmall: TextStyle(
        fontSize: 18, fontWeight: FontWeight.w600, color: Colors.black),
    displayLarge: TextStyle(
        fontSize: 16, fontWeight: FontWeight.bold, color: Colors.black),
    displayMedium: TextStyle(
        fontSize: 14, fontWeight: FontWeight.bold, color: Colors.black),
    displaySmall: TextStyle(
        fontSize: 12, fontWeight: FontWeight.bold, color: Colors.black),
  );
  static const darkTextTheme = TextTheme(
    titleLarge: TextStyle(
        fontSize: 32, fontWeight: FontWeight.w600, color: Colors.white),
```

```
    titleMedium: TextStyle(
        fontSize: 24, fontWeight: FontWeight.w600, color: Colors.white),
    titleSmall: TextStyle(
        fontSize: 18, fontWeight: FontWeight.w600, color: Colors.white),
    displayLarge: TextStyle(
        fontSize: 16, fontWeight: FontWeight.bold, color: Colors.white),
    displayMedium: TextStyle(
        fontSize: 14, fontWeight: FontWeight.bold, color: Colors.white),
    displaySmall: TextStyle(
        fontSize: 12, fontWeight: FontWeight.bold, color: Colors.white),
  );
}
```

app_theme.dart

```
import 'package:car_car/ui/styles/text_styles/app_text_styles.dart';
import 'package:flutter/material.dart';

import '../colors/app_colors.dart';

const mainFont = "Montserrat";
final darkTheme = ThemeData(
    textTheme: AppTextStyles.darkTextTheme,
    useMaterial3: true,
    scaffoldBackgroundColor: AppColors.bgDark,
    colorScheme: ColorScheme.fromSeed(
      brightness: Brightness.dark,
      primary: AppColors.primaryDark,
```

```
      secondary: AppColors.secondaryColor,
      seedColor: AppColors.grayLight,
    ),
    fontFamily: mainFont,
    elevatedButtonTheme: ElevatedButtonThemeData(
      style: ElevatedButton.styleFrom(
        foregroundColor: AppColors.grayLight,
        surfaceTintColor: AppColors.grayLight,
      ),
    ),
    iconTheme: const IconThemeData(color: AppColors.grayLight),
    bottomNavigationBarTheme: const BottomNavigationBarThemeData(
        backgroundColor: AppColors.grayDark));
final lightTheme = ThemeData(
  textTheme: AppTextStyles.lightTextTheme,
  useMaterial3: true,
  scaffoldBackgroundColor: AppColors.bgLight,
  colorScheme: ColorScheme.fromSeed(
    brightness: Brightness.light,
    primary: AppColors.primary,
    secondary: AppColors.grayDark,
    seedColor: AppColors.grayDark,
  ),
  fontFamily: mainFont,
  elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
      foregroundColor: AppColors.grayDark,
      surfaceTintColor: AppColors.grayDark,
```

```
    ),
  ),
  iconTheme: const IconThemeData(color: AppColors.grayDark),
  bottomNavigationBarTheme:
      const BottomNavigationBarThemeData(backgroundColor: AppColors.grayLight),
);
```

button_delete.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_slidable/flutter_slidable.dart';
class ButtonDelete extends StatelessWidget {
  const ButtonDelete({super.key, required this.onTap});

  final Function onTap;
  @override
  Widget build(BuildContext context) {
    final styles = Theme.of(context).textTheme.displayLarge;
    return SlidableAction(
      onPressed: (context) {
        showDialog(
            context: context,
            builder: (context) {
              return Center(
                  child: Container(
                      width: 200,
                      height: 100,
                      decoration: BoxDecoration(
                          color: const Color.fromARGB(255, 103, 103, 103),
```

```
    borderRadius: BorderRadius.circular(10)),
child: Padding(
  padding: const EdgeInsets.all(8.0),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text("Are you sure?", style: styles),
      const SizedBox(
        height: 5,
      ),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          TextButton(
              onPressed: () {
                onTap();
                Navigator.pop(context);
              },
              child: Text("Yes", style: styles)),
          TextButton(
              onPressed: () => Navigator.pop(context),
              child: Text("No", style: styles))
        ],
      )
    ],
  ),
)));
```

```
            });
      },
      backgroundColor: Colors.transparent,
      foregroundColor: Theme.of(context).unselectedWidgetColor,
      icon: Icons.delete,
      spacing: 2,
      borderRadius: BorderRadius.circular(12),
    );
  }
}
```

main_tile.dart

```
import 'package:car_car/bloc/note_cubit/note_cubit.dart';
import 'package:car_car/ui/widgets/custom_slidable_action_buttons/button_delete.dart';
import
'package:car_car/ui/widgets/custom_slidable_action_buttons/button_editing.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_slidable/flutter_slidable.dart';
import '../styles/colors/app_colors.dart';
class MainTile extends StatelessWidget {
  final String title;
  final String? trailingText;
  final IconData? leadingIcon;
  final Color backgroundColor;
  final VoidCallback? onTap;
  final String id;
  const MainTile({
```

```
  super.key,
  required this.title,
  this.trailingText,
  this.leadingIcon,
  required this.backgroundColor,
  required this.id,
  this.onTap,
});
@override
Widget build(BuildContext context) {
  callBackTap() {
    BlocProvider.of<NoteCubit>(context).deleteNote(id);
  }
  return Slidable(
    endActionPane: ActionPane(motion: const ScrollMotion(), children: [
      const ButtonEditing(),
      ButtonDelete(
        onTap: callBackTap,
      )
    ]),
    child: Material(
      borderRadius: BorderRadius.circular(12),
      color: Colors.transparent,
      child: Card(
        elevation: 0,
        color: backgroundColor,
```

```
margin: const EdgeInsets.symmetric(horizontal: 12),
shape: RoundedRectangleBorder(
  borderRadius: BorderRadius.circular(12),
  side: const BorderSide(
    width: 1,
    color: AppColors.bgDark,
  ),
),
child: ListTile(
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(12),
  ),
  onTap: () => onTap,
  leading: Icon(
    leadingIcon,
    color: Theme.of(context).iconTheme.color,
  ),
  title: Text(
    title,
    maxLines: 1,
    overflow: TextOverflow.ellipsis,
    style: Theme.of(context)
        .textTheme
        .displayLarge
        ?.copyWith(color: Colors.black),
  ),
  tileColor: Colors.transparent,
  trailing: trailingText != null
```

```
              ? Text(
                  trailingText ?? '',
                  style: Theme.of(context)
                      .textTheme
                      .displayLarge
                      ?.copyWith(color: Colors.black),
                )
              : null,
          ),
        ),
      ),
    );
  }
}
```

modal_window_reminder.dart

```
import 'package:car_car/data/enums/reminders.dart';
import 'package:flutter/material.dart';

import '../features/reminder/screens/create_reminder.dart';
import '../styles/colors/app_colors.dart';

class ModalWindowReminder extends StatelessWidget {
  const ModalWindowReminder({
    super.key,
    required this.listReminderType,
    required this.listReminderTypeIcons,
  });
```

```
final List<ReminderType> listReminderType;
final List<String> listReminderTypeIcons;
@override
Widget build(BuildContext context) {
  return Center(
      child: AnimatedOpacity(
    opacity: 1,
    duration: const Duration(milliseconds: 10000),
    child: Container(
      alignment: Alignment.center,
      width: 350,
      height: 460,
      decoration: const BoxDecoration(
          color: Color.fromARGB(255, 67, 67, 67),
          borderRadius: BorderRadius.all(Radius.circular(10))),
      child: Padding(
        padding: const EdgeInsets.all(15.0),
        child: Column(
          children: [
            const Text(
              "Select Reminder Type",
            ),
            const SizedBox(
              height: 15,
            ),
            Expanded(
              child: ListView.separated(
```

```
itemBuilder: (context, index) {
  final currentValue =
      convertFromEnumToValue(listReminderType[index]);
  return GestureDetector(
    onTap: () => Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => CreateReminderScreen(
                reminderType: currentValue,
              ))),
    child: Container(
        height: 50,
        alignment: Alignment.center,
        decoration: BoxDecoration(
            color: const Color.fromARGB(255, 152, 152, 152),
            borderRadius: BorderRadius.circular(10),
            border: Border.all(
              color: Colors.amber,
            )),
        child: Row(
          children: [
            const SizedBox(
              width: 10,
            ),
            Image.asset(listReminderTypeIcons[index]),
            const SizedBox(
              width: 10,
            ),
```

```
                    Text(
                      currentValue,
                      style: Theme.of(context)
                          .textTheme
                          .headlineLarge
                          ?.copyWith(
                              color: AppColors.bgDark,
                              fontWeight: FontWeight.w600),
                    ),
                  ],
                )),
              );
            },
            separatorBuilder: (context, index) => const Divider(),
            itemCount: listReminderType.length,
          ),
        )
      ],
    ),
  ),
));
}
convertFromEnumToValue(ReminderType value) {
  switch (value) {
    case ReminderType.docs:
      return "Documents";
```

```
      case ReminderType.oil:
        return "Oil";
      case ReminderType.wash:
        return "Wash";
      case ReminderType.part:
        return "Part";
      case ReminderType.techCheck:
        return "Tech Check";
      case ReminderType.other:
        return "Other";
    }
  }
}
```

onboarding_items_bottom.dart

```
import 'package:flutter/material.dart';

import '../styles/colors/app_colors.dart';

class OnboardingCapsuleItem extends StatelessWidget {
  const OnboardingCapsuleItem({super.key});
  @override
  Widget build(BuildContext context) {
    return Container(
        width: 32,
        height: 4,
        decoration: const BoxDecoration(
```

```
          color: AppColors.primary,
          borderRadius: BorderRadius.all(Radius.circular(32))));
  }
}

class TonalityItem extends StatelessWidget {
  final String whichTonalityType;

  const TonalityItem({super.key, required this.whichTonalityType});

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Container(
          width: 22,
          height: 22,
          decoration: BoxDecoration(
              border: Border.all(color: AppColors.grayLight),
              color: AppColors.gray,
              borderRadius: const BorderRadius.all(Radius.circular(50))),
          child: Padding(
              padding: const EdgeInsets.fromLTRB(6, 0, 6, 2),
              child: Text(
                whichTonalityType,
                //TODO Get from theme
                //style: AppTextStyles.songsAuthorStyle
              ))),
```

```
    );
  }
}

class OnboardingDotItem extends StatelessWidget {
  final bool isChecked;
  const OnboardingDotItem({super.key, required this.isChecked});

  @override
  Widget build(BuildContext context) {
    return isChecked
        ? const OnboardingCapsuleItem()
        : Container(
            height: 5,
            width: 5,
            decoration: BoxDecoration(
              shape: BoxShape.circle,
              color: Theme.of(context).disabledColor,
            ));
  }
}

onboarding_next_button.dart
import 'package:flutter/material.dart';

import '../../main_icons.dart';
import '../styles/colors/app_colors.dart';
import '../styles/dimens/dimens.dart';
```

```
class OnboardingNextButton extends StatelessWidget {
  final VoidCallback onPressed;

  const OnboardingNextButton({super.key, required this.onPressed});
  @override
  Widget build(BuildContext context) {
    return Align(
        alignment: Alignment.bottomRight,
        child: Padding(
            padding: const EdgeInsets.only(right: 16, bottom: 40),
            child: ElevatedButton(
              onPressed: () {
                onPressed();
              },
              style: ElevatedButton.styleFrom(
                  shape: const CircleBorder(),
                  backgroundColor: AppColors.primary,
                  fixedSize: const Size(64, 64),
                  alignment: Alignment.center,
                  padding: const EdgeInsets.only(right: 1)),
              child: const Icon(
                MainIcons.icon_right_arrow,
                size: AppDimens.iconSize,
              ),
            )));
  }
}
```

ДОДАТОК Б.

ТЕКСТ СТАТТІ

УДК 004.4

ЗАСТОСУНОК ДЛЯ МОНІТОРИНГУ АВТОМОБІЛЯ: СУЧАСНЕ РІШЕННЯ ДЛЯ КОНТРОЛЮ ВИТРАТ, ОБСЛУГОВУВАННЯ ТА АНАЛІТИКИ

Володимир Бондар
Магістр, Харківський національний автомобільно-дорожній університет
Олег Савенко
Магістр, Харківський національний автомобільно-дорожній університет
Керівник: Олена ШАПОШНІКОВА
К.т.н., доцент, Харківський національний автомобільно-дорожній університет

Анотація. Пропонується розробка мобільного застосунку, який матиме чітку бізнес-логіку, продумані функції та зручний інтерфейс, що зробить застосунок ефективним рішенням для управління автомобільними витратами. Для цього було визначено потреби користувача з використанням інструменту Value Proposition Canvas та сформульовано ціннісну пропозицію.

Ключові слова: Веб-застосунок, Value Proposition Canvas (VPC), технічне обслуговування автомобіля

У сучасному світі, де технології розвиваються надзвичайно швидко, автомобілі стають не лише засобом пересування, а й важливим елементом повсякденного життя. Оскільки автомобілі є складними системами, що потребують постійного технічного обслуговування та контролю, виникає потреба у простих, зручних інструментах для моніторингу їхнього стану. Саме тому розробка застосунку для моніторингу стану автомобіля набуває актуальності для сучасних автовласників.

Такий застосунок може надавати власнику авто вичерпну інформацію про терміни зносу деталей, що дозволяє не лише забезпечити належний догляд за транспортним засобом, але й попередити несподівані поломки, підвищити безпеку на дорозі та продовжити термін експлуатації автомобіля.

Наразі на ринку існує певна кількість мобільних застосунків для моніторингу технічного стану автомобілів та відстеження витрат на їх експлуатацію [1-3]. Проте багато з них не враховують або частково враховують такі потреби власників, як:

– відстеження витрат на паливо;

– документування технічного обслуговування;
– управління запасними частинами;
– тощо.

Враховуючи це, пропонується розробити зручний, компактний мобільний застосунок, який має задовільнити базовий набір потреб користувачів – власників автомобілів.

Для визначення потреб користувачів та для того, щоб сформулювати унікальну ціннісну пропозицію було використано інструмент Value Proposition Canvas (Рис.1), який дав можливість:

1. Визначити потреби клієнтів: що саме клієнти хочуть вирішити за допомогою програмного продукту, маючи на увазі не тільки проблеми, з якими вони стикаються, а й очікувані вигоди від нього.

2. Сформулювати цінність продукту виділити, які конкретно аспекти продукту задовольняють потреби клієнтів, щоб зосередитися на розробці максимально корисних функцій [4, 5].

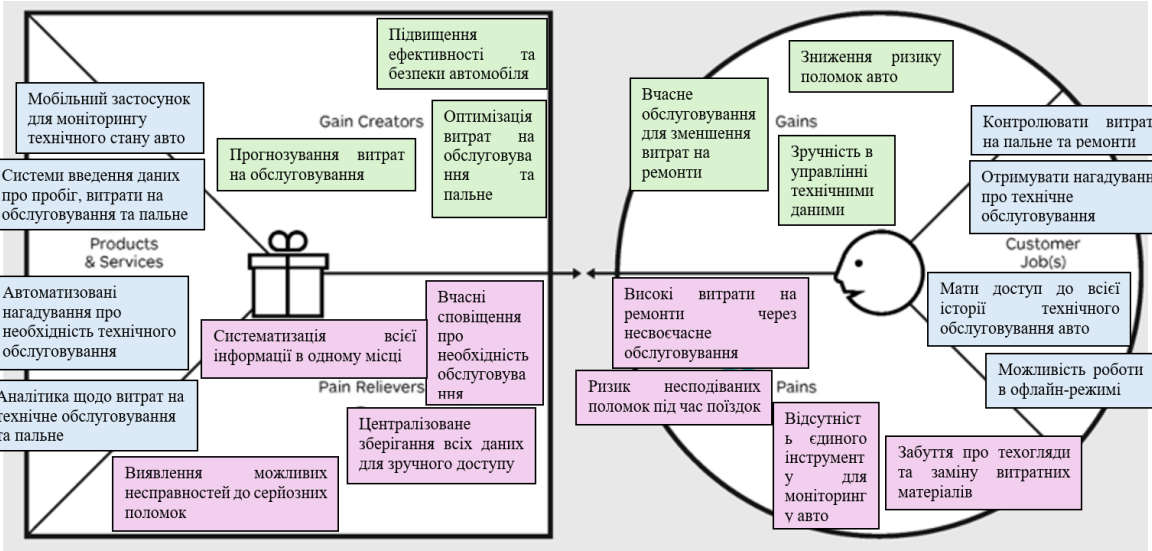


Рисунок 1 - Value Proposition Canvas для застосунку моніторингу технічного стану автомобіля

Відповідно до рисунку 1 основні функції застосунку, які забезпечують його корисність та зручність наступні.

1. Запис витрат на паливо:

– введення даних - користувач може вводити інформацію про заправки, включаючи дату, обсяг пального, вартість, місце заправки та тип пального;

автоматичний розрахунок - застосунок розраховує середню витрату пального на 100 км, що дозволяє користувачам швидко оцінити ефективність використання пального;

ПРОДОВЖЕННЯ ДОДАТКА Б

– графічний аналіз - відображення витрат у вигляді графіків та діаграм за різні періоди (тиждень, місяць, рік) для виявлення тенденцій.

2. Відстеження пробігу:

– запис пробігу - користувач може вносити дані про пробіг на початку та в кінці поїздки, що дозволяє автоматично розраховувати загальний пробіг;

– історія пробігу - можливість переглядати історію пробігу та отримувати нагадування про необхідність технічного обслуговування на основі пробігу.

3. Запис технічного обслуговування:

– ведення журналу - можливість фіксувати дати та види проведеного технічного обслуговування, а також пов’язані витрати;

– нагадування - інтеграція функцій нагадування про заплановані роботи, що допомагає уникнути пропусків.

4. Керування запчастинами:

– введення даних - користувач може записувати інформацію про куплені запчастини, їх ціни та дати придбання;

– аналіз витрат - застосунок може аналізувати витрати на запчастини та обслуговування, допомагаючи користувачам краще планувати бюджет.

5. Аналітика та звіти:

– графічні звіти - візуалізація даних про витрати, пробіг та технічне обслуговування, що дозволяє користувачам легко розуміти свою автомобільну історію;

– експорт даних - можливість експорту даних у різні формати (CSV, PDF) для подальшого аналізу або для надання фінансових звітів.

Очевидно, що мобільний застосунок для відстеження технічного стану автомобіля є важливим інструментом. За умов його використання водії можуть значно зменшити фінансові витрати на обслуговування своїх автомобілів, отримувати нагадування про заплановані роботи та покращувати загальний технічний стан своїх транспортних засобів. Розробка та подальше вдосконалення застосунку сприятиме розвитку нових можливостей для власників автомобілів, дозволяючи їм краще керувати своїм автомобільним досвідом.

ПРОДОВЖЕННЯ ДОДАТКА Б

ДОДАТОК В.
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет

Механічний факультет

Кафедра комп'ютерних наук і інформаційних систем

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ДИПЛОМНОЇ РОБОТИ
магістра

РОЗРОБЛЕННЯ ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ ВІДСТЕЖЕННЯ ТЕХНІЧНОГО СТАНУ АВТОМОБІЛЯ

Завідувачка кафедри канд. техн. наук, доцентка — Ганна ПЛЄХОВА
Нормоконтролер, доктор філософії — Андрій ЛЕБЕДИНСЬКИЙ
Керівник канд. техн. наук, доцент — Олена ШАПОШНІКОВА
Студент гр. МК-61-23 — Володимир БОНДАР

Харків – 2024

ПОСТАНОВКА ЗАДАЧІ

Мета роботи - аналіз особливостей проектування та розробка мобільного застосунку.

Об'єкт дослідження – інтерфейс мобільного застосунку.

Предмет дослідження – інформаційні технології для реалізації інтерфейсу мобільного застосунку.

Бажані результати – розробити інтерфейс мобільного застосунку, що дозволить користувачам мати можливість вести статистику та витрати на автомобіль

Продовження додатку В

UI ТА UX ДИЗАЙН ДЛЯ УСПІХУ ПРОЕКТУ

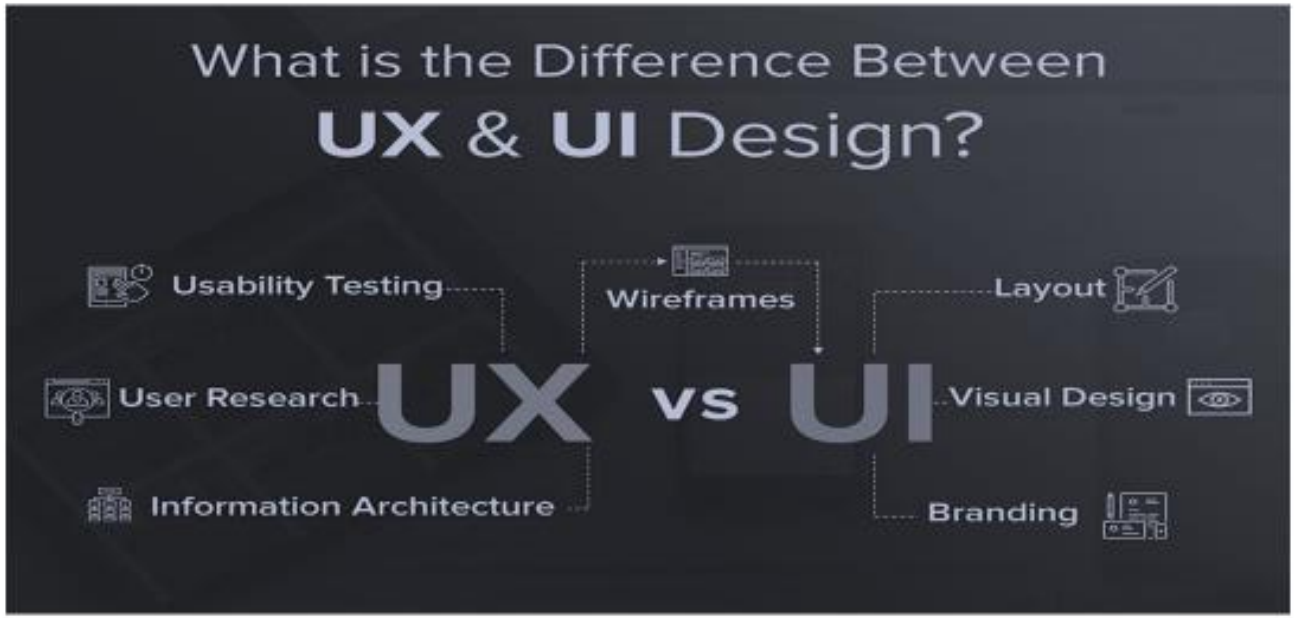


Рисунок Б.1 - UI-дизайн і UX-дизайн

3/14

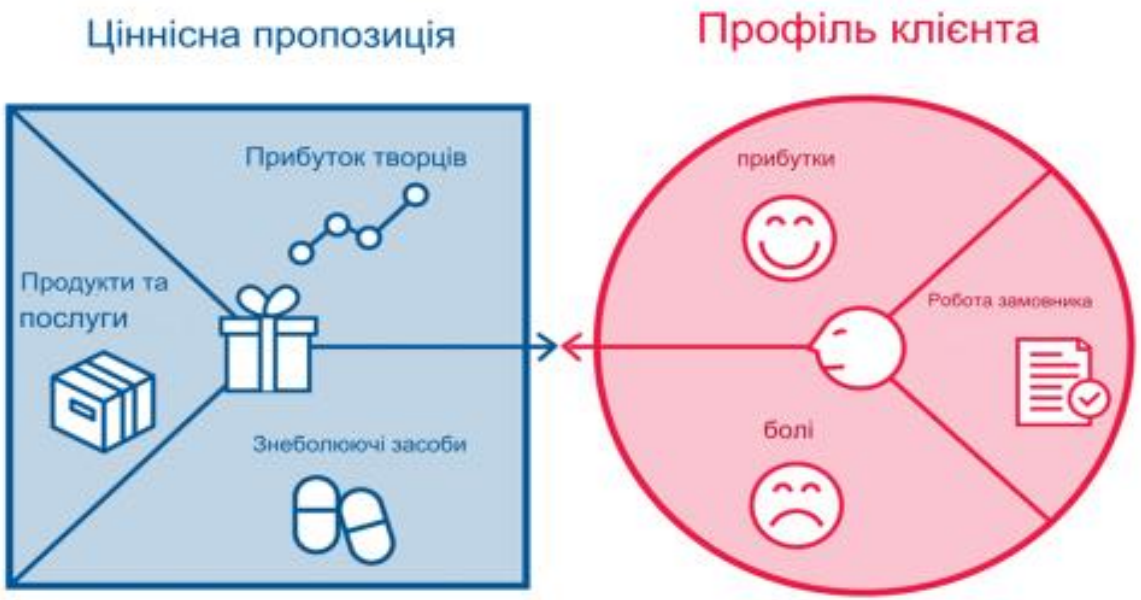


Рисунок Б.2 - Інструмент Value Proposition Canvas

Продовження додатку В

5/14

ЦІННІСТНА ПРОПОЗИЦІЯ ЗАСТОСУНКУ

Профіль клієнта	**Карта цінісної пропозиції**
Завдання клієнта	**Продукти та сервіси**
- Стежити за технічним станом авто.	- Мобільний додаток для моніторингу технічного стану авто.
- Контролювати витрати на пальне та ремонти.	- Системи введення даних про пробіг, витрати на обслуговування та пальне.
- Отримувати нагадування про технічне обслуговування.	- Автоматизовані нагадування про необхідність технічного обслуговування.
- Мати доступ до всієї історії технічного обслуговування авто.	- Аналітика щодо витрат на технічне обслуговування та пальне.
- Можливість роботи в офлайн-режимі.	
Болі клієнта	**Те, що зменшує біль**
- Високі витрати на ремонти через несвоєчасне обслуговування.	- Вчасні сповіщення про необхідність обслуговування.
- Забуття про техогляди та заміну витратних матеріалів.	- Централізоване зберігання всіх даних для зручного доступу.
- Ризик несподіваних поломок під час поїздок.	- Виявлення можливих несправностей до серйозних поломок.
- Відсутність єдиного інструменту для моніторингу авто.	- Систематизація всієї інформації в одному місці.
Вигоди	**Те, що створює вигоди**
- Зниження ризику поломок авто.	- Підвищення ефективності та безпеки автомобіля.
- Вчасне обслуговування для зменшення витрат на ремонти.	- Прогнозування витрат на обслуговування.
- Зручність в управлінні технічними даними.	- Оптимізація витрат на обслуговування та пальне.

АНАЛІЗ ІНТЕРФЕЙСІВ ІСНУЮЧИХ МОБІЛЬНИХ ЗАСТОСУНКІВ

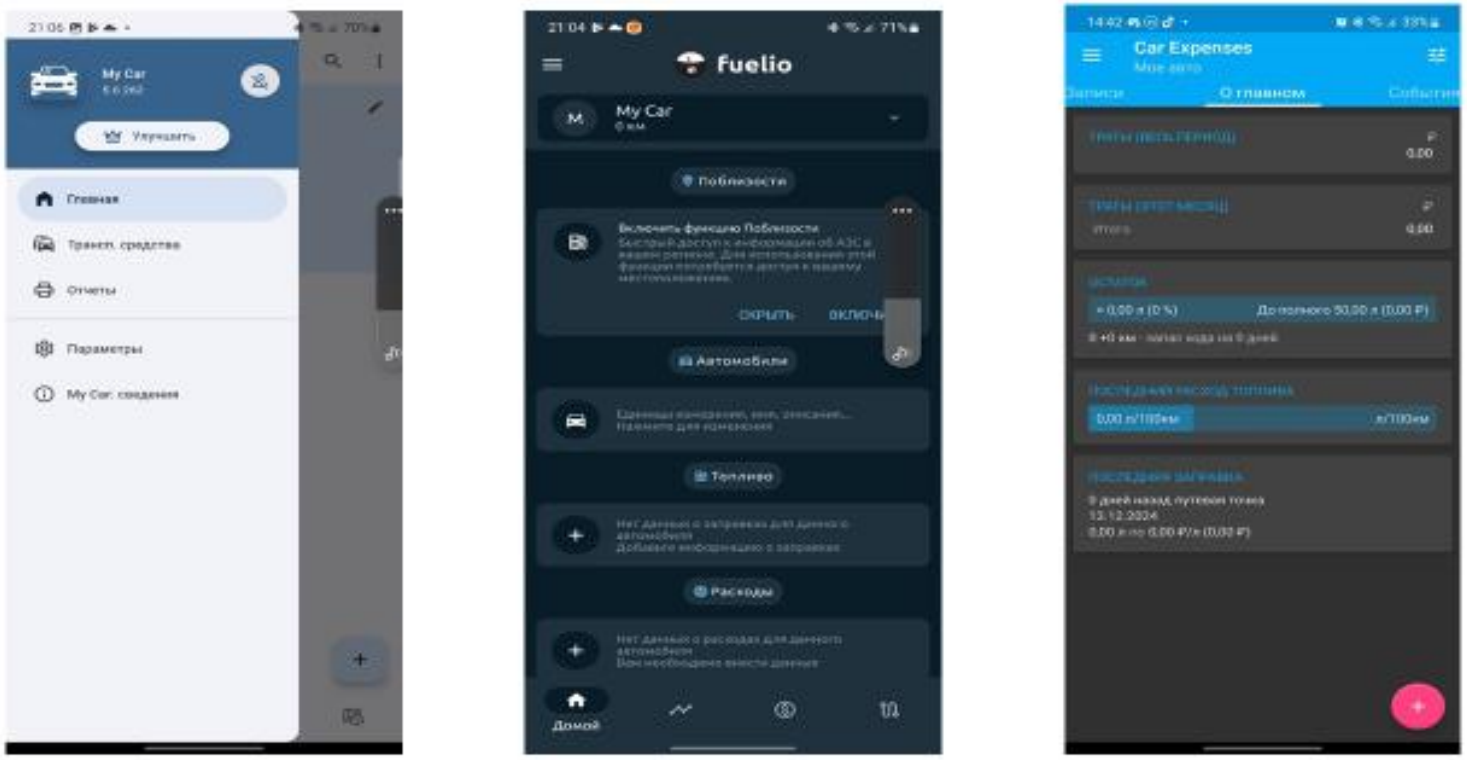


Рисунок Б.3 – Інтерфейси: "MyCar", "Fuelio", "Car Expenses"

6/14

Продовження додатку В

СТРУКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ

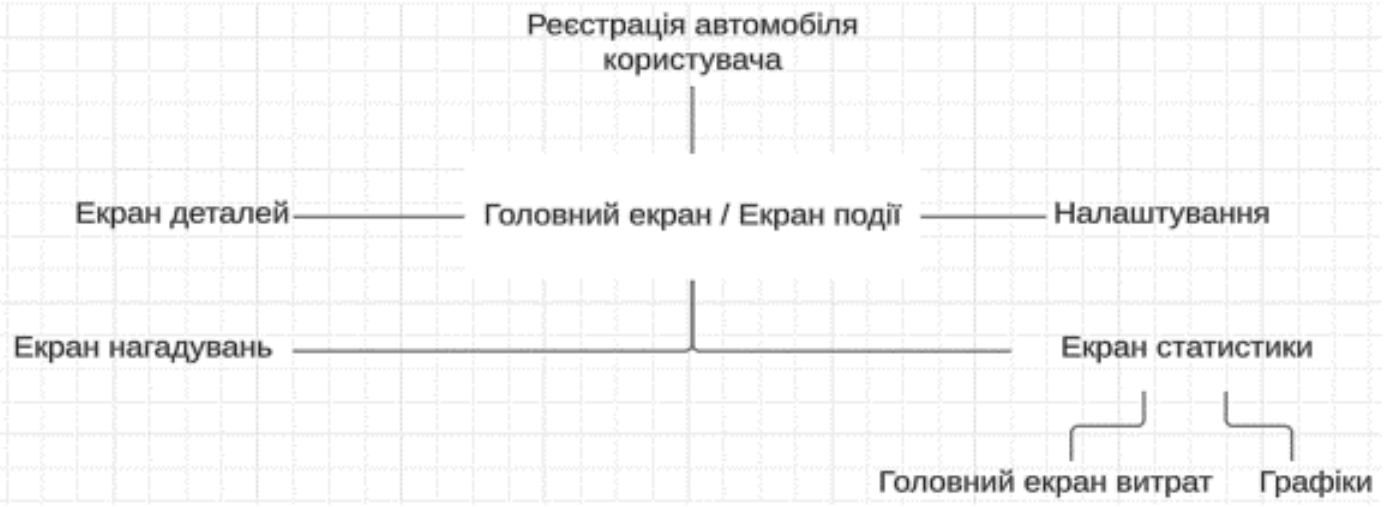


Рисунок Б.4 – Структура екранів інформаційної системи

РОЗРОБКА ВАЙРФЕЙМІВ

Принципи вайрфреймів:

- ✓ Консистентність
- ✓ Гнучкість
- ✓ Взаємодія
- ✓ Простота
- ✓ Чіткість
- ✓ Функціональність

8/14

Продовження додатку В

РОЗРОБКА ВАЙРФЕЙМІВ

Рисунок Б.5 – Вайрфрейми: Екран реєстрація автомобіля, Екран нагадувань

9/14

Продовження додатку В

РОЗРОБКА ВАЙРФЕЙМІВ

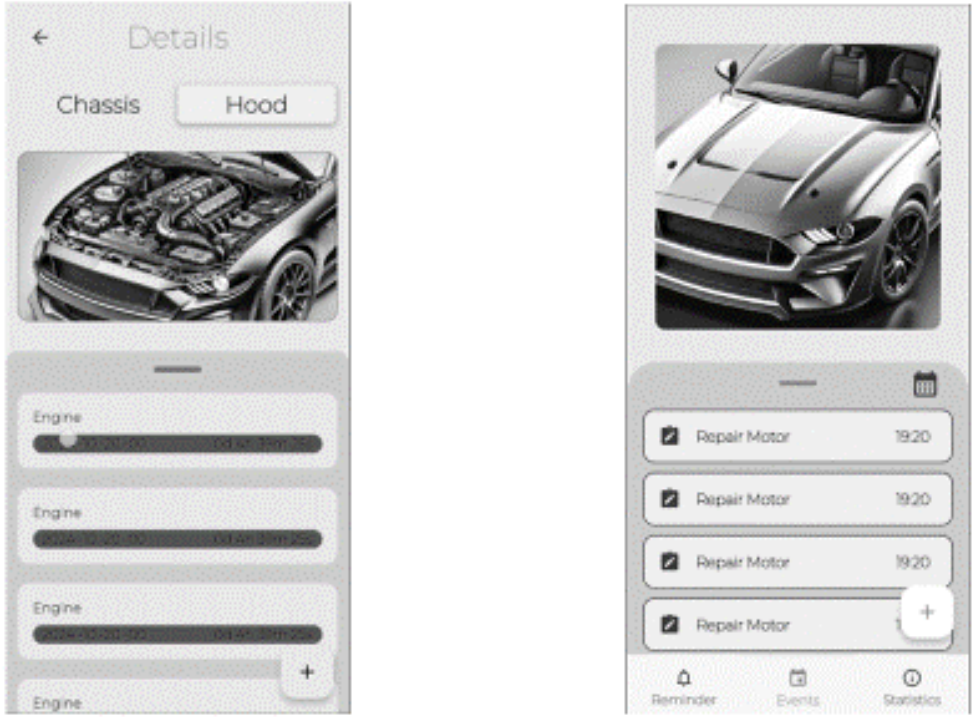


Рисунок Б.6 - Вайрфрейми: Екран деталей, Екран подій.

ШЛЯХИ КОРИСТУВАЧА ІНФОРМАЦІЙНОЇ СИСТЕМИ

Рисунок Б.7 – User flow мобільного застосунку

РОЗРОБКА ІНТЕРФЕЙСУ МОБІЛЬНОГО ЗАСТОСУНКУ

Рисунок Б.8 – Реєстрація автомобіля; Екран події з автомобілєм

12/14

Продовження додатку В

а) Без заповнених даних

б) З заповненими даними

а) Діагональна

б) Кругова

Рисунок Б.9 – Екрані: Додавання нової події; Перегляд статистики

ВИСНОВКИ

1. Було досліджено підходи до проєктування та розробку користувацького інтерфейсу мобільного застосунку.
2. Досліджено цільову аудиторію та аналіз інтерфейсів існуючих мобільних застосунків.
3. Була створена структура майбутньої системи та розроблено користувацьку візію.
4. Описані та проведені такі етапи проектування як: створення user flow, прототипування, створення вайрфреймів.
5. Було реалізований макет застосунку для подальшої розробки.

Продовження додатку В