Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Інститут Комп'ютерних наук та інформаційних технологій
(повна назва)
Кафедра Інформаційних технологій видавничої справи
(повна назва)

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Роль «Варіантів» та «Компонентів» в процесі проектування інтерфейсу користувача»

"The role of "Variables" and "Components" in the user interface design process"

Студентка гр. ВП-41 _____ Марія-Софія БЄЛИШЕВА
(шифр групи) (підпис) (ім'я, прізвище)

Керівник роботи _____ к.т.н., доц. Микола ЛОГОЙДА
(підпис) (ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ІТВС _____ к.т.н., доц. Наталія КУСТРА
(підпис) (ім'я, прізвище)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут Комп'ютерних наук та інформаційних технологій

(номер, назва)

Кафедра Інформаційних технологій видавничої справи

(номер, назва)

Спеціальність 186 "Видавництво та поліграфія"

(номер, назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« 01» травня 2025 р.

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ**
**студентки групи ВП-41**

Бєлишева Марія-Софія Юріївна

1. Тема роботи: «Роль «Варіантів» та «Компонентів» в процесі проектування інтерфейсу користувача», затверджена наказом по університету №1588-4-08 від 02.05.2025
2. Термін подання закінченої роботи 9 червня 2025
3. Вихідні дані до роботи:

3.1. Технічне завдання;

3.2. Методології проведення досліджень користувачів та проблематики - дерево проблем, дерево цілей, тестування зручності;

3.3. Середовище розробки дизайну Figma.

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити): Вступ, основна частина (характеристика об'єкту розроблення, постановка задачі, аналіз літературних джерел; системний аналіз, проектування та розробка; експериментальна частина, висновок), додатки.
5. Перелік графічного матеріалу: дерево цілей, дизайн-система, інформаційна архітектура, дизайн застосунку
6. Консультанти із зазначенням розділів роботи кожного

| **Найменування розділу** | **Консультант** (посада, прізвище, ім'я, по батькові) | **Позначка консультанта про виконання розділу** | |
|---|---|---|---|
| | | (підпис) | (дата) |
| Вступ | к.т.н., доц. Микола Логойда | | |
| Розділ 1 | к.т.н., доц. Микола Логойда | | |
| Розділ 2 | к.т.н., доц. Микола Логойда | | |
| Розділ 3 | к.т.н., доц. Микола Логойда | | |
| Висновки | к.т.н., доц. Микола Логойда | | |

7. Дата видачі завдання xx.xx.2025

Керівник _____ к.т.н., доц. Микола ЛОГОЙДА
(підпис) (ім'я, прізвище)

Завдання прийняв до виконання _____
(підпис студента)

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Затвердження теми роботи | | |
| 2 | Представлення характеристики об'єкту, 1 розділу роботи науковому керівникові | | |
| 3 | Побудова дерев проблем та цілей | | |
| 4 | Опис етапів дизайну | | |
| 5 | Представлення 2 розділу науковому керівникові | | |
| 6 | Тестування дизайну на зручність | | |
| 7 | Представлення 3 розділу науковому керівникові | | |
| 8 | Оформлення бакалаврської роботи згідно вимог | | |
| 9 | Представлення звіту | | |
| 10 | Доопрацювання роботи | | |

Студентка гр. ВП-41 _____ Марія-Софія БЄЛИШЕВА
(шифр групи) (підпис) (ім'я, прізвище)

Керівник роботи _____ к.т.н., доц. Микола ЛОГОЙДА
(підпис) (ім'я, прізвище)

# АНОТАЦІЯ

У бакалаврській кваліфікаційній роботі розглянуто теоретичні основи використання змінних (анг. Variables) та компонентів (анг. Components) при створенні дизайн-систем, а також продемонстровано їх практичне застосування.

У першій частині роботи детально вивчено, як змінні та компоненти сприяють створенню ефективних і зручних інтерфейсів. Особливу увагу приділено інструментам для розробки дизайн-систем, зокрема програмі Figma та її можливостям у роботі з змінними (анг. Variables) та компонентами (анг. Components). Також проведено порівняння цих інструментів з іншими популярними програмами, такими як Sketch і Adobe XD.

Другий розділ роботи присвячений розробці структури і алгоритмів інтерфейсу користувача. Тут визначено основні етапи створення інтерфейсу для додатку, а також проведено аналіз вимог і цілей проекту. У практичній частині було розроблено додаток, в якому за допомогою дизайн-системи реалізовано інтерфейс, що забезпечує зручність у використанні.

Експериментальна частина роботи включає тестування зручності додатку, аналіз результатів тестів і впровадження рекомендацій для покращення досвіду користувачів. Також у роботі розраховано економічні витрати на розробку додатку, включаючи витрати на роботу команди та необхідні ресурси.

*Ключові слова: Змінні, Компоненти, інформаційна архітектура, дизайн-система, тестування зручності, інтерфейс, дослідження користувачів.*

*Пояснювальна записка складається зі вступу, трьох розділів, висновку, переліку посилань; містить 75 сторінок, 16 рисунків, та 1 додаток.*

*Список використаних джерел включає 21 бібліографічних найменування*

## ABSTRACT

The bachelor's thesis discusses the theoretical foundations of using variables and components in the creation of design systems and demonstrates their practical application.

In the first part of the paper, the authors study in detail how variables and components contribute to the creation of efficient and user-friendly interfaces. Particular attention is paid to tools for developing design systems, in particular, the Figma program and its capabilities in working with variables and components. A comparison of these tools with other popular programs, such as Sketch and Adobe XD, is also made.

The second section of the paper is devoted to the development of the structure and algorithms of the user interface. It defines the main stages of creating an interface for an application, as well as analyzes the requirements and goals of the project. In the practical part, an application was developed in which a user-friendly interface was implemented using a design system.

The experimental part of the work includes testing the usability of the application, analyzing the test results, and implementing recommendations to improve the user experience. The paper also calculates the economic costs of developing the application, including the costs of team work and the necessary resources.

*Keywords: Variables, Components, information architecture, design system, usability testing, interface, user research.*

*The explanatory note consists of an introduction, three chapters, conclusion, list of references; contains 75 pages, 16 figures, and 1 appendix.*
*The list of references includes 21 bibliographic titles*

# ЗМІСТ

АНОТАЦІЯ..... 5
ЗМІСТ..... 8
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ..... 10
ВСТУП..... 13
РОЗДІЛ 1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ РОЗРОБЛЕННЯ, ПОСТАНОВКА ЗАДАЧІ, АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ..... 15
1.1 Визначити, що таке Variables та Components у UX/UI дизайні.....15
1.2 Роль та значення варіантів і компонентів для створення дизайн-систем..... 16
1.3 Як Variables використовуються для зберігання повторюваних значень (кольори, шрифти, відступи тощо)..... 18
1.4 Роль Components як основних будівельних блоків для створення інтерфейсів.....20
1.5 Атомарний дизайн.....21
1.6 Як Variables та Components розвивалися у дизайні..... 22
1.8 Основні принципи використання Variables та Components у дизайн-системах..... 26
1.9 Консистентність у дизайні..... 27
1.10 Порівняння використання Variables та Components у Figma, Sketch і Adobe XD..... 29
1.11 Висновок..... 29
РОЗДІЛ 2. СИСТЕМНИЙ АНАЛІЗ, РОЗРОБЛЕННЯ СТРУКТУРИ, АЛГОРИТМІВ ТА ІНФОРМАЦІЙНОГО..... 30
2.1 Поняття та призначення дерева цілей..... 30
2.2 Принципи побудови дерева цілей.....31
2.3 Методи побудови дерева цілей..... 33
2.4 Поняття інформаційної архітектури.....36
2.5 Дерево проблем..... 39
2.6 Розробка та опис інтерфейсу користувача..... 41
2.6 Висновок..... 49
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА..... 50
3.1 Інструкція.....50
3.2 Правила використання дизайн-системи та токенів для дизайнерів.. 53
3.3 Тестування зручності..... 56
3.4 Рекомендації для майбутніх покращень на основі тестування.....59
3.5 Економічна частина.....62

3.5 Висновок..... 64
ВИСНОВОК.....66
СПИСОК ЛІТЕРАТУРИ.....68
ДОДАТОК А..... 70

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

UI (User Interface) - це користувацький інтерфейс. Він представляє собою сукупність елементів та способів взаємодії користувача (людини) з програмним забезпеченням, веб-сайтом або будь-якою іншою інформаційною системою, яка спрямована на забезпечення зручного та інтуїтивного досвіду.

UX (User Experience) - це те, що людина відчуває при користуванні продуктом, системою чи сервісом (послугою). Це стосується всіх аспектів взаємодії, від початку і до кінця, включаючи емоції, зручність та задоволеність користувача.

Variables - це іменовані значення, які використовуються для повторного застосування стилістичних параметрів, таких як кольори, шрифти, розміри, відступи тощо. Вони дозволяють централізовано керувати стилем інтерфейсу та забезпечують його узгодженість

Components - це багаторазово використовувані елементи інтерфейсу, які містять структуру, візуальне оформлення і, за потреби, взаємодію. При зміні головного компонента, всі його екземпляри оновлюються автоматично.

Style Guides - це ключовий інструмент, який містить шаблони та вказівки щодо використання компонентів зокрема шрифтів, типографіки, логотипу, кольорової палітри та зображень.

UI Kit - це набір графічних елементів інтерфейсу, виконаних в одному стилі й промальованих у всіх можливих станах (заблокованому, у фокусі, порожньому, заповненому правильно/з помилкою).

Design System - це набір стандартизованих компонентів, стилів, правил та інструментів, які використовуються для створення та підтримки продуктів з єдиним, послідовним дизайном та функціоналом.

Плагін - це додаткові інструменти та функції, які розширюють можливості програми. Вони розроблені як зовнішніми розробниками, так і самим Figma, і

інтегруються в додаток, щоб зробити роботу з дизайном більш гнучкою та продуктивною.

Інформаційна архітектура (Information architecture)- це основа, на якій будується структура будь-якого цифрового продукту: вебсайту, мобільного додатку чи онлайн-сервісу. У найширшому сенсі, IA - це процес організації, структурування та позначення інформаційного контенту таким чином, щоб користувачі могли інтуїтивно орієнтуватися в інтерфейсі, швидк знаходити потрібне та ефективно взаємодіяти з продуктом.

Карта сайту (Sitemap) - це структурований список або схема, що містить інформацію про всі сторінки, які доступні на вебсайті. Вона є важливим інструментом для навігації, що дозволяє користувачам і пошуковим системам швидко знайти потрібні сторінки сайту.

Ментальні карти (Mind maps) - це інструмент для організації та візуалізації інформації, що дозволяє відображати ідеї, концепції та взаємозв'язки між ними у вигляді схеми. Ментальні карти використовують ієрархічну структуру, де центральна ідея або проблема розміщується в центрі карти, а інші ідеї або підпункти розгалужуються від неї, формуючи гілки.

Керування шляхом користувача (User journey) - це концепція в UX/UI дизайні, яка описує весь досвід користувача, коли він взаємодіє з продуктом або послугою, починаючи від початкового контакту і до завершення взаємодії. Це візуалізація або карта, яка відображає ключові етапи, через які проходить користувач, а також емоційний відгук, який він переживає на кожному етапі.

Тестування дерева (Tree testing) - це метод дослідження в юзабіліті-тестуванні, який використовується для перевірки ефективності структури навігації на сайті або в додатку. Воно дозволяє оцінити, наскільки зручно і логічно організовані категорії, підкатегорії та інші елементи навігаційної системи. Цей метод допомагає виявити проблеми в інформаційній архітектурі ще до запуску продукту або на ранніх етапах його розробки.

# ВСТУП

**Тема роботи:** Роль «Варіантів» та «Компонентів» в процесі проектування інтерфейсу користувача.

**Мета:** Проаналізувати теоретичні й практичні засади використання Змінних (анг. Variables) та Компонентів (анг. Components) у створенні дизайну, визначити їхню роль та вплив на UX/UI дизайн, а також продемонструвати їх ефективне застосування.

**Предмет:** Предметом роботи є засади використання змінних та компонентів у створенні дизайну, та методи проектування інтерфейсу користувача.

**Актуальність:** У сучасному світі, де цифрові продукти відіграють ключову роль у житті людей, якісний UX/UI дизайн є вирішальним фактором успіху будь-якого бізнесу чи організації. Користувачі очікують інтуїтивно зрозумілі та естетично привабливі інтерфейси, які забезпечують зручність та ефективність використання.

В умовах постійного розвитку технологій та зростання вимог до цифрових продуктів, створення консистентних, масштабованих та ефективних дизайн-систем є важливим завданням для дизайнерів. Саме тому дослідження ролі Змінних (анг. Variables) та Компонентів (анг. Components) у створенні дизайну є надзвичайно актуальним.

**Об’єкт роботи:** мобільний гібридний застосунок, розроблений для пошуку музики за мелодією. Застосунок орієнтований на ефективну взаємодію користувачів з інтерфейсом, який забезпечує простоту та зручність у використанні на різних платформах.

**Для досягнення поставленої мети, слід виконати наступні завдання:**

1. **Проаналізувати теоретичні засади використання Variables та Components у створенні дизайну**

- Визначити, що таке Variables та Components у UX/UI дизайні

- Чому ці елементи важливі для створення дизайн-систем
- Як Variables використовуються для зберігання повторюваних значень (кольори, шрифти, відступи тощо)
- Роль Components як основних будівельних блоків для створення інтерфейсів
- Атомарний дизайн
- Як Variables та Components розвивалися у дизайні
- Розглянути сучасні інструменти Figma та її функціонал, пов’язаний із Variables та Components
- Сформулювати основні принципи використання Variables та Components у дизайн-системах

2. **Проаналізувати роль та вплив Variables та Components на UX/UI дизайн**
3. **Продемонструвати ефективне застосування Variables та Components на практичному прикладі дизайну застосунку для пошуку музики за мелодією.**
4. **Провести користувацьке тестування**
5. **Внести правки за результатами тестування**

# РОЗДІЛ 1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ РОЗРОБЛЕННЯ, ПОСТАНОВКА ЗАДАЧІ, АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

## 1.1 Визначити, що таке Variables та Components у UX/UI дизайні

У сучасному світі цифрових продуктів, UX/UI дизайн відіграє важливу роль у створенні естетичних та головне зручних для користувачів інтерфейсів. Для створення таких інтерфейсів дизайнери використовують різноманітні інструменти, серед яких особливо популярною є платформа Figma — сучасне середовище для розробки інтерактивних та масштабованих інтерфейсів. Два ключові поняття, які допомагають дизайнерам досягти успіху в цій області - це змінні (анг. Variables) та Компоненти (анг. Components).

У середовищі **Figma** змінні (анг. Variables) використовуються для збереження повторюваних значень, таких як кольори, типографіка, відступи, розміри тощо. Це дає змогу створювати гнучкі, масштабовані та легко керовані дизайн-системи. Наприклад, замість того щоб змінювати колір у кожному окремому елементі вручну, дизайнер може оновити значення змінної — і зміни автоматично застосуються до всіх пов'язаних об'єктів. Такий підхід значно прискорює редизайн і забезпечує узгодженість інтерфейсу.

Компоненти (анг. Components) у Figma — це повторно використовувані блоки інтерфейсу, що складаються з візуальних елементів і взаємодій. Вони слугують будівельними модулями, які можна багаторазово застосовувати в різних частинах макету. Наприклад, стандартна кнопка з заданими стилями (колір, розмір, тінь) створюється один раз як компонент і потім вставляється в інші екрани, що гарантує візуальну і функціональну консистентність [1].

Основною перевагою використання змінних (анг. Variables) є централізоване управління стилістичними параметрами. Це не лише полегшує підтримку великих проєктів, а й дає змогу швидко адаптувати дизайн до нових вимог, наприклад — змінити кольорову тему або типографіку в кілька кліків. Змінні також значно зменшують ризик помилок, пов'язаних з несистемним

використанням стилів, особливо в роботі над масштабними інтерфейсами або в команді [2].

Компоненти (анг. Components), у свою чергу, дозволяють багаторазово використовувати однакові елементи інтерфейсу без потреби створювати їх з нуля щоразу. Це підвищує швидкість роботи дизайнера, забезпечує візуальну узгодженість усіх екранів і спрощує впровадження змін: достатньо оновити головний компонент, і зміни автоматично застосуються до всіх його екземплярів. Компонентний підхід є основою для побудови масштабованих та ефективних дизайн-систем.

## 1.2 Роль та значення варіантів і компонентів для створення дизайн-систем

Дизайн-система — це набір правил, компонентів і інструментів, що допомагають швидко створювати нові продукти і підтримувати вже існуючі, при цьому зберігаючи якість продукту.

Тобто, якщо дизайнеру треба створити нову форму на сайті, — наприклад, для збору зворотнього зв’язку, — за наявності дизайн-системи йому не потрібно вигадувати зовнішній вигляд полей чи кнопок, їх станів, композиційної компоновки тощо. Вони просто беруть вже описані рішення з відповідної дизайн-системи та створюють на їх основі нову форму.

Такий підхід не лише дозволяє досягти візуальної цілісності, але й на додачу до цього значно пришвидшує роботу команди. І Змінні (анг. Variables) та компоненти (анг. Components) відіграють ключову роль у створенні ефективних та успішних дизайн-систем [3].

Змінні (анг. Variables) та компоненти (анг. Components) допомагають забезпечити візуальну та функціональну консистентність всього продукту показано приклад на (Рис.1.2.1) Використовуючи єдині значення змінних для кольорів, шрифтів та інших параметрів, дизайнери можуть бути впевнені, що всі елементи інтерфейсу виглядають та поводяться однаково на різних сторінках та платформах [4].

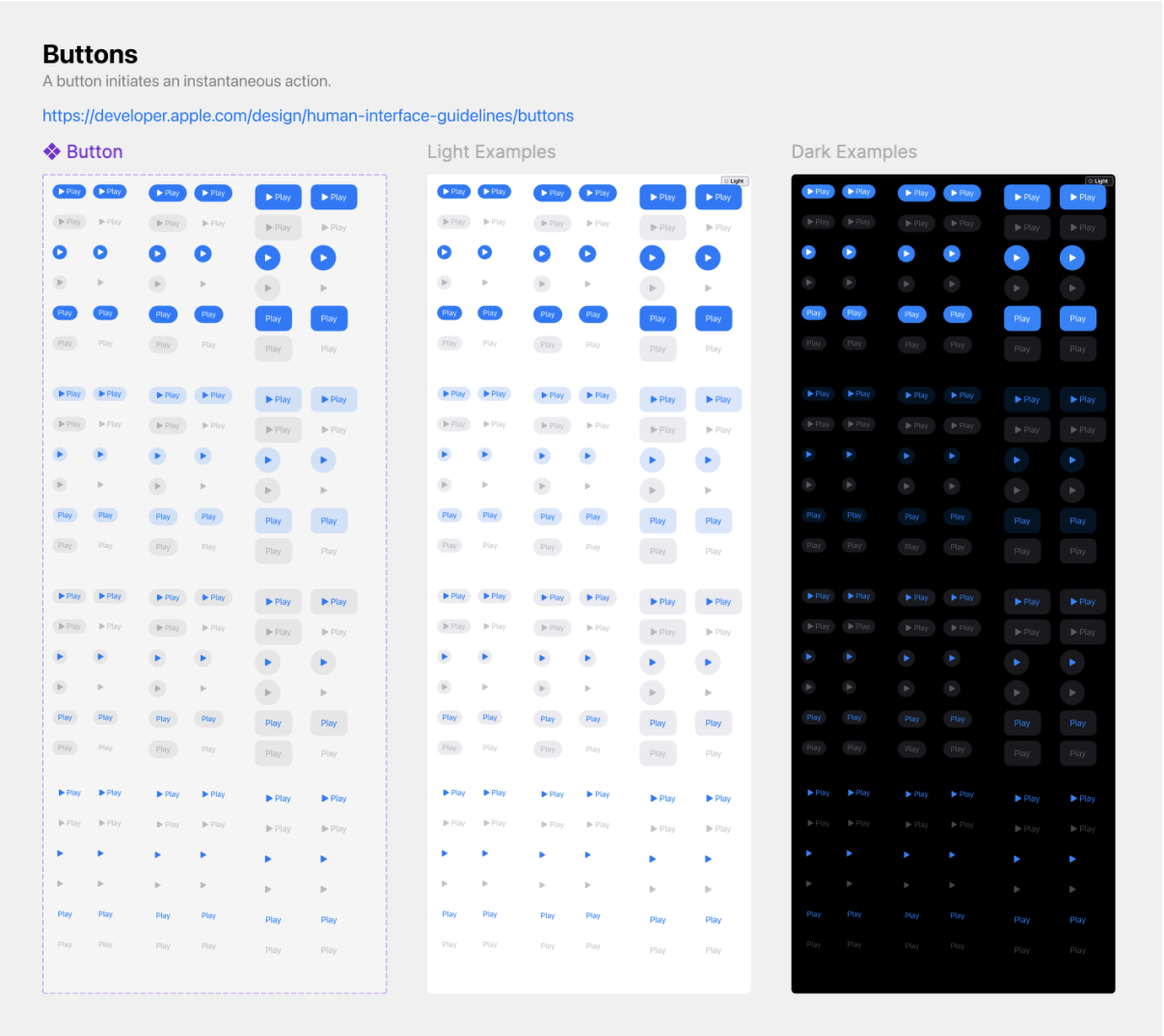


*Рис.1.2.1 (Приклад дизайн-системи)*

Зі зростанням проєкту, дизайн-система повинна бути здатною легко масштабуватися та адаптуватися до нових вимог. Змінні (анг. Variables) та компоненти (анг. Components) дозволяють додавати нові елементи без необхідності вручну змінювати кожен елемент. Змінюючи значення змінних або додаючи нові компоненти, дизайнери можуть швидко оновити весь інтерфейс [5].

Основною перевагою дизайн-систем є їхня здатність швидко тиражувати дизайн, використовуючи готові компоненти та елементи інтерфейсу. Команди можуть продовжувати використовувати ті самі елементи знову і знову,

зменшуючи необхідність винаходити велосипед і, таким чином, зменшуються ризики ненавмисної неузгодженості [6].

### 1.3 Як Variables використовуються для зберігання повторюваних значень (кольори, шрифти, відступи тощо)

Змінні (анг. Variables) відіграють ключову роль у створенні та підтримці консистентності дизайн-систем. Вони дозволяють зберігати та централізовано керувати значеннями, які часто використовуються в інтерфейсі, такими як кольори, шрифти, відступи та інші параметри стилю [7].

Ви - дизайнер продукту, працюєте над невеликим новим продуктом, не маючи існуючої дизайн системи. У вас є фірмовий колір — яскраво-синій (#3640F5). Ви використовуєте цей колір у різних місцях продукту, внутрішніх системах та на вашому веб-сайті: текст заголовка, основні фони кнопок, текст другорядних кнопок, текст посилань тощо.

Можливо, дизайнер обирає синій колір на око, який виглядає схожим, але насправді є іншим #4254F5. Це залишається непоміченим і потрапляє в код. Подібні речі відбуваються ще кілька разів і в результаті ви отримуєте п'ять різних відтінків синього і палітру, яка виглядає приблизно ось так (Рис.1.3.1).

Більшість дизайн систем включають базові палітри кольорів, на основі яких будуються всі інші. Оскільки ваша дизайн команда це не одна людина, а ваша дизайн система це багато компонентів, екранів та різних інтерфейсних тем, з часом ви побачите, що спроба додати нові кольори швидко перетворює всю систему у непослідовну та важко керовану.

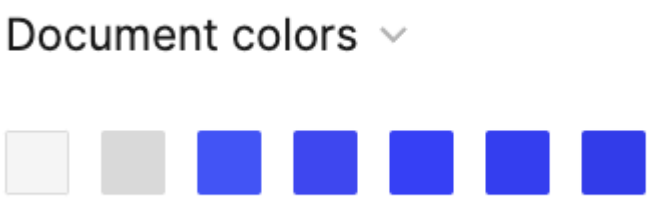


*Рис.1.3.1 (Document colours)*

Загальний підхід полягає в тому, щоб мати кілька базових відтінків, включаючи кольори вашого бренду та акцентні кольори, а також деякі відтінки для таких речей, як помилки, попередження, інформування про успіх, нейтральні кольори тощо. Кожен з цих відтінків матиме шкалу яскравості, позначену числом, яке вказує на те, наскільки він світліший або темніший. Наприклад, neutral-100 — дуже світло-сірий, neutral-500 — середньо-сірий, а neutral-900 — дуже темно-сірий. Деякі дизайн системи використовують шкалу від 0 до 100, це вже на ваш вибір. Варіанти можуть посилатися на конкретні значення, але вони також можуть посилатися на інші варіанти.

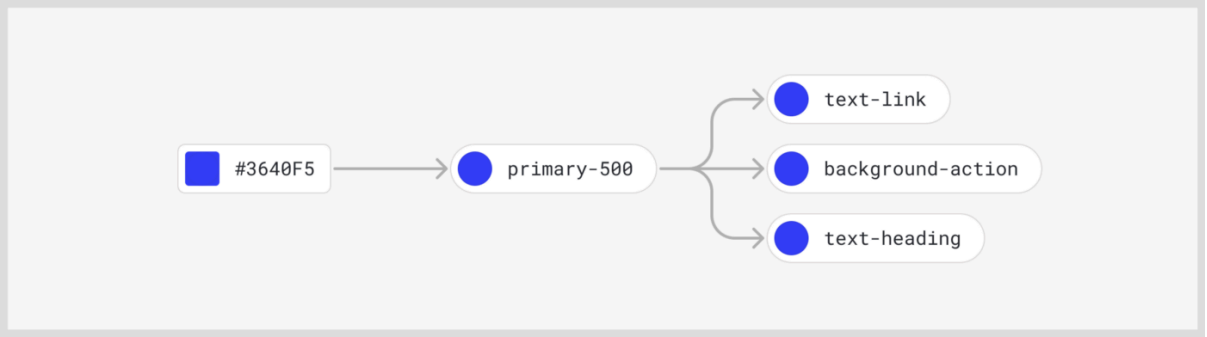


*Рис.1.3.2 (Варіанти кольорів)*

У наведеному вище прикладі (Рис.1.3.2) ми бачимо, що визначено той самий основний варіант, але потім три додаткові варіанти псевдоніми, один для текстових посилань, один для фону кнопок і один для тексту заголовка. Всі вони мають однакове значення синього кольору, тож навіщо ці дублікати?

Коли ваша дизайн система стає більш складною, скажімо, 20-30 кольорів, ви отримаєте 4 або 5 різних відтінків одного і того ж кольору, і потрапляєте в ситуацію, коли дизайнери або розробники не можуть запам'ятати в яких контекстах, які з них використовувати. Особливо це стосується різних відтінків сірого для фонів, рамок, тіней тощо, оскільки варіації можуть бути дуже незначними. Додавши зміст і контекст до імен, ви можете зробити використання кольорів дуже зрозумілим та передбачуваним для всіх членів команди.

## 1.4 Роль Components як основних будівельних блоків для створення інтерфейсів

Компоненти (анг. Components) - це інтерактивні будівельні блоки для створення користувацького інтерфейсу. Вони можуть бути організовані в категорії на основі їх призначення: дія, стримування, комунікація, навігація, вибір і введення тексту. Вони відіграють вирішальну роль у створенні сучасних інтерфейсів, забезпечуючи ефективність, консистентність та масштабованість дизайн-систем [8].

На основі аналізу існуючих підходів можна визначити основні ролі Компонентів (анг. Components) у процесі проєктування інтерфейсів:

Перше, це прискорення роботи над дизайном. Використання готових компонентів дозволяє дизайнерам та розробникам значно скоротити час на створення інтерфейсу. Замість того, щоб розробляти кожен елемент з нуля, вони можуть використовувати існуючі компоненти, налаштовуючи їх відповідно до потреб проекту [9].

Друге, це забезпечення єдиного стилю та поведінку елементів інтерфейсу на всіх сторінках та платформах. Використання однакових компонентів гарантує, що всі елементи виглядають та працюють однаково, що сприяє кращій зручності використання та покращує враження користувача.

Третє, завдяки централізованому управлінню компонентами, оновлення інтерфейсу стає значно простішим. Змінюючи компонент в одному місці, зміни автоматично відображаються у всіх його екземплярах, що значно зменшує зусилля, необхідні для внесення змін.

Четверта і одна з головних ролей, це масштабованість, іншими словами атомарний дизайн. Завдяки компонентам, дизайн-система легко адаптується до зростаючих потреб проєкту, дозволяючи додавати нові функціональні можливості без порушення існуючої структури.

Компоненти є основними будівельними блоками для створення сучасних інтерфейсів. Вони забезпечують ефективність, консистентність,

масштабованість та спрощують підтримку інтерфейсу. Використання компонентів є запорукою успіху будь-якого цифрового продукту [10].

### 1.5 Атомарний дизайн

Методологію атомарного дизайну сформулював автор книги «Atomic Design» Бред Фрост, використовуючи гасло «Створюйте системи, а не сторінки» [11].

Іншими словами, замість того, щоб проєктувати, наприклад сторінку інтерфейсу відразу цілком, логічніше збирати її з деталей. З найдрібніших елементів та формувати таким чином модульні сторінки та блоки, що при необхідності їх буде легко змінити в будь-якому місці та будь-якому обсязі, прибравши або замінивши потрібні елементи [12].

**Три етапів атомарного дизайну:**

1. **Атоми (Atoms)**

   Базові елементи, з яких складається інтерфейс. Їх неможливо розбити на більш дрібні елементи без втрати функціонального значення. Сюди входять: іконки, кнопки, поля введення. Як і у природі, атоми інтерфейсу неспроможні існувати окремо, вони мають сенс лише у тих чи інших поєднаннях один з одним. Слугують основою для створення молекули.

2. **Молекули (Molecules)**

   Найпростіші групи атомів, що функціонують разом як єдиний елемент. Такий елемент можна повністю вставляти, переміщати та копіювати в тих місцях інтерфейсу, в яких потрібна функція даного елементу. Молекули утворюють повноцінні блоки призначені для багаторазового застосування. Наприклад поле пошуку: складається з атома іконки та поля введення. Слугують основою для створення організму.

3. **Організми (Organisms)**

   Найвища група атомів та молекул, що становлять відносно складні секції інтерфейсу. Можуть складатися з однакових та/або різних типів молекул.

Наприклад, організм шапки сайту може складатися з різних молекул: лого, навігації, пошуку, блоку посилань на соціальні мережі. А організм картки товару складатиметься з однієї молекули, що багаторазово повторюється та містить, наприклад, фото, назву і ціну товару. Слугують основою для створення шаблонів [13].

**1.6 Як Variables та Components розвивалися у дизайні**

Розвиток змінних (анг. Variables) та компонентів (анг. Components) в дизайні є відображенням еволюції підходів до створення інтерфейсів, від простих макетів до складних дизайн-систем. Їхня роль змінювалась з часом, відображаючи зростаючі вимоги до консистентності, масштабованості та ефективності дизайн-процесів.

На ранніх етапах розвитку дизайну, креслення та макети були основними інструментами для візуалізації інтерфейсів. Дизайнери створювали статичні зображення, які відображали зовнішній вигляд та структуру майбутнього продукту. Про змінні (анг. Variables) та компоненти (анг. Components) як такі ще не йшлося, але вже тоді існувала потреба в стандартизації та уніфікації елементів дизайну.

З розвитком веб-дизайну, виникла потреба в більш системному підході до створення інтерфейсів. З'явилися посібники зі стилю (анг. Style Guides) та набори інтерфейсів (анг. UI Kits), які містили набори типових елементів інтерфейсу (кнопки, форми, іконки) та рекомендації щодо їх використання. Це були перші кроки до стандартизації та уніфікації елементів дизайну, хоча змінні (анг. Variables) та компоненти (анг. Components) як такі ще не були сформовані [14].

Проте, через деякий час стало зрозуміло, що посібники зі стилю (анг. Style Guides) та набори інтерфейсів (анг. UI Kits) недостатньо для забезпечення консистентності та масштабованості дизайну. Так з'явилися дизайн системи (анг. Design System) - комплексні системи, які включають в себе не тільки візуальні елементи, але й правила, принципи та процеси, що регулюють

створення та підтримку інтерфейсів. Дизайн системи (анг. Design System) стали логічним продовженням розвитку підходів до стандартизації та уніфікації дизайну, а змінні (анг. Variables) та компоненти (анг. Components) стали їх невід'ємними елементами. Оскільки вони дозволяють централізовано керувати параметрами стилю та забезпечувати консистентність на всіх рівнях інтерфейсу. З'явилися різні типи змінних, такі як змінні кольору (анг. color tokens), змінні шрифтів (анг. typography tokens), змінні відступів (анг. spacing tokens) та інші. Використання змінних (анг. Variables) дозволило дизайнерам легко змінювати параметри стилю, не редагуючи кожен елемент окремо, що значно підвищило ефективність роботи.

Розвиток Figma сприяє популяризації змінних (анг. Variables) та компонентів (анг. Components). Ці інструменти надають потужні можливості для створення, управління та використання змінних та компонентів, що значно спрощує процес розробки дизайн-систем.

## 1.7 Розглянути сучасні інструменти Figma та її функціонал, пов’язаний із Variables та Components

Figma є одним з найпопулярніших інструментів для UX/UI дизайну, який надає потужні можливості для роботи з змінними (анг. Variables) та компонентами (анг. Components). Її функціонал дозволяє дизайнерам створювати дизайн-системи, забезпечуючи консистентність, масштабованість та ефективність дизайн-процесу.

Figma підтримує різні типи змінних, такі як кольори, шрифти, числа та інші, середовище дозволяє створювати колекції змінних, що спрощує їх організацію та управління.

Figma, має такий функціонал для змінних, як:

- Створення та редагування змінних (Figma дозволяє легко створювати нові змінні, змінювати їх значення та властивості.)
- Організація змінних (Змінні можуть бути згруповані в колекції для кращої організації та управління.)

- Використання змінних в компонентах (анг. Components) (Змінні можуть використовуватися для параметрів компонентів, що забезпечує консистентність та гнучкість дизайну.)
- Режими змінних (Figma підтримує різні режими змінних, що дозволяє створювати адаптивні дизайн-системи, які враховують різні контексти (наприклад, темна/світла тема))

Figma надає широкі можливості для роботи з компонентами (анг. Components), включаючи створення, редагування, організацію та використання. Компоненти (анг. Components) можуть бути простими, як говорилось раніше атомами (наприклад, кнопка) або складними, організмами (наприклад, картка) [15].

Figma, має такий функціонал для компонентів, як:

- Створення та редагування компонентів (анг. Components) (Figma дозволяє легко створювати нові компоненти, змінювати їх структуру та властивості.)
- Організація компонентів (анг. Components) (Компоненти можуть бути організовані в бібліотеки для кращого управління та повторного використання.)
- Змінні (анг. Variables) (Figma підтримує створення різних варіантів компонентів з різними властивостями, що дозволяє створювати більш гнучкі та адаптивні інтерфейси.)
- Властивості компонентів (Figma дозволяє додавати властивості до компонентів, що дозволяє налаштовувати їх зовнішній вигляд та поведінку.)

Figma має велику кількість плагінів, які полегшують роботу з змінними (анг. Variables) та компонентами (анг. Components). Деякі з них дозволяють автоматизувати рутинні завдання, спростити процес створення дизайн-систем та покращити взаємодію між дизайнерами та розробниками.

Я виділила декілька, якими користуюсь:

- Rename It (Плагін з легкістю переіменовує велику кількість елементів, що є зручним для додовання семантичних варіантів)

- Styler (Легко перетворює велику кількість елементів в стилі)
- Styles to Variables (Плагін з в декілька кліків перетворює стилі в змінні)
- Styles & Variables Organizer (Чудовий плагін, який підтримує в чистоті наші дизайн проєкти)

Figma є потужним інструментом для UX/UI дизайну, який надає широкі можливості для роботи з змінними (анг. Variables) та компонентами (анг. Components). Її функціонал дозволяє дизайнерам створювати складні дизайн-системи, забезпечуючи консистентність, масштабованість та ефективність дизайн-процесу. Використання Figma в поєднанні з плагінами дозволяє значно спростити та автоматизувати багато завдань, пов'язаних з роботою з змінними (анг. Variables) та компонентами (анг. Components).

Figma є одним із найпопулярніших інструментів для створення UX/UI дизайну завдяки своїй зручності, гнучкості та можливостям для командної роботи. Це хмарна платформа, яка не потребує встановлення програмного забезпечення, що особливо цінно для команд, які працюють віддалено або у змішаних середовищах. Інтерфейс Figma інтуїтивно зрозумілий, що робить її доступною як для початківців, так і для досвідчених дизайнерів [16].

На відміну від інших інструментів, таких як Sketch або Adobe XD, Figma працює напряму в браузері, не вимагає потужного комп’ютера та автоматично зберігає всі зміни. Вона має активну спільноту, розвинену екосистему плагінів і регулярні оновлення, що робить її гнучким та інноваційним інструментом для сучасного дизайну інтерфейсів.

## 1.8 Основні принципи використання Variables та Components у дизайн-системах

Ефективне використання змінних (анг. Variables) та компонентів (анг. Components) є запорукою успіху будь-якого дизайну. Для досягнення максимальної користі від цих інструментів, необхідно дотримуватися певних принципів.

**Принцип гнучкості**

- Адаптивність: змінні (анг. Variables) та компоненти (анг. Components) повинні бути здатними адаптуватися до різних контекстів використання (різні розміри екранів, теми оформлення, стани елементів тощо).
- Редагування: компоненти (анг. Components) повинні мати можливість редагування за допомогою властивостей та параметрів, що дозволяє створювати різні варіанти використання.

**Принцип повторного використання**

- Універсальність: компоненти (анг. Components) повинні бути розроблені таким чином, щоб їх можна було використовувати багаторазово в різних частинах інтерфейсу та вирішувати різні завдання.
- Модульність: компоненти (анг. Components) повинні бути модульними та складатися з менших, незалежних елементів, що дозволяє комбінувати їх для створення складніших структур.

Дотримання цих принципів дозволить створити ефективну та зручну дизайн-систему, яка буде служити фундаментом для створення якісних та консистентних інтерфейсів. Дизайн-система - це живий організм, який потребує постійного розвитку та підтримки [17].

## 1.9 Консистентність у дизайні

Принцип консистентності (узгодженості) є фундаментальним у проєктуванні зручних, зрозумілих і ефективних користувацьких інтерфейсів. Його суть полягає в тому, що елементи інтерфейсу повинні поводитись передбачувано і виглядати однаково в подібних контекстах. Коли користувач бачить знайомі елементи або взаємодіє з повторюваними структурами, йому не потрібно щоразу вчитись заново — це скорочує когнітивне навантаження і підвищує ефективність використання продукту.

У дизайні виділяють кілька рівнів консистентності:

- Візуальна консистентність — однакове використання кольорів, шрифтів, розмірів, відступів, іконографіки. Наприклад, усі кнопки "Основна дія" мають однаковий колір і розмір по всьому додатку.
- Функціональна консистентність — передбачувана поведінка елементів: однакові дії викликають однаковий результат. Наприклад, свайп вліво завжди відкриває додаткове меню, незалежно від розділу.
- Змістова консистентність — застосування однакових термінів, стилю написання та логіки в усіх частинах інтерфейсу. Наприклад, повідомлення про помилку завжди оформлені за єдиним шаблоном.
- Структурна консистентність — однакова ієрархія контенту, розташування навігації та логіка переходів між екранами.

Сучасні цифрові продукти активно використовують дизайн-системи - стандартизовані набори стилів, компонентів, патернів, які забезпечують узгодженість між усіма частинами інтерфейсу. Використання компонентів (анг. Components) і змінних (анг. Variables) у Figma дозволяє досягти високого рівня консистентності навіть у складних проєктах.

Наприклад:

- Компонент (анг. Components) кнопки створюється один раз із заданими кольорами, шрифтами, відступами та станами (normal, hover, disabled). Після цього він використовується в усіх місцях, де потрібна кнопка, без необхідності кожного разу налаштовувати стиль заново.
- Змінні (анг. Variables) дозволяють централізовано змінювати кольори, шрифти чи інші значення. При зміні значення змінної — наприклад, кольору фону — ця зміна застосовується автоматично до всіх пов’язаних елементів, що гарантує візуальну цілісність.

Переваги дотримання принципу консистентності:

- Зрозумілість — користувачі швидше орієнтуються в інтерфейсі, навіть якщо бачать його вперше.
- Продуктивність — менше часу витрачається на навчання і взаємодію.

- Ефективна підтримка — легше оновлювати продукт, коли всі елементи побудовані за єдиним принципом.
- Зменшення кількості помилок — передбачуваність інтерфейсу знижує ризик неправильних дій.
- Професійне враження — узгоджений інтерфейс виглядає надійніше та якісніше.

Консистентність реалізована у таких системах, як:

- Material Design (Google) — має суворо визначені правила для відстаней, кольорів, шрифтів, кнопок і поведінки елементів.
- Carbon Design System (IBM) — використовує суворі гайдлайни для UI/UX, засновані на компонентному підході.
- Apple Human Interface Guidelines — забезпечують єдність у дизайні всіх iOS/macOS-додатків.

Принцип консистентності — не просто естетична вимога, а стратегічний підхід до створення зручного й передбачуваного користувацького досвіду. Його реалізація у Figma за допомогою компонентів та змінних дозволяє не лише досягти візуальної єдності, а й оптимізувати командну роботу та процес оновлення інтерфейсів.

## 1.10 Порівняння використання Variables та Components у Figma, Sketch і Adobe XD

Для ефективного використання компонентного підходу та змінних у UI-дизайні важливо розуміти, як ці функції реалізовані у популярних середовищах. Найбільш поширеними інструментами сьогодні є Figma, Sketch і Adobe XD, кожен з яких має свої сильні сторони та обмеження.

Figma пропонує найрозвинутішу систему роботи з компонентами та змінними. Завдяки хмарній основі, вона дозволяє дизайнерам працювати одночасно в одному файлі, а також легко створювати і оновлювати компоненти, змінні та стилі. Нещодавно введені Variables у Figma дозволяють централізовано керувати кольорами, шрифтами, числовими значеннями та навіть режимами (наприклад,

темна/світла тема), що значно покращує підтримку масштабованих дизайн-систем.

Sketch також підтримує символи (аналог компонентів) та стилі, однак відсутність нативної хмарної роботи і менш гнучка реалізація змінних обмежують його у великих командних проєктах. Крім того, змінні реалізуються переважно через сторонні плагіни або зовнішні бібліотеки.

Adobe XD має зручний інтерфейс для створення компонентів і використання кольорів/шрифтів у вигляді стилів. Проте, у порівнянні з Figma, інструменти змінних менш гнучкі. Спільна робота в Adobe XD можлива лише при використанні Creative Cloud, що іноді ускладнює доступ до файлів.

Отже, Figma є найбільш повноцінним рішенням для побудови дизайн-систем з акцентом на консистентність, адаптивність та швидкість командної розробки.

### 1.11 Висновок

У першому розділі було проведено детальний аналіз ключових понять та інструментів, пов’язаних із використанням змінних (анг. Variables) та компонентів (анг. Components) у сучасному UX/UI дизайні. Розглянуто роль цих елементів у побудові масштабованих і консистентних дизайн-систем, зокрема на прикладі середовища Figma, що надає найширші можливості для централізованого керування стилями та структурування інтерфейсу. Також досліджено еволюцію підходів до дизайну, принцип атомарного проєктування та значення консистентності. Результати аналізу створюють міцну теоретичну основу для реалізації практичної частини дипломної роботи — розробки додатку з використанням створеної дизайн-системи.

# РОЗДІЛ 2. СИСТЕМНИЙ АНАЛІЗ, ПРОЕКТУВАННЯ ТА РОЗРОБКА

## 2.1 Поняття та призначення дерева цілей

Для ефективного планування та управління проєктом застосовується дерево цілей – ієрархічна модель, яка наочно демонструє взаємозв'язок між основною метою та її декомпозицією на цілі нижчого рівня. Ця методологія дозволяє не лише систематизувати всі завдання, але й чітко визначити їхню послідовність виконання, що є запорукою успішного управління. Візуально дерево цілей схоже на перевернуту структуру, де головна мета слугує основою (коренем), а другорядні цілі та завдання – її розгалуженнями. Суть цього підходу полягає в деталізації: розбитті великої мети на менші, досяжні частини, що спрощує розуміння проєкту та оптимізує розподіл ресурсів. Кожен наступний рівень цієї ієрархії повинен бути логічним продовженням попереднього, забезпечуючи таким чином цілісність та послідовність усього задуму.

Призначення дерева цілей:

- **Структурування дослідження:** Дерево цілей допомагає розкласти головну мету на логічні пункти та завдання, що дозволяє дотриматись послідовності та чіткого виконання. Це дозволяє краще зрозуміти обсяг роботи та необхідні ресурси
- **Візуалізація взаємозв'язків:** Графічна форма дерева цілей демонструє, як досягнення одних завдань сприяє реалізації проміжних цілей, а їх сукупність веде до досягнення головної мети. Це дозволяє бачити повну картину дослідження та взаємозалежність елементів
- **Визначення пріоритетів та послідовності:** Побудова дерева цілей дозволяє ідентифікувати ключові етапи та завдання, що потребують першочергової уваги. Це полегшує планування роботи, розподіл часу та ресурсів, а також допомагає уникнути дублювання зусиль

Таким чином, використання дерева цілей у процесі написання дипломної роботи є не просто формальною вимогою, а потужним інструментом методологічного планування, що сприяє досягненню високої якості дослідження та його успішному завершенню.

## 2.2 Принципи побудови дерева цілей

Ефективне дерево цілей є основою високоякісного планування та реалізації проєкту, особливо у сфері UI/UX дизайну, де суттєве значення мають структурованість, послідовність та масштабність дій. У межах дипломного дослідження, присвяченого розробці дизайн-системи на основі змінних (анг. Variables) та компонентів (анг. Components), дерево цілей дозволяє не лише візуалізувати логіку під час реалізації проєкту, але й можливість визначити ключові напрями дослідження, розробки та очікуваного впливу [18].

Принципи побудови дерева цілей:

- **Принцип розбиття цілі**: Головна мета - створення дизайн-системи на базі змінних (анг. Variables) та компонентів (анг. Components) - була логічно розбита на основні чотири підцілі: теоретичне дослідження, розробка структури системи, практична реалізація та підвищення ефективності UX/UI. Кожна підціль, у свою чергу, деталізована на конкретні завдання - від вивчення атомарного дизайну до тестування консистентності інтерфейсу. У результаті такий підхід забезпечив послідовність та керованість процесу проектування.
- **Принцип ієрархії:** Цілі нижчого рівня логічно походять із вищих: наприклад, формування бібліотеки змінних є неможливим без вивчення теоретичних основ, а сама реалізація інтерфейсу базується на створеній системі компонентів. Це в свою чергу гарантує причинно-наслідкову логіку проєкту, де кожен крок є передумовою для наступного
- **Принцип вичерпності:** Кожен блок дерева цілей охоплює собою всі суттєві аспекти, які є необхідними для досягнення генеральної мети, а саме: методологічні, технічні та результативні. Завдання сформульовані

так, щоб уникнути прогалин у розробці: до прикладу, у блоці “Розробка структури” є враховано: створення бібліотеки змінних (анг. Variables), варіанти компонентів (анг. Components) та адаптивність системи.

- **Принцип конкретизації:** Формулювання які присутні в дереві є чіткими та спрямованими на досяжні дії: “Створення бібліотеки змінних”, “Формування варіантів компонентів”, “Тестування інтерфейсу”, тощо. Це дозволяє нам об’єктивно проконтролювати такі завдання на етапі реалізації та звітування, що є особливо важливим в рамках дипломної роботи.
- **Принцип незалежності:** Хоча усі підцілі спрямовані на досягнення спільної мети, кожну з них можливо опрацювати відносно автономно: теоретичне дослідження, побудова структури системи та тестування UX/UI не дублюють одна одну, а доповнюють. Це дозволяє забезпечити можливість гнучкого планування часу та роботи над дипломною роботою.
- **Принцип візуалізації:** Графічна форма дерева цілей, реалізована у вигляді таблиці що у свою чергу дозволяє охопити структуру проєкту загалом. Це дозволяє спростити не лише планування та контроль виконання, а й комунікацію з керівником роботи. Візуалізація підкреслює логічну та змістовну цілісність усіх частин дипломного дослідження.

Дотримання цих принципів дозволяє створити не просто формальний документ, а дієвий інструмент для планування, організації та контролю за виконанням дипломної роботи, забезпечуючи її логічну стрункість та успішне завершення.

### 2.3 Методи побудови дерева цілей

Побудова дієвого дерева цілей є повторюваним процесом, що потребує системного підходу. Існує декілька методів та підходів, які допомагають структурувати цей процес, забезпечуючи логічну декомпозицію загальної мети на конкретні задачі. Вибір способу залежить від складності проєкту, кількості учасників та наявних ресурсів.

**Метод “Згори до низу”:**

Це один із найпоширеніших інтуїтивно зрозумілих способів, який починається з формулювання найвищої мети - загальної цілі дослідження чи проєкту. Після цього загальна мета поступово декомпонізується на цілі нижчого рівня, доки не будуть визначені конкретні, вимірювані та досяжні завдання [19].

Етапи:

- Визначення головної мети (наприклад: “Дослідити роль змінних (анг. Variables) та компонентів (анг. Components) у розробці дизайну”).
- Декомпозиція головної мети на кілька ключових підцілей (наприклад: “Дослідити теоретичні засади”, “Проаналізувати практичне використання”, “Розробити рекомендації”)
- Подальша декомпозиція кожної підцілі на більш дрібні завдання та дії доти, доки вони не стануть операційними.

Переваги: Дозволяє забезпечити чітку логіку від загального до часткового та дає змогу підтримувати фокус на головній меті.

Недоліки: може бути достатньо складно виявити усі необхідні підцілі та завдання на початкових етапах, якщо проблема є недостатньо виченою.

**Метод “Знизу вгору”:**

Цей метод власне є альтернативним до класичного підходу “зверху вниз”. Його особливістю є те, що побудова структури цілей починається не з формулювання загальної мети, а з виявлення саме конкретних дій та завдань, які потенційно можуть бути виконані в межах проєкту. Далі ці завдання об’єднуються у підцілі, з яких формулюється загальна мета [20].

Етапи методу:

- Визначення та фіксація усіх можливих мікро-завдань, які стосуються теми дослідження
- Групування завдань у логічні блоки, що формують підцілі
- Об’єднання підцілей у загальну мету проєкту

Переваги: метод є особливо ефективним тоді, коли на старті важко окреслити цілі високого рівня, але вже є чітке уявлення про необхідні дії або деталі.

Недоліки: якщо не встановити контроль на рівні логіки, можлива втрата фокусу або створення структури, не прив'язаної до чіткої мети проєкту.

**Метод "Відцентровий":**

Цей підхід базується на визначенні ключової проблеми чи теми дослідження, навколо якої далі розгортається уся структура цілей та завдань. Візуально цей метод нагадує побудову ментальної карти, де центральна ідея - "змінні (анг. Variables) та компоненти (анг. Components) у дизайні" - розгалужується у різні напрямки, такі як: функціональність, ефективність, дизайн-система, інструменти, тощо.

Етапи методу:

- Визначення центрального елемента (тематики або проблеми);
- Виявлення всіх аспектів, що пов'язані з цією темою;
- Організація отриманих елементів у структуру з ієрархічним підпорядкуванням.

Переваги: метод сприяє генерації ідей, розширенню охоплення проблематики та дозволяє побачити тематику дослідження у широкому контексті.

Недоліки: через відсутність чіткої ієрархії на початку роботи можливе ускладнення структуризації без додаткових інструментів планування.

**Метод "SMART-цілей":**

Цей метод не є самостійною схемою побудови дерева, але застосовується для перевірки якості сформульованих цілей та завдань. Його суть полягає в тому, що кожна ціль має відповідати п'яти критеріям SMART:

- Specific (конкретна): має бути чітко визначена і зрозуміла;
- Measurable (вимірювана): результат повинен підлягати оцінці;
- Achievable (досяжна): має бути реалістичною для виконання;

- Relevant (релевантна): відповідати загальній меті;
- Time-bound (обмежена в часі): мати визначений термін виконання.

Переваги: використання SMART-критеріїв покращує якість планування, мінімізує невизначеність та дозволяє здійснювати контроль на кожному етапі.

Для побудови дерева цілей у межах цієї дипломної роботи було використано поєднання двох методів — “згори до низу” та SMART-підходу (рис.2.3.1). Спочатку було сформульовано загальну мету дослідження — розробка дизайн-системи з використанням змінних (анг. Variables) та компонентів (анг. Components), після чого цю мету було деталізовано на підцілі та завдання згідно з принципом декомпозиції. Одночасно при формулюванні кожного елементу дерева було взятодо уваги критерії SMART, що забезпечило ясність, вимірюваність і реалістичність поставлених цілей.

Такий підхід дав змогу створити логічну, структуровану та керовану систему цілей, яка ефективно відображає послідовність реалізації дипломного проєкту.

*Рис.2.3.1. Дерево цілей*

## 2.4 Поняття інформаційної архітектури

Інформаційна архітектура (анг. Information architecture) - це основа, на якій будується структура будь-якого цифрового продукту: вебсайту, мобільного додатку чи онлайн-сервісу. У найширшому сенсі, IA - це процес організації, структурування та позначення інформаційного контенту таким чином, щоб користувачі могли інтуїтивно орієнтуватися в інтерфейсі, швидк знаходити потрібне та ефективно взаємодіяти з продуктом.

Інформаційну архітектуру часто називають “скелетом” або “логікою” інтерфейсу, тому що вона визначає:

- які елементи будуть наявні в системі;
- як ці елементи пов’язані між собою;
- якою буде структура навігації;
- як реалізується шлях користувача до цілі.

У проєктуванні додатків інформаційна архітектура (анг. Information architecture) не зводиться лише до навігації. Вона охоплює також:

- структуру екранних блоків;
- принципи ієрархії інформації;
- логіку послідовності дій користувача;
- зручність доступу до контенту.

Сучасна інформаційна архітектура (анг. Information architecture) - це не тільки перелік сторінок чи екранів, а комплексне представлення логіки взаємодії з інтерфейсом. Наприклад, у мобільному додатку для пошуку музики, інформаційна архітектура (анг. Information architecture) охоплює: головний екран, процес розпізнавання, збереження результатів, перегляд історії, тощо - з урахуванням їх логічного розташування, взаємозв’язку та ролі в загальній структурі.

Цілі побудови ІА (анг. Information architecture):

- Забезпечити зрозумілу структуру для користувача;
- Мінімізувати кількість кліків до виконання цільової дії;
- Створити логічну та передбачувану навігацію;
- Підвищити якість взаємодії та задоволеність користувача;
- Забезпечити масштабованість системи.

Ключові складові ІА (анг. Information architecture):

- Ієрархія: як впорядкована структура інформації (що головне, що другорядне);
- Навігація: шляхи, по яких користувач пересувається по інтерфейсу;

- Категоризація: як контент розподіляється по розділах або екранах;
- Маркуванн: які терміни та назви використовуються (меню, кнопки, розділи)
- Шляхи користувача: сценарій дій, які виконує кристувач у межаї інтерфейс.

Інформаційна архітектура (анг. Information architecture) тісно пов’язана з поняттям usability (зручність використання), контент-стратегією та дизайном взаємодії. Від якості ІА (анг. Information architecture) залежить, наскільки легко користувачі зможуть користуватись додатком, чи знайдуть вони потрібну функцію, і чи повернуться знову.

Інформаційна архітектура (анг. Information architecture) є критично важливою складовою при створенні будь-якого цифрового продукту, зокрема у сфері UX/UI дизайну. Вона визначає логіку, структуру та взаємозв’язки між інформаційними елементами інтерфейсу, а також забезпечує зручну навігацію, інтуїтивну ієрархію та послідовність дій користувача.

Виділяють кілька типів ІА (анг. Information architecture), залежно від способу організації:

- ієрархічна (структура з головного до підрозділів);
- послідовна (лінійна, наприклад, онбординг);
- матрична (користувач обирає шлях самостійно);
- мережна (взаємозалежні сторінки без єдиного центру).

У дипломному проєкті, що передбачає створення дизайн-системи та інтерфейсу мобільного додатку для пошуку музики, інформаційна архітектура (анг. Information architecture) відіграє вирішальну ролі у формуванні логіки переходів між екранами, зокрема: реєстрацію, головним пошуком музики, історією результатів та профілем користувача.

Методи створення ІА (анг. Information architecture) є різноманітними: наприклад, від побудови карти сайту (анг. sitemap) до розробки ментальних карт (анг. mind maps), інвентаризації контенту, керування шляху користувача (анг.

user journey) та тестування дерева (анг. tree testing). Для побудови структури архітектури використовуються такі інструменти, як Miro або FigJam (для флоу-карт і ескізів), Figma (для деталізації екранів), XMind чи Lucidchart (для логічних схем), а також Notion або Airtable (для структурування контенту). Важливим принципом у побудові ІА є узгодженість, простота, зрозуміла логіка та можливість масштабування [21].

Якісна інформаційна архітектура (анг. Information architecture) досить позитивно впливає на загальний користувацький досвід, а саме: спрощує пошук інформації, зменшує час на виконання задач, знижує ймвірність помилок і покращує навігацію. У той же час, поширеними помилками при проєктуванні ІА залишаються: дублювання розділів, заплутана ієрархія, надто загальні або абстарктні назви, відсутність логіки у згрупуванні функцій. Уникнення таких помилок забезпечує логічну, послідовну та інтуїтивно зрозумілу структуру цифрового продукту.

Інформаційна архітектура (анг. Information architecture) є невід’ємною частиною процесу розробки цифрового продукту, яка визначає логіку розміщення, структурування та взаємозв’язків між елементами інтерфейсу. Саме вона забезпечує зручну навігацію, передбачуваність взаємодії та послідовність користувацьких дій. Ретельне опрацювання ІА (анг. Information architecture) дозволяє уникнути хаотичності в інтерфейсі, спростити орієнтацію користувача та створити цілісну і ефективну систему. У межах даної дипломної роботи інформаційна архітектура (рис.2.2), слугує основою для формування логіки мобільного додатку, забезпечуючи зручний та інтуїтивно зрозумілий шлях до виконання ключових функцій.

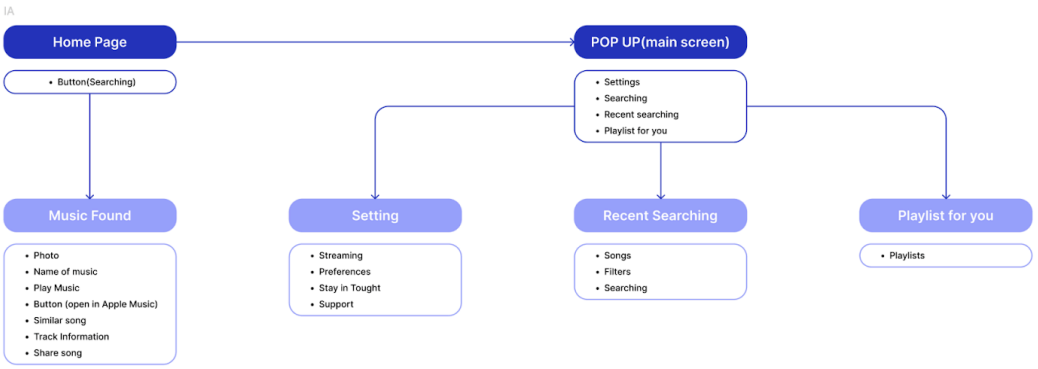


*Рис.2.4.1. Інформаційна архітектура*

## 2.5 Дерево проблем

Аналіз проблем є важливою складовою системного підходу до проєктування цифрових продуктів. Він дає змогу виявити витоки труднощів, що виникають у процесі розробки, та сформувати обґрунтовані цілі для їх подолання. В контексті UX/UI-дизайну, особливо при створенні дизайн-систем, аналіз проблем допомагає зрозуміти, чому виникає неузгодженість інтерфейсу, складності в масштабуванні та інші перепони, що впливають на якість кінцевого продукту.

Одна із головних проблем, що лежить в основі труднощів під час створення ефективного інтерфейсу користувача, є відсутність чітко побудованої та впорядкованої візуальної системи, що власне формує свого роду певну основу для всієї дизайн-системи. Коли інтерфейс створюється без єдиної структури, без поділу на базові стилі, компоненти, це призводить до функціональних і візуальних проблем, які погано впливають як на дизайнерів, розробників так і на кінцевого користувача. Проблема стає помітною в умовах командної роботи, кожен дизайнер створює власні стилі кнопок, відступів, кольорів або типографіки не перевіряючи чи існують вже стандартизовані варіанти. У кінцевому результаті це призводить до дублювання елементів

інтерфейсу, порушення візуальної ієрархії та надмірних витрат часу на оновлення та підтримку продукту.

Наступна поширена проблема - це відсутність задокументованих правил використання візуальних елементів. Якщо немає зафіксованих стандартів для кольорів, шрифтів, відступів, станів компонентів це створює середовище, де кожен дизайнер чи розробник ухвалює рішення самостійно, що призводить до відсутності узгодженості на рівні всього продукту. Відсутність чіткої структури ускладнює не лише сам процес створення інтерфейсу, але й подальше масштабуванн системи. Коли продукт розширюється, з'являються нові екрани і підтримувати візуальну цілісність стає практично неможливо. Також проблема відсутності візуальної системи проявляється в складності впровадження змін. Якщо виникає потреба змінити колір або оновити стилі типографіки, без змінних або чіткої системи компонентів це доводиться робити вручну в десятках, іноді й сотнях окремих файлів або фреймів.

Окрім того, така неструктурованість ускладнює передачу дизайну до розробки. Відсутність правил, призводить до того що розробники часто змушені здогадуватись про логіку роботи компонентів, їх стани та адаптивність. Це не лише сповільнює роботу, але й підвищує ризик неправильного програмування інтерфейсу. Тобто, це стає початковою точкою для більшості наступних проблем у проєктуванні інтерфейсів. Саме тому формування дизайн системи, з чіткими правилами, змінними та компонентами є основою сучасних цифрових продуктів.

Ще один аспект який впливає на якість та ефективність дизайну є відсутність або поверхневе використання таких функцій як, змінні (анг. Variables) та компонентів (анг. Components) у Figma. Хоча на перший погляд ці поняття можуть здаватись технічними деталями на практиці вони є основою будь-якої дизайн системи. Їх неправильне впровадження призводить до ускладнення робочого процесу, збільшення кількості помилок та загальної неузгодженості інтерфейсу.

Якщо в системі не використано змінних (анг. Variables), чи вони не продумані логічно, виникає низка проблем. Найперше - це необхідність ручного редагування. Наприклад, у ситуації коли дизайнер має змінити базовий відтінок кнопок, йому доведеться вручну знайти всі екземпляри з цим кольором і оновити їх окремо. Це не лише неефективно, а й небезпечно з точки зору людського фактору, тому що завжди існує ризик щось пропустити. Відсутність змінних (анг. Variables) суттєво ускладнює масштабування продукту. У сучасних додатках важливо мати можливість швидко змінювати теми, адаптувати стиль під вимоги доступності. Без змінних неможливо реалізувати темну та світлу теми, без дублювання всіх елементів дизайну.

Відсутність або неправильне використання компонентів (анг. Components) створює не менш серйозні ускладнення. При правильному використанні вони дозволяють централізовано змінювати всі елементи одночасно - наприклад, оновити відступи або стиль однієї кнопки і автоматично застосувати зміни до всіх її екемплярів у проєкті. Без використання компонентів (анг. Components) кожен елемент створюється окремо і як наслідок інтерфейс наповнюється великою кількістю схожим, але не ідентичних елементів. Ці проблеми знижують якість користувацького досвіду, викликають плутанину  та знижують довіру користувача.

**2.6 Розробка та опис інтерфейсу користувача**

Розробка інтерфейсу користувача (анг. User Interface) для мобільного додатку була важливою частиною дипломного проєкту, що поєднує в собі як творчі, так і технічні аспекти дизайну. Основним завданням на цьому етапі було створення сучасного, візуально цілісного, доступного та масштабованого інтерфейсу, заснованого на принципах дизайн-системи (анг. Design System).

Особливу увагу було приділено використанню змінних (анг. Variables) та компонентів (анг. Components) у Figma, які дозволили реалізувати логічну та адаптивну структуру інтерфейсу. В результаті було створено набір екранів для

онбордингу, авторизації, головного функціоналу (пошуку музики), а також перегляду історії, результатів і налаштувань.

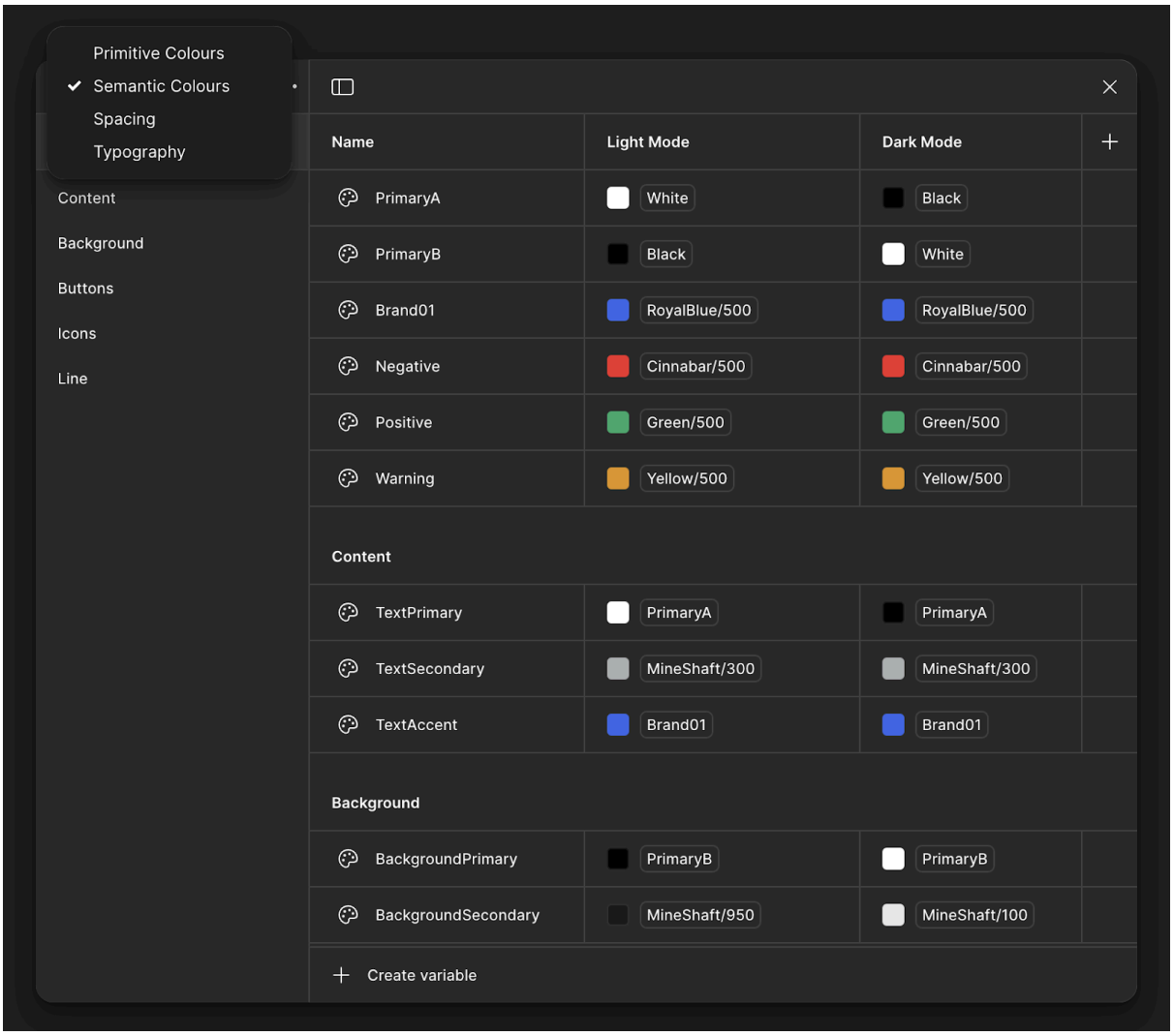


*Рис.2.6.1 Змінні*

На початковому етапі була сформована базова дизайн-система, яка стала основою для всіх подальших рішень. У Figma було створено бібліотеку токенів, що включала:

- Примітивні кольори,
- Семантичні кольори,
- Типографіку ,
- Відступи ,
- Заокруглення.

Завдяки використанню змінних (анг. Variables), було забезпечено централізоване керування стилями (Рис. 2.6.1). Наприклад, зміна кольору у токенах автоматично оновлювала всі відповідні елементи інтерфейсу.

Для забезпечення консистентності дизайну були створені основні компоненти інтерфейсу.

Серед них:

- Кнопки;
- Заголовки та тайтли;
- Інтерактивні елементи;
- Контейнерні блоки;
- Системні елементи.

Ці компоненти були створені в одному вихідному стані та використовувались у всьому проєкті (Рис 2.6.2.). Це дозволило швидко редагувати вміст і візуальне представлення, не порушуючи структури макетів.

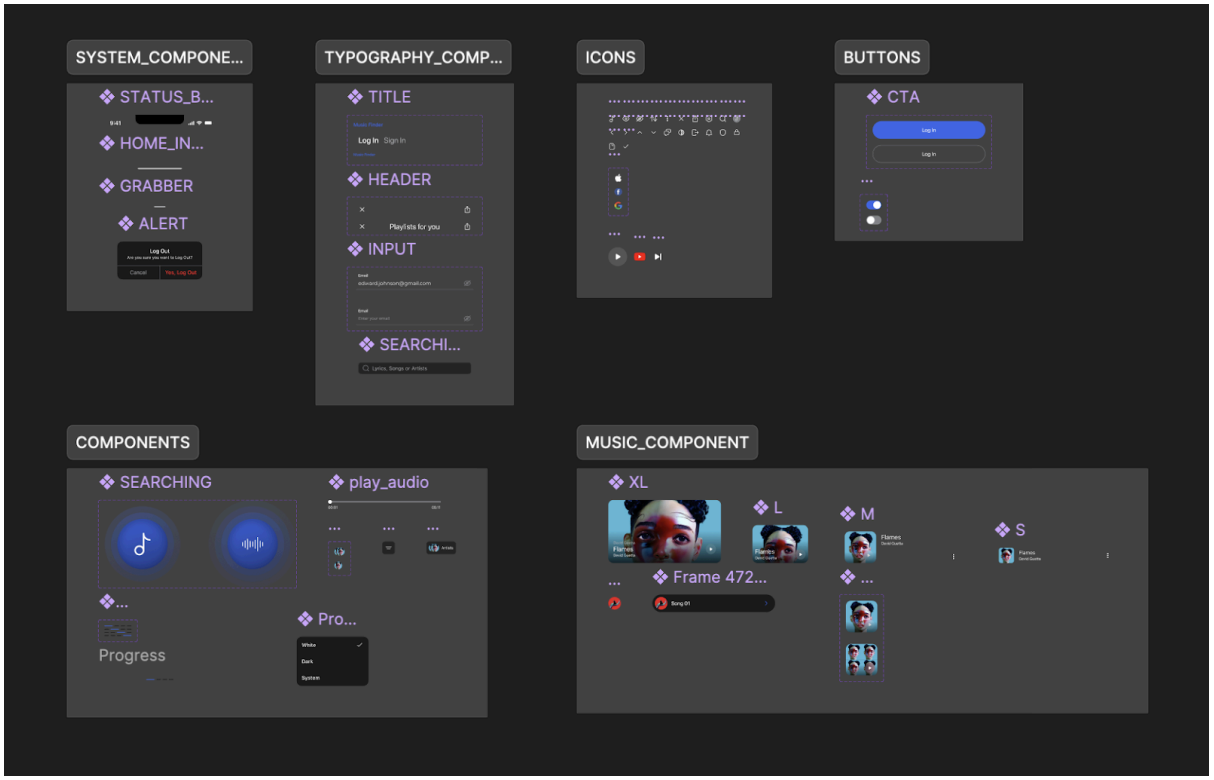


*Рис.2.6.2. Компоненти*

Інтерфейс був побудований за принципами інформаційної архітектури, з дотриманням логіки користувацьких сценаріїв. Основні екрани додатку:

1. **Онбординг** – це процес ознайомлення нових користувачів з основними функціями додатку чи вебсайту після першого запуску. Він допомагає зрозуміти, як ефективно використовувати продукт, формуючи перше враження, що може вплинути на рішення залишитися з ним. Користувач переглядає слайди з підказками.(Рис.2.6.3).

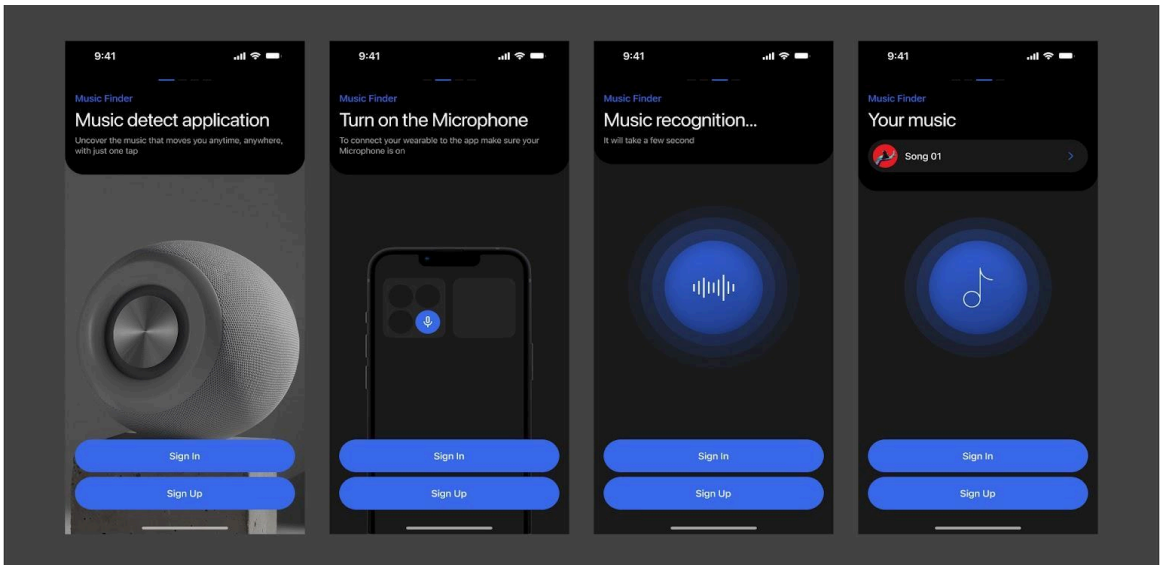


*Рис.2.6.3. Онбординг*

2. **Реєстрація та вхід** – форма авторизації з можливістю зареєструватися або увійти до профілю (Рис. 2.6.4).

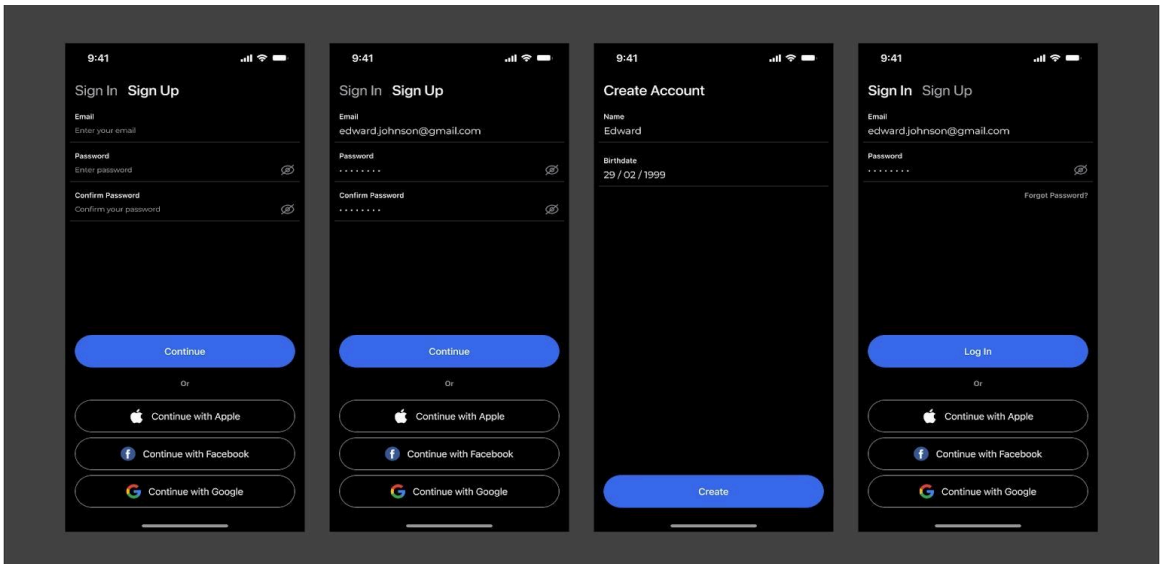


*Рис.2.6.4. Реєстрація та вхід*

3. **Пошук та результати** – центральний елемент додатку, що містить велику кнопку активації пошуку музики, статус розпізнавання, візуальні ефекти. Картка з треком, назвою виконавця, обкладинкою альбому та посиланням на прослуховування.(Рис. 2.6.5.).

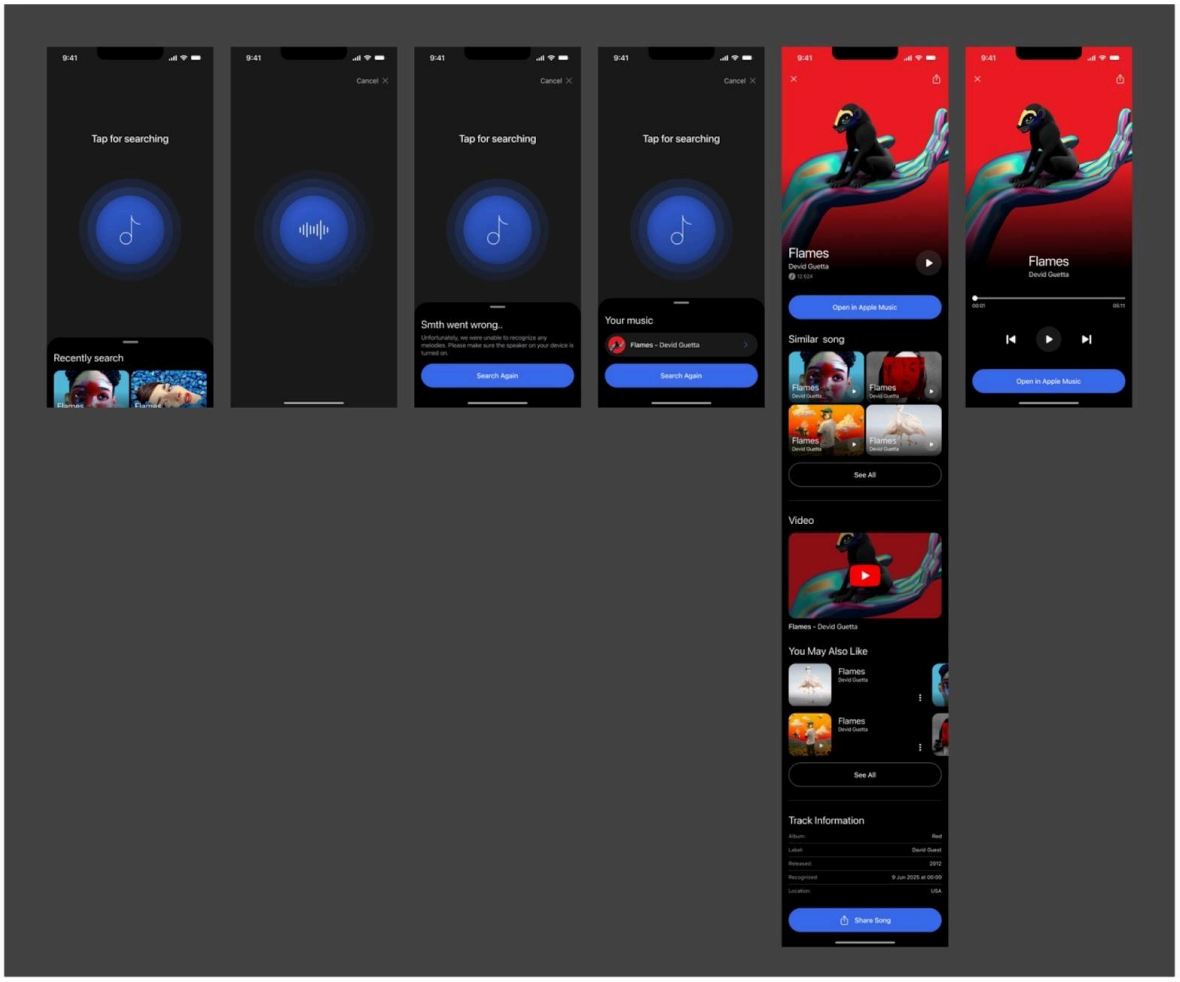


*Рис.2.6.5. Пошук музики*

4. **Історія та пошук** – користувач переглядає список раніше розпізнаних пісень, може скористатись функцією пошуку. Може переглянути раніше прослуховувані пісні. Може скористатись обширним пошуком з фільтрацією (Рис. 2.6.6).
5. **Плейлист** – ще одним важливим аспектом додатку є взаємодія з плейлистами (Рис. 2.6.7), адже користувачі часто хочуть мати можливість не лише знайти пісню, а й організувати свої вподобання. Це є особливо актуальним для тих, хто хоче зберігати пісні, які були знайдені через

гудіння, і створювати персоналізовані списки для подальшого прослуховування. Оскільки ця функція є природною частиною досвіду користувача, вона повинна бути інтегрована в процес використання додатку.

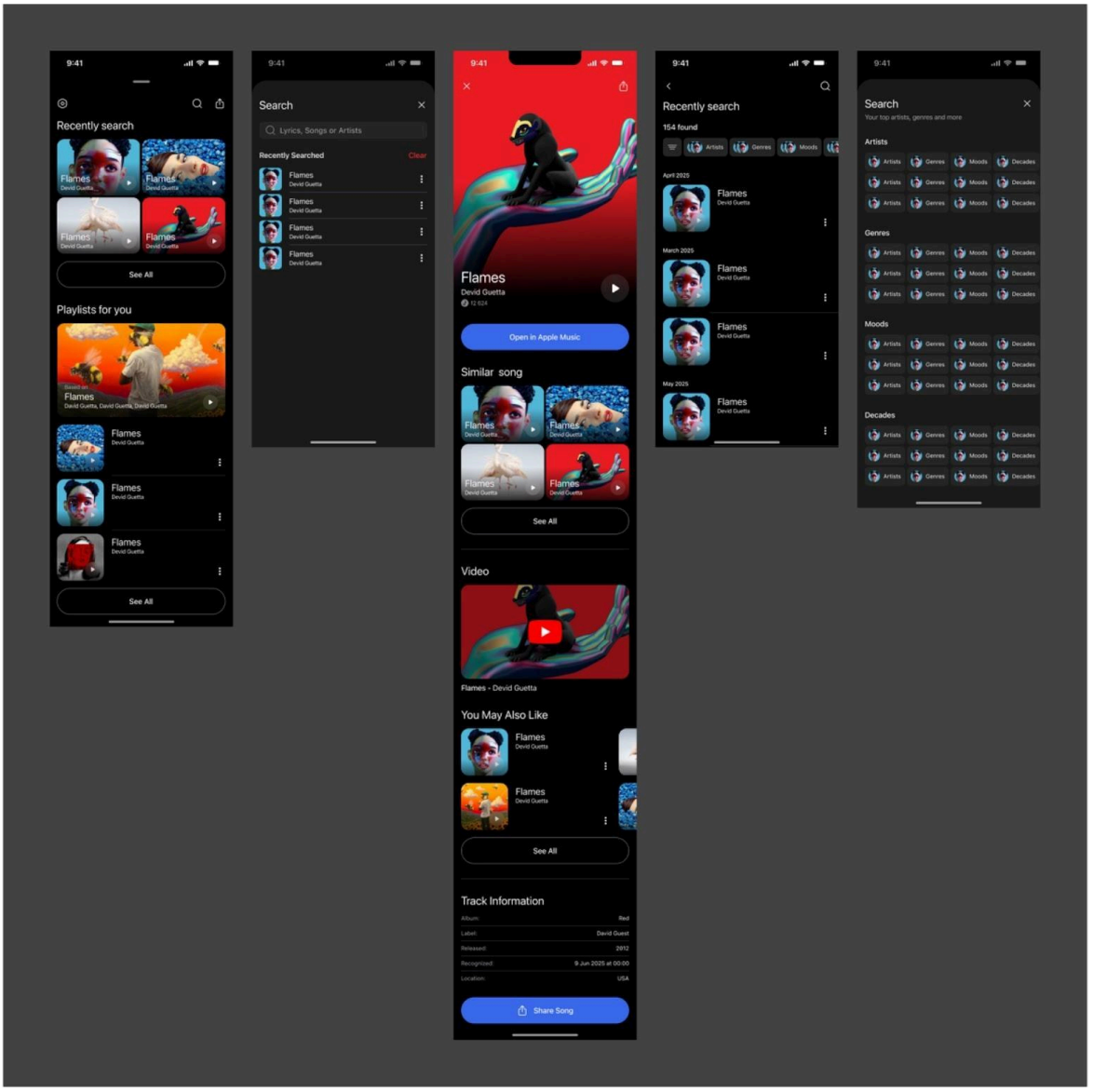


*Рис.2.6.6. Історія та пошук*

6. **Налаштування профілю** – процес взаємодії користувача з налаштуваннями додатку має бути максимально простим і зрозумілим. Оскільки додаток пов'язаний із музикою, користувачі повинні мати можливість змінювати параметри, які дозволяють їм краще налаштувати

досвід використання додатку. Це включає зміну теми інтерфейсу, налаштування сповіщень, зміни персональних даних та інші важливі параметри (Рис.2.6.8).

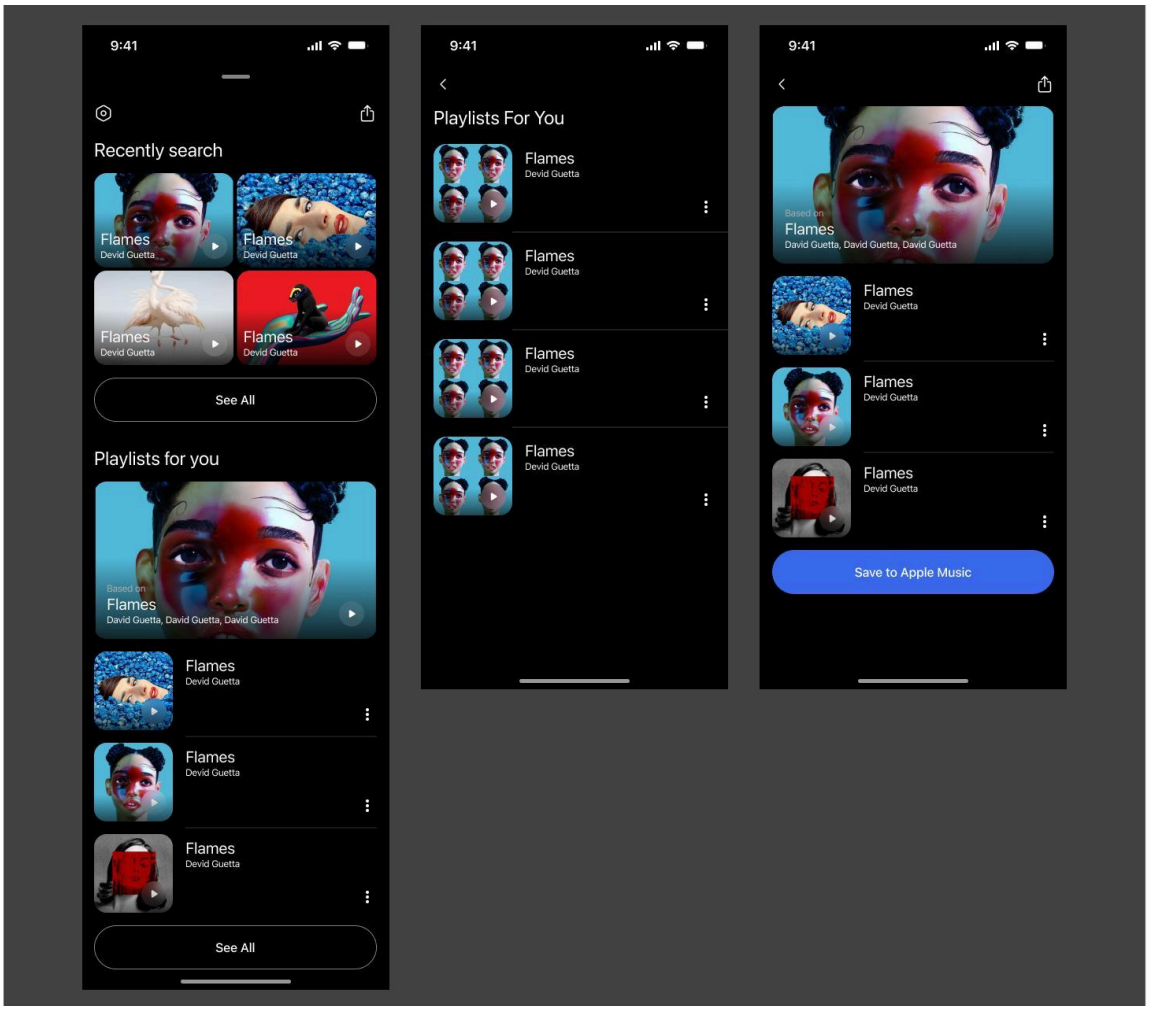


*Рис.2.6.7. Плейлисти*

Для розробки всіх екранів додатку було використано готові компоненти та стилі, що дозволило забезпечити узгодженість і цілісність інтерфейсу. Використання стандартних елементів дозволило не лише пришвидшити процес створення екранів, а й зберегти єдину візуальну концепцію додатку. Компоненти, такі як кнопки, поля введення, заголовки та інші елементи інтерфейсу, були обрані з уже існуючих шаблонів, що гарантувало їх функціональність і відповідність загальним вимогам дизайну. Завдяки такому

підходу додаток виглядає естетично привабливо і зручний у використанні, а також відповідає сучасним стандартам розробки інтерфейсів.

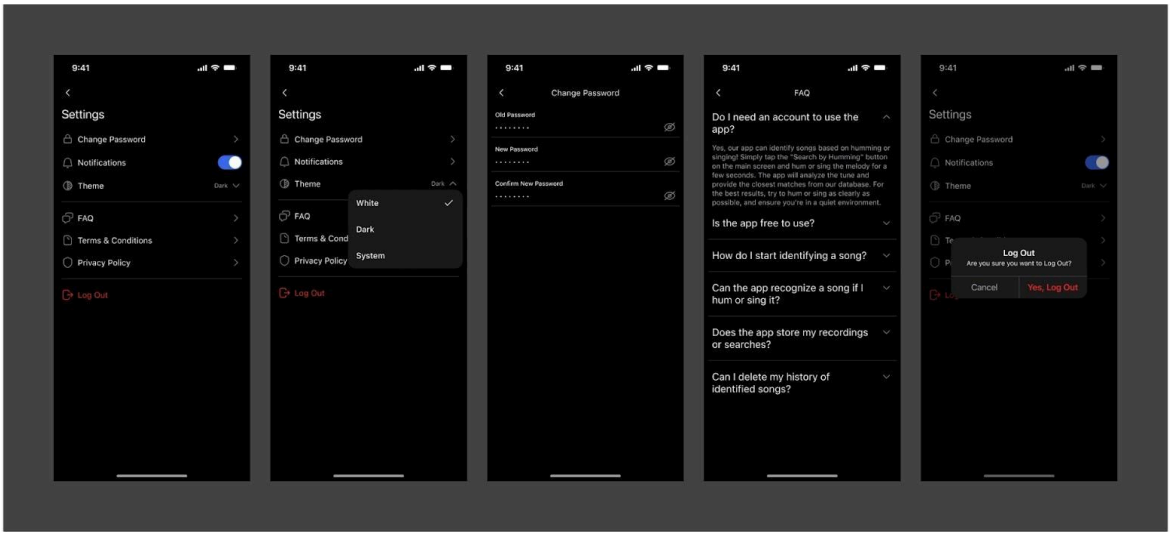


*Рис.2.6.8. Налаштування*

Однією з важливих особливостей інтерфейсу стала підтримка двох тем оформлення — світлої та темної (Рис. 2.6.9). Вони базувалися на однакових компонентах, а переключення між темами реалізовувалось через зміну колірних токенів.

Це рішення стало можливим саме завдяки використанню змінних, що дозволили централізовано керувати палітрою без зміни самих елементів.

Використання власної дизайн-системи дало змогу:

- Забезпечити консистентність інтерфейсу;
- Зменшити кількість ручної роботи;
- Полегшити подальше оновлення UI;
- Створити зрозумілу структуру для спільної роботи або передачі проєкту розробникам.

Робота з дизайн-системою також сприяла розвитку навичок системного підходу до проєктування інтерфейсів, що відповідає сучасним вимогам у професійній сфері дизайну.

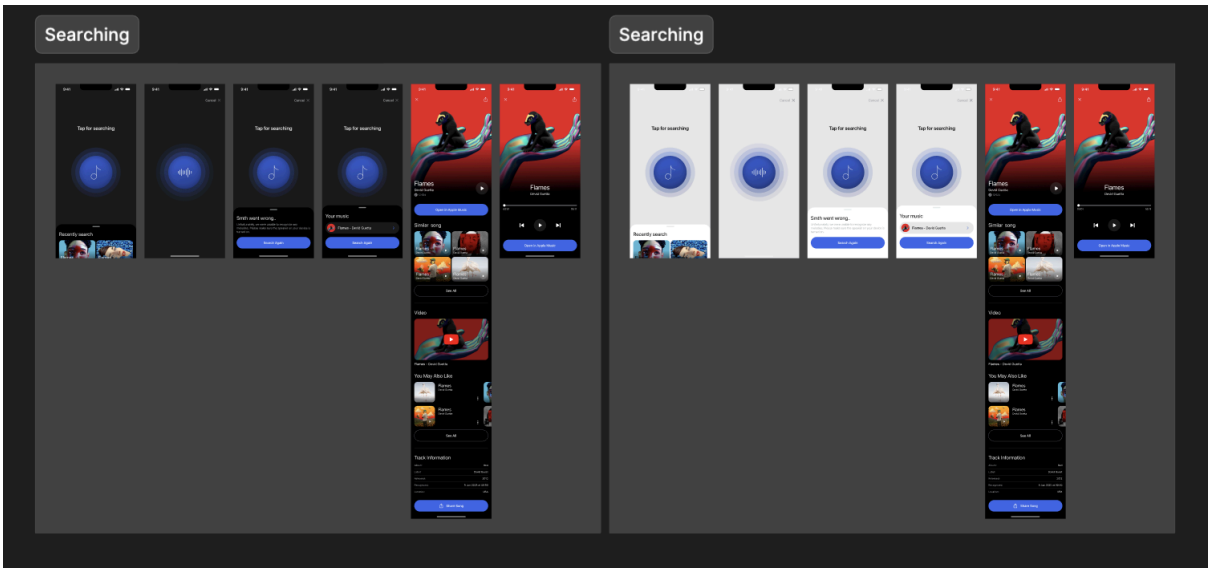


*Рис.2.6.9. Темна та світла теми*

## 2.6 Висновок

Побудова дерева цілей дала змогу чітко визначити структуру роботи, сформулювати основну мету дослідження та розкласти її на конкретні підзадачі, які логічно ведуть до бажаного результату. Водночас, побудова дерева проблем дала можливість глибше осмислити, з якими труднощами стикаються дизайнери у процесі створення інтерфейсів без структурованого підходу. Було виявлено, що відсутність чіткої візуальної системи, а також неправильне чи недостатнє застосування змінних (анг. Variables) та компонентів (анг. Components) у Figma призводить до неузгодженості інтерфейсів, збільшення часу на розробку, ускладнення командної роботи та погіршення користувацького досвіду. Також окреслено методологічні основи побудови інформаційної архітектури, котра виступає фундаментом для подальшого дизайну додатку та організації навігації.

Таким чином, розробка інтерфейсу користувача була реалізована на основі принципів дизайн-системи та з активним використанням змінних і компонентів у Figma. Це дозволило створити гнучкий, зручний, сучасний інтерфейс, який легко масштабувати, підтримувати та адаптувати під нові потреби користувачів. Структура інтерфейсу відповідає завданням додатку, а реалізація світлої й темної тем підтверджує адаптивність та глибину опрацьованого дизайну.

# РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

## 3.1 Інструкція

Важливим етапом у створенні зручного та інтуїтивно зрозумілого інтерфейсу є правильна інструкція для користувача, яка дозволяє йому ефективно взаємодіяти з додатком. Інтерфейс має бути простим і не

перевантаженим, а кожна функція повинна бути логічно зрозумілою. Інструкція для користувача містить покрокові рекомендації для виконання базових завдань у додатку, таких як реєстрація, пошук пісень, взаємодія з результатами пошуку та зміна налаштувань.

Перш за все, користувач має можливість увійти в додаток або зареєструвати новий обліковий запис. Це можна зробити за допомогою адреси електронної пошти або через соціальні мережі (Facebook, Google, Apple). Після цього користувач отримає доступ до основних функцій додатку.

- **Крок 1**: Користувач натискає на кнопку "Реєстрація/Вхід".
- **Крок 2**: Вибір способу реєстрації або входу — через електронну пошту або через соцмережі.
- **Крок 3**: Введення даних, таких як ім’я, email, пароль, дата народження. Потім натискання кнопки «Створити акаунт» або «Увійти».

Для початку пошуку пісні користувач натискає на головну кнопку «Пошук». Це відкриває екран, де він може гудіти або співати мелодію.

- **Крок 1**: Користувач натискає на кнопку «Пошук за гудінням» на головному екрані.
- **Крок 2**: Додаток починає записувати звуки користувача.
- **Крок 3**: Після декількох секунд запису додаток аналізує звук і надає список пісень, що відповідають мелодії.

Після того, як додаток знайде відповідні пісні, користувач може переглянути список знайдених треків. Система виведе список пісень, що найбільше підходять під мелодію користувача.

- **Крок 1**: Користувач може вибрати одну з пісень з запропонованого списку.
- **Крок 2**: Після вибору пісні, користувач може прослухати її або перейти до інших функцій додатку.

Користувач може змінювати налаштування акаунта, включаючи зміну пароля, налаштування сповіщень, а також обрати темну або світлу тему інтерфейсу.

- **Крок 1**: У меню налаштувань користувач може змінити пароль.
- **Крок 2**: Увімкнути або вимкнути сповіщення.
- **Крок 3**: Змінити тему додатку — темна або світла.

Простота використання є важливою складовою додатку. Інтерфейс був спроектований таким чином, щоб кожен етап був логічно пов'язаний з попереднім. Перш за все, користувач вводить свої дані для реєстрації або входу, після чого має можливість здійснити пошук пісні, використовуючи свій голос. Всі кнопки та елементи інтерфейсу мають чітке маркування, що допомагає уникнути непорозумінь. Інтерфейс виконано у стилі, що поєднує елементи сучасного дизайну з простотою.

Пошук пісень за допомогою гудіння чи співу – це зручний та інтуїтивно зрозумілий процес. Користувачеві не потрібно вводити текст або згадувати точну назву пісні чи виконавця. Алгоритм розпізнавання мелодії працює швидко та точно, навіть якщо користувач не може відтворити точний оригінал пісні. У результатах пошуку відображається список найбільш схожих пісень, що дозволяє користувачеві швидко знайти потрібну композицію.

Користувачі можуть налаштувати додаток під свої вподобання, змінюючи тему інтерфейсу на темну або світлу, а також отримувати повідомлення про нові функції чи оновлення додатку. Окрім того, є можливість змінити пароль та налаштувати інші параметри облікового запису. Це забезпечує додатковий рівень персоналізації та зручності.

Користувачі можуть зіткнутися з деякими проблемами під час пошуку пісень, зокрема, якщо запис мелодії занадто нечіткий або якщо в навколишньому середовищі багато фонових шумів. Для усунення цих проблем додаток може містити функції для попередження користувача про необхідність чистоти звуку, а також автоматичну корекцію незначних спотворень.

### 3.2 Правила використання дизайн-системи та токенів для дизайнерів

Дизайн-система — це не лише набір стилів, шрифтів та кольорів, а й своєрідна основа для побудови інтерфейсу користувача, яка забезпечує єдину

візуальну мову в рамках всього додатку. Вона є важливою складовою процесу проектування інтерфейсів і допомагає забезпечити послідовність у роботі над продуктом, даючи чіткі рекомендації для дизайнерів та розробників. Правильне використання дизайн-системи дозволяє спростити роботу над проектом, знижує ризик виникнення помилок і забезпечує зручність для користувачів.

Дизайн-система має включати декілька основних елементів, кожен з яких має чітко визначену роль у формуванні інтерфейсу:

- Компоненти інтерфейсу - є основними елементами, з яких складається додаток. Вони включають такі елементи, як кнопки, поля введення, спливаючі вікна, вкладки, меню тощо. Всі ці компоненти мають бути однаковими за стилем і функціональністю на всіх екранах додатку, щоб забезпечити послідовність взаємодії для користувачів.

  Наприклад, кнопка на головному екрані додатку для пошуку пісень повинна мати однакові розміри, кольори та шрифти з іншими кнопками додатку. Так само повинні бути чітко визначені стани кнопки (активний, неактивний, натиснутий) — це дозволить користувачеві легко орієнтуватися та зрозуміти, коли він взаємодіє з елементом інтерфейсу.
- Типографіка в дизайн-системі включає всі правила, що стосуються шрифтів, їх розмірів, відстаней між рядками та абзацами, а також шрифтів для заголовків, основного тексту та допоміжних елементів (наприклад, підписів до кнопок або підказок). Вибір правильного шрифта є важливим аспектом дизайну, оскільки він визначає зручність для читання тексту. Шрифт має бути чітким і легко читабельним на різних пристроях та при різних роздільних здатностях екранів.

  Приклад: Для заголовків можна використовувати більш жирний шрифт, який буде чітко виділяти важливу інформацію на екрані, а для основного тексту підійде більш легкий шрифт, щоб забезпечити зручність читання. Розмір шрифту повинен змінюватися залежно від контексту: заголовки

повинні бути більшими за основний текст, щоб привертати увагу користувача.

- Колірна палітра є одним з найважливіших аспектів дизайн-системи, оскільки вона визначає, як будуть виглядати всі елементи інтерфейсу. Правильне використання кольорів дозволяє створити приємний для ока інтерфейс, що допомагає користувачам швидше орієнтуватися в додатку. У дизайн-системі зазвичай визначається кілька основних кольорів для фону, кнопок, тексту та інших елементів.

  Приклад: Для фону можна вибрати нейтральні кольори, такі як білий або сірий, щоб не відволікати увагу користувача від основного контенту. Для кнопок та важливих елементів інтерфейсу використовуються більш яскраві кольори, щоб привернути до них увагу. Потрібно уникати використання занадто яскравих кольорів на великій площі, оскільки це може зробити інтерфейс перенасиченим і важким для сприйняття.

- Іконки є невід’ємною частиною інтерфейсу, і їх використання вимагає особливої уваги. Іконки мають бути універсальними та зрозумілими для користувачів, незалежно від того, яка мова інтерфейсу використовується в додатку. Іконки повинні бути чіткими, простими, але в той же час достатньо інформативними, щоб користувачі могли одразу зрозуміти їх функцію.

  Приклад: Іконка для пошуку пісні може бути у вигляді мікрофона або слухавки, що чітко сигналізує користувачеві, що це елемент для пошуку за допомогою голосу. Іконки мають бути однаковими за стилем та розмірами на всіх екранах додатку.

- Змінні (анг. Variables) — це одиничні значення, які використовуються для визначення стилів у дизайн-системі. Це можуть бути значення кольорів, шрифтів, відступів, розмірів елементів тощо. Використання токенів допомагає забезпечити узгодженість візуальних елементів на різних екранах та етапах розробки.

- ВВідступи та кути елементів є важливими складовими дизайн-системи, оскільки вони не лише покращують зовнішній вигляд інтерфейсу, а й забезпечують комфорт користувачів під час взаємодії з продуктом. Відступи між елементами інтерфейсу відіграють ключову роль у запобіганні перевантаженню екрану та створенні достатнього простору між різними компонентами. Округлі кути можуть надати інтерфейсу м'якший і дружелюбніший вигляд, роблячи його більш сучасним і приємним для користувача, тоді як прямокутні кути можуть створювати більш строгий, мінімалістичний вигляд, що підходить для продуктів з професійною або корпоративною тематикою.

  Приклад: Кнопки можуть мати округлі кути для надання м’якішого вигляду, а відступи між полями введення можуть становити 20 пікселів, щоб не було занадто тісно і користувачі могли легко орієнтуватися.
- Шрифтові токени визначають стилі шрифтів, які використовуються для текстових елементів в додатку. Це включає такі параметри, як розмір шрифтів для заголовків, підзаголовків, основного тексту та кнопок, а також вага шрифтів (наприклад, жирний, звичайний, тонкий) (Рис.3.2.1). Токени для шрифтів допомагають підтримувати однаковий вигляд тексту на всіх екранах додатку.

  Приклад: Шрифт для заголовків має бути більшим і жирнішим, ніж для основного тексту. Для кнопок використовується середній розмір шрифта, що легко читається.

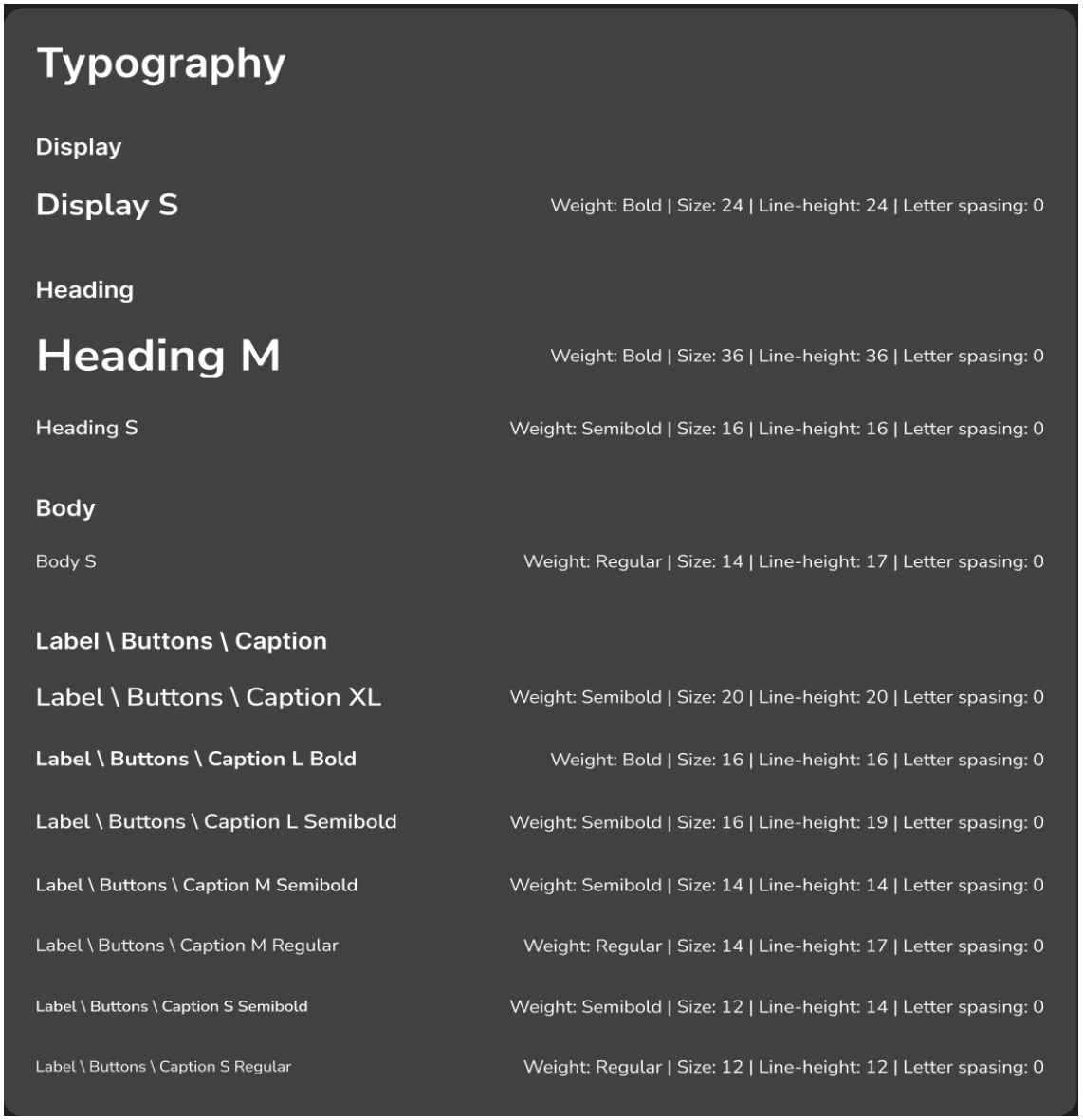


*Рис.3.2.1. Приклад використання шрифтів*

Дизайн-система та токени відіграють ключову роль у забезпеченні консистентності та зручності для користувачів додатку (Рис. 3.2.2). Вони дозволяють створювати зручний та ефективний інтерфейс, що відповідає вимогам користувачів і підтримує єдиний стиль на всіх екранах. Всі компоненти, кольори, шрифти та відступи повинні бути обрані так, щоб користувач відчував зручність та інтуїтивну зрозумілість під час використання додатку.

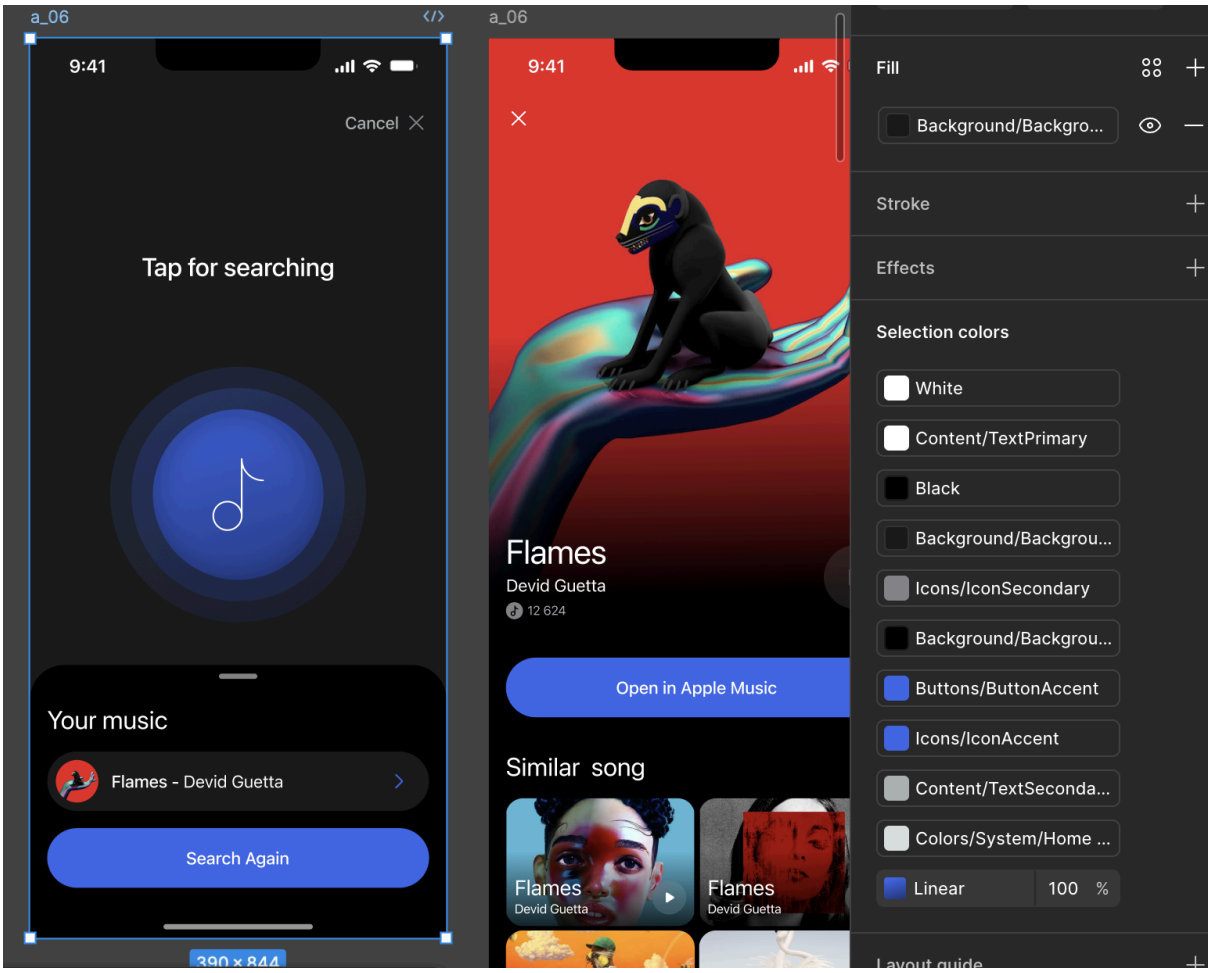


*Рис.3.2.2. Приклад використання змінних*

## 3.3 Тестування зручності

Тестування зручності використання є важливим етапом при розробці будь-якого додатку, адже воно дозволяє зрозуміти, як кінцеві користувачі взаємодіють з інтерфейсом та наскільки зручним є додаток. У межах цього розділу було проведено серію тестів, спрямованих на оцінку інтерфейсу додатку з точки зору користувачів. Тестування базувалося на кількох ключових методиках, завданнях для користувачів та спостереженнях, які дозволили зібрати зворотний зв’язок.

Для тестування зручності використання було обрано дві основні методики:

- Тестування за сценаріями: Користувачам пропонували виконувати певні завдання в рамках сценаріїв, що відображають реальні умови

використання додатку. Наприклад, завдання для користувачів полягало в тому, щоб знайти пісню, використовуючи лише гудіння, або зареєструвати новий акаунт.

- Тестування за часом: Цей тип тестування передбачає вимірювання часу, який користувач витрачає на виконання завдання. Це дозволяє оцінити, наскільки інтерфейс є швидким та ефективним.

Також важливим елементом тестування є спостереження за тим, як користувачі взаємодіють з інтерфейсом. Це дозволяє виявити потенційні проблеми, які можуть виникнути через незрозумілу навігацію або неочевидні функції.

Для проведення тестування було обрано дві групи учасників:

- Група 1: Новачки у користуванні додатками для пошуку музики. Це користувачі, які не мали попереднього досвіду в користуванні подібними додатками. Зазвичай вони мали базові навички роботи зі смартфонами, але не мали досвіду в пошуку музики через голосовий ввід або гудіння.
- Група 2: Досвідчені користувачі мобільних додатків. Це були користувачі, які активно використовують додатки для пошуку музики або мають досвід взаємодії з технологіями розпізнавання звуків та голосових команд.

Кожній групі було запропоновано виконати однакові завдання, щоб порівняти, як вони впораються з різними етапами взаємодії з додатком.

Для кожної групи учасників було підготовлено набір завдань, що дозволяють оцінити різні аспекти роботи з додатком. Ось деякі з основних завдань, які виконували учасники:

- Завдання 1: Пошук пісні за допомогою гудіння. Користувачі мали знайти пісню, використовуючи лише гудіння або спів. Це завдання дозволяло оцінити точність роботи алгоритму розпізнавання звуку, а також зрозуміти, наскільки зручним є сам процес пошуку пісні.

- Завдання 2: Реєстрація акаунта. Користувачі повинні були створити новий акаунт, введіть свої дані, включаючи ім’я, email та пароль. Це завдання дозволило оцінити, наскільки інтуїтивно зрозумілою є форма реєстрації.
- Завдання 3: Перегляд результатів пошуку. Після виконання пошуку, користувачі повинні були вибрати пісню зі списку результатів. Це завдання дозволило оцінити, наскільки коректно та зрозуміло представлені результати пошуку і чи легко користувачам знаходити потрібну пісню.

У ході тестування проводились спостереження за поведінкою користувачів. Зазвичай це включало фіксацію моментів, коли учасники не могли зрозуміти, як виконати те чи інше завдання, або коли вони зробили помилки через незрозумілі елементи інтерфейсу.

Перші результати:

- Група 1 (новачки): Користувачі цієї групи мали труднощі на етапі пошуку пісні за допомогою мелодії. Багато хто з учасників не зрозумів, що потрібно робити після натискання на кнопку «Пошук». Крім того, деякі користувачі не могли розпізнати, що саме додаток просить їх зробити.
- Група 2 (досвідчені користувачі): Ця група впоралася із завданням швидше і без значних труднощів. Однак вони виявили, що інтерфейс пошуку можна зробити більш чітким, додавши інструкції або підказки. Користувачі цієї групи також зазначили, що підсвічування кнопок при натисканні зробило додаток більш інтуїтивно зрозумілим.

Результати тестування:

- Час виконання завдань: Група 2 зазвичай виконувала завдання на 10-15% швидше, ніж група 1. Це вказує на те, що інтерфейс додатку вимагає мінімальних удосконалень для користувачів, які не мають досвіду роботи з подібними технологіями.

- Кількість помилок: У групи 1 було зафіксовано більше помилок, що свідчить про те, що інтерфейс потребує покращень у питанні підказок та інструкцій для новачків.
- Задоволеність користувачів: У цілому учасники з групи 2 високо оцінили точність розпізнавання пісень за мелодією. Вони також зазначили, що додаток є швидким та зручним, але потребує покращення в плані інструкцій для новачків. Група 1, в свою чергу, відзначила, що процес реєстрації та пошуку пісень був інтуїтивно зрозумілим лише після кількох спроб.

Тестування показало, що додаток працює добре для досвідчених користувачів, але має деякі проблеми для новачків, особливо у відсутності підказок і зрозумілих інструкцій на етапах пошуку пісні за допомогою гудіння та співу. Тому варто рекомендувати:

- Додавати підказки на основному екрані для нових користувачів.
- Поліпшити інтерфейс з точки зору відображення результатів пошуку, щоб користувачі могли швидше знаходити потрібні пісні.
- Зробити процес реєстрації та налаштувань ще простішим, щоб нові користувачі могли виконувати всі кроки без зайвих труднощів.

Тестування зручності допомогло зрозуміти слабкі місця інтерфейсу та зібрати цінні відгуки, що допоможе в майбутньому вдосконалити додаток.

### 3.4 Рекомендації для майбутніх покращень на основі тестування

Після проведення тестування зручності використання, яке було описано в попередньому розділі, були отримані цінні відгуки та спостереження від користувачів. Ось кілька основних аспектів, які потребують покращення, а також рекомендації щодо того, як можна удосконалити додаток, щоб зробити його більш зручним і ефективним для кінцевих користувачів.

Покращення процесу пошуку пісень за допомогою звуків:

- Проблеми, що були виявлені: однією з основних проблем, яку виявили користувачі під час тестування, було те, що деякі з них не зовсім

зрозуміли, як саме працює функція пошуку пісні за допомогою звуків. Це призвело до певних труднощів на початкових етапах використання додатку, особливо серед новачків, які не були знайомі з таким способом пошуку.

Рекомендації:

- Додати чіткі інструкції: Щоб уникнути непорозумінь, необхідно додати прості й зрозумілі підказки або інструкції, які пояснюють користувачеві, як правильно використовувати функцію гудіння для пошуку пісні. Це можуть бути підказки на екрані перед початком пошуку.
- Візуальні підказки під час процесу: Під час самого процесу пошуку можна додавати анімації або змінювати кольори кнопок, що показуватиме, коли додаток активно записує звук, а також надаватиме користувачеві візуальне підтвердження, що додаток чує його. Це допоможе зменшити кількість помилок і зробить процес більш інтуїтивно зрозумілим.

Очікуваний результат: Це покращить процес пошуку для новачків і зробить його більш інтуїтивно зрозумілим, що підвищить загальну задоволеність користувачів.

Покращення інтерфейсу для нових користувачів

- Проблеми, що були виявлені: Під час тестування нові користувачі стикалися з труднощами в навігації додатком. Вони не одразу знаходили потрібні функції, зокрема в розділі налаштувань або під час пошуку пісень. Хоча досвідчені користувачі могли легко орієнтуватися, новачки потребували більше вказівок та підказок.

Рекомендації:

- Додавання інтерактивного онбордингу: Для нових користувачів варто додати інтерактивне введення в додаток, що пояснює основні функції (наприклад, "Як знайти пісню за допомогою гудіння"). Це може бути у вигляді короткого анімаційного відео або послідовних кроків із підказками.

- Покращення навігації: Додаток має бути ще більш інтуїтивно зрозумілим. Крім того, варто провести оптимізацію меню, щоб не було необхідності робити зайві кілька кроків для доступу до функцій.
- Тестування на нових користувачах: Потрібно провести додаткові тести з більшою кількістю нових користувачів для того, щоб точно оцінити, які елементи інтерфейсу викликають труднощі. Це допоможе зрозуміти, на яких етапах користувачі часто роблять помилки або забувають, що потрібно зробити.

Очікуваний результат: Це покращить досвід нових користувачів і зробить процес навчання та використання додатку більш комфортним, зменшивши кількість помилок та запитів на технічну підтримку.

Поліпшення процесу реєстрації та налаштувань акаунта

- Проблеми, що були виявлені: Процес реєстрації та налаштування акаунта був зрозумілим для більшості користувачів, але деякі з них не одразу розуміли, що потрібно робити на етапі підтвердження пароля або коли з'являлася помилка введення (наприклад, занадто слабкий пароль). Це призводило до незначних затримок.

Рекомендації:

- Забезпечити чіткі підказки при введенні пароля: Додати інструкції, які підказують користувачеві, що має бути в паролі (наприклад, мінімальна кількість символів, наявність цифр чи спеціальних символів). Це дозволить знизити кількість помилок при реєстрації.
- Автоматичні підказки та валідація: Важливо додати валідацію в реальному часі, коли користувач вводить дані. Наприклад, коли користувач вводить електронну пошту або пароль, система одразу виводить підказку, чи правильний формат, або які зміни необхідно внести. Це зменшить кількість помилок і зробить процес реєстрації швидшим.

Очікуваний результат: Це дозволить користувачам швидше і безпомилково пройти процес реєстрації та налаштування акаунта, що покращить їх перше враження від використання додатку.

### 3.5 Економічна частина

Економічна частина дипломної роботи має на меті оцінити вартість робіт на різних етапах розробки додатку, а також допомогти зрозуміти, скільки коштуватимуть послуги команди, яка працює над проектом.

Для оцінки вартості проекту необхідно врахувати, скільки часу кожен член команди витратить на виконання своїх обов'язків, а також які години роботи будуть оплачуватися за відповідними ставками.

Команда проекту:

- Дизайнер: відповідає за створення інтерфейсу додатку, розробку дизайн-системи, вибір кольорів, шрифтів, компонентів і т. д. Оскільки цей етап включає в себе багато творчої роботи, розрахунок часу буде базуватися на створенні кількох екранів додатку.
- Аналітик: проводить дослідження ринку, аналізує конкурентів, вивчає потреби користувачів, а також збирає зворотний зв'язок від тестувальників. Аналітик також відповідає за формування вимог до продукту.
- Проджект-менеджер: займається координацією команди, визначенням термінів виконання завдань, контролює якість та кількість роботи, а також забезпечує, щоб проект був завершений вчасно.

Годинні ставки:

- Дизайнер: $15/год (середня ставка для junior-дизайнера в більшості компаній).
- Аналітик: $20/год (аналітики зазвичай мають вищу ставку через досвід і специфіку роботи).
- Проджект-менеджер: $30/год (висока ставка пов'язана з управлінськими та координаційними функціями).

Кожен член команди має свої завдання, і потрібно точно визначити, скільки часу кожен з них витратить на виконання цих завдань. Ось приклад розподілу часу:

**Дизайнер**:

- Розробка концепту дизайну та прототипів для основних екранів додатку (головний екран, екран пошуку, екран результатів пошуку, екран налаштувань) – 40 годин.

- Створення та налаштування дизайн-системи (кнопки, поля введення, блоки, шрифти, кольори) – 20 годин.
- Робота з фідбеком від тестування та адаптація дизайну згідно з рекомендаціями – 20 годин.

Загальний час для дизайнера: 80 годин.

**Аналітик**:

- Дослідження ринку та конкурентів, аналіз подібних додатків (визначення сильних та слабких сторін конкурентів) – 20 годин.
- Розробка вимог до додатку на основі аналізу – 20 годин.
- Проведення тестувань з користувачами, збір та обробка результатів – 20 годин.

Загальний час для аналітика: 60 годин.

**Проджект-менеджер**:

- Керування проектом, координація команди, контроль дедлайнів та задач – 20 годин.
- Проведення зустрічей, комунікація з замовниками, звітність – 10 годин.
- Фінальний огляд роботи, підготовка документації – 10 годин.

Загальний час для проджект-менеджера: 40 годин.

Тепер, коли у нас є кількість годин для кожного учасника команди, ми можемо розрахувати загальну вартість робіт.

- Дизайнер: 80 годин * $15/год = $1200
- Аналітик: 60 годин * $20/год = $1200
- Проджект-менеджер: 40 годин * $30/год = $1200

Загальна вартість робіт (без податків):

$1200 (дизайнер) + $1200 (аналітик) + $1200 (менеджер) = $3600

Згідно з чинним законодавством, ми повинні враховувати податки, які додаються до загальної вартості проекту. Податкова ставка складає 20%, тому потрібно порахувати додаткові витрати: 20% * $6800 = $720

Загальна вартість проекту з податками: $3600 (основна вартість) + $720 (податки) = $4320

Проект оцінюється на період 2 місяці. За цей час кожен член команди виконує свої завдання. Кількість годин, яку потрібно витратити на проект, наводиться в попередньому розділі, і це дозволяє краще оцінити загальний обсяг роботи.

Крім витрат на робочу силу, варто врахувати додаткові витрати на інші ресурси:

- Програмне забезпечення: Для розробки дизайну та тестування можуть бути використані платні інструменти, Figma, а також сервери для хостингу додатків.
- Тестування: Може знадобитися певна кількість коштів для найму тестувальників або придбання спеціального обладнання для тестування продуктивності.

Проте для більшості студентських проектів ці витрати часто не є значними, оскільки використовуються безкоштовні версії програм або вже наявні ресурси.

Загальна вартість робіт для проекту включає оплату роботи команди (дизайнера, аналітика, проджект-менеджера) та податки, що складає:

- Основна вартість: $3600
- Податки: $720
- Загальна вартість: $4320

Ця сума є орієнтовною і може варіюватися залежно від специфіки проекту та необхідних додаткових витрат на інші ресурси (наприклад, програмне забезпечення або тестування).

**3.5 Висновок**

У цьому розділі було детально розглянуто експериментальну частину розробки додатку для пошуку музики за допомогою мелодії. Пройдено через кілька етапів, починаючи від тестування зручності використання додатку і закінчуючи економічною оцінкою витрат на проект.

Під час тестування зручності були використані різні методики, включаючи тестування за сценаріями та за часом. Виявилось, що додаток працює добре для досвідчених користувачів, але потребує вдосконалення для новачків. Більшість користувачів, які не мали досвіду в пошуку пісень за допомогою гудіння, стикалися з труднощами, зокрема в процесі реєстрації та пошуку пісень. Це підтвердило необхідність покращення інтерфейсу, додавання підказок та інтерактивного введення для нових користувачів.

На основі тестування було визначено кілька основних напрямків для покращень, таких як оптимізація процесу пошуку пісень, поліпшення навігації для новачків, вдосконалення алгоритмів пошуку і додавання підказок для користувачів..

У фінальному підсумку розділ також містив економічну частину, в якій були розраховані витрати на розробку проекту, включаючи годинні ставки для дизайнерів, аналітиків та проджект-менеджерів. Загальна вартість проекту, з урахуванням податків, склала $4320. Ці розрахунки дають чітке уявлення про фінансові аспекти проекту та допомагають краще планувати ресурси для подальшої реалізації.

Загалом, експериментальна частина показала важливість тестування зручності для покращення користувацького досвіду, а також надала необхідну економічну оцінку, що є важливим для розуміння масштабів проекту. Всі отримані дані дозволяють внести правки в розробку додатку і сприяють підвищенню його ефективності та зручності для кінцевого користувача.

# ВИСНОВОК

У процесі виконання дипломної роботи було проаналізовано та обгрунтовано роль і значення таких елементів UX/UI дизайну, як Змінних (анг. Variables) та Компонентів (анг. Components), а також їх вплив на створення ефективних та зручних дизайн-систем. В результаті дослідження було виявлено, що ці компоненти є важливими складовими для досягнення консистентності та зручності використання інтерфейсів, підвищують ефективність та продуктивність робити дизайнерів.

У першому розділі роботи були розглянуті основні принципи використання змінних (анг. Variables) та компонентів (анг. Components) у дизайні, а також їх еволюція. Було визначено, що змінні (анг. Variables) використовуються для зберігання повторюваних значень, таких як кольори, шрифти та відступи, що дозволяє забезпечити єдність і швидкість у створенні інтерфейсів.

Компоненти (анг. Components), у свою чергу, є основними будівельними блоками, які допомагають створювати структуровані та зручні інтерфейси, а також скорочують час на розробку за допомогою повторного використання готових елементів.

Крім того, в роботі було детально розглянуто, як сучасні інструменти, зокрема Figma, використовують ці концепції для спрощення роботи дизайнерів і забезпечення високої якості створених інтерфейсів. У цьому контексті було проведено порівняння інструментів Figma, Sketch та Adobe XD, що дозволило з'ясувати їх переваги та недоліки в контексті використання змінних (анг. Variables) та компонентів (анг. Components).

У другому розділі було проведено системний аналіз, розроблено структуру і алгоритми для створення інтерфейсу користувача. Окрему увагу було приділено створенню дерева цілей та дерева проблем, що дозволило наочно візуалізувати завдання та необхідні кроки для досягнення цілей проекту.

На основі цих даних була розроблена архітектура інформації та інтерфейсу додатку, що є основою для подальшого тестування та вдосконалення користувацького досвіду.

В експериментальній частині роботи, на основі розроблених інтерфейсів, було проведено тестування зручності використання додатку, а також збір зворотного зв'язку від користувачів. Виявлені проблеми дозволили запропонувати низку рекомендацій для майбутніх покращень, таких як удосконалення процесу пошуку пісень за допомогою звуків, поліпшення навігації та додавання інтерфейсних підказок для нових користувачів.

Економічна частина роботи дозволила оцінити вартість розробки додатку з точки зору затрат на робочу силу. Загальна вартість проекту, з урахуванням робочих годин для дизайнерів, аналітиків та проджект-менеджерів, склала $4320, що є орієнтовною вартістю для реалізації даного проекту в умовах сучасного ринку. Ці розрахунки допомагають зрозуміти фінансову сторону проекту і врахувати всі можливі витрати на етапі планування.

Таким чином, виконана дипломна робота охоплює всі етапи розробки додатку, від теоретичного аналізу концепцій змінних та компонентів до реального тестування інтерфейсу, розробленого шляхом застосування змінних та компонентів та економічної оцінки проекту. Результати роботи демонструють, як важливо інтегрувати сучасні інструменти та методи в процес розробки, щоб досягти високої якості продукту та зручності для кінцевих користувачів.

# СПИСОК ЛІТЕРАТУРИ


1. Guide to variables in Figma. *Figma Learn - Help Center*. URL: https://help.figma.com/hc/en-us/articles/15339657135383-Guide-to-variables-in-Figma
2. Guide to variables in Figma. *Figma Learn - Help Center*. URL: https://help.figma.com/hc/en-us/articles/360038662654-Guide-to-components-in-Figma
3. Creative Practice. Огляд дизайн-систем. *CASES*. URL: https://cases.media/en/article/oglyad-dizain-sistem?srsltid=AfmBOopBTtDP-cJFrBHz-b6c2w__Q6HdNKQKdMqVLco8SzERCWNrl__y
4. Atomic Design Methodology | Atomic Design by Brad Frost. *Atomic Design by Brad Frost*. URL: https://atomicdesign.bradfrost.com/chapter-2/#consistency
5. Fessenden T. Design Systems 101. *Nielsen Norman Group*. URL: https://www.nngroup.com/articles/design-systems-101/
6. shelest. Дизайн токени. База. *UXPUB 🇺🇦 Дизайн-спільнота*. URL: https://www.nngroup.com/articles/design-systems-101/
7. Components – Material Design 3. *Material Design*. URL: https://m3.material.io/components
8. Atomic Design by Brad Frost. *Atomic Design by Brad Frost*. URL: https://atomicdesign.bradfrost.com
9. Panchenko V. Atomic design system & Design System. *Medium*. URL: https://medium.com/%40vitalka057/atomic-design-system-design-system-874518162c36
10. Turumburum. *Компонентний дизайн: гнучкість, е-комерція і оптимізація ресурсів клієнта*. URL: https://turumburum.ua/blog/komponentnyy-dizayn-gayd-gibkost-e-kommers-i-optimizatsiya-resursov-klienta
11. Norman D. A. Design of Everyday Things. MIT Press, 1998. 270 p.

12. Gordon K. Design Systems vs. Style Guides. *Nielsen Norman Group*. URL: https://www.nngroup.com/articles/design-systems-vs-style-guides/
13. Explore component properties. *Figma Learn - Help Center*. URL: https://help.figma.com/hc/en-us/articles/5579474826519-Explore-component-properties
14. Saini R. S. Design Systems: A Guide to Consistency. *Medium*. URL: https://medium.com/design-bootcamp/design-systems-a-guide-to-consistency-4f8c405b743e
15. 3 Побудова моделі «дерево цілей». *StudFiles*. URL: https://studfile.net/preview/9235071/page:3/
16. 7.4. «Дерево цілей» підприємства - Бібліотека BukLib.net. *Головна - Бібліотека BukLib.net*. URL: https://buklib.net/books/25617/
17. 5.Метод побудови дерева цілей. *StudFiles*. URL: https://studfile.net/preview/9247717/page:3/
18. Laubheimer P. Information Architecture: Study Guide. *Nielsen Norman Group*. URL: https://www.nngroup.com/articles/ia-study-guide/
19. Information Architecture: 3 Key Models (Video). *Nielsen Norman Group*. URL: https://www.nngroup.com/videos/information-architecture-models/
20. Paavan. The big problem with variables in Figma. *Medium*. URL: https://uxdesign.cc/the-big-problem-with-variables-in-figma-61a4ea82640
21. Panchenko V. Atomic design system & Design System. *Medium*. URL: https://medium.com/@vitalka057/atomic-design-system-design-system-874518162c36

# ДОДАТОК А

## Дизайн-система

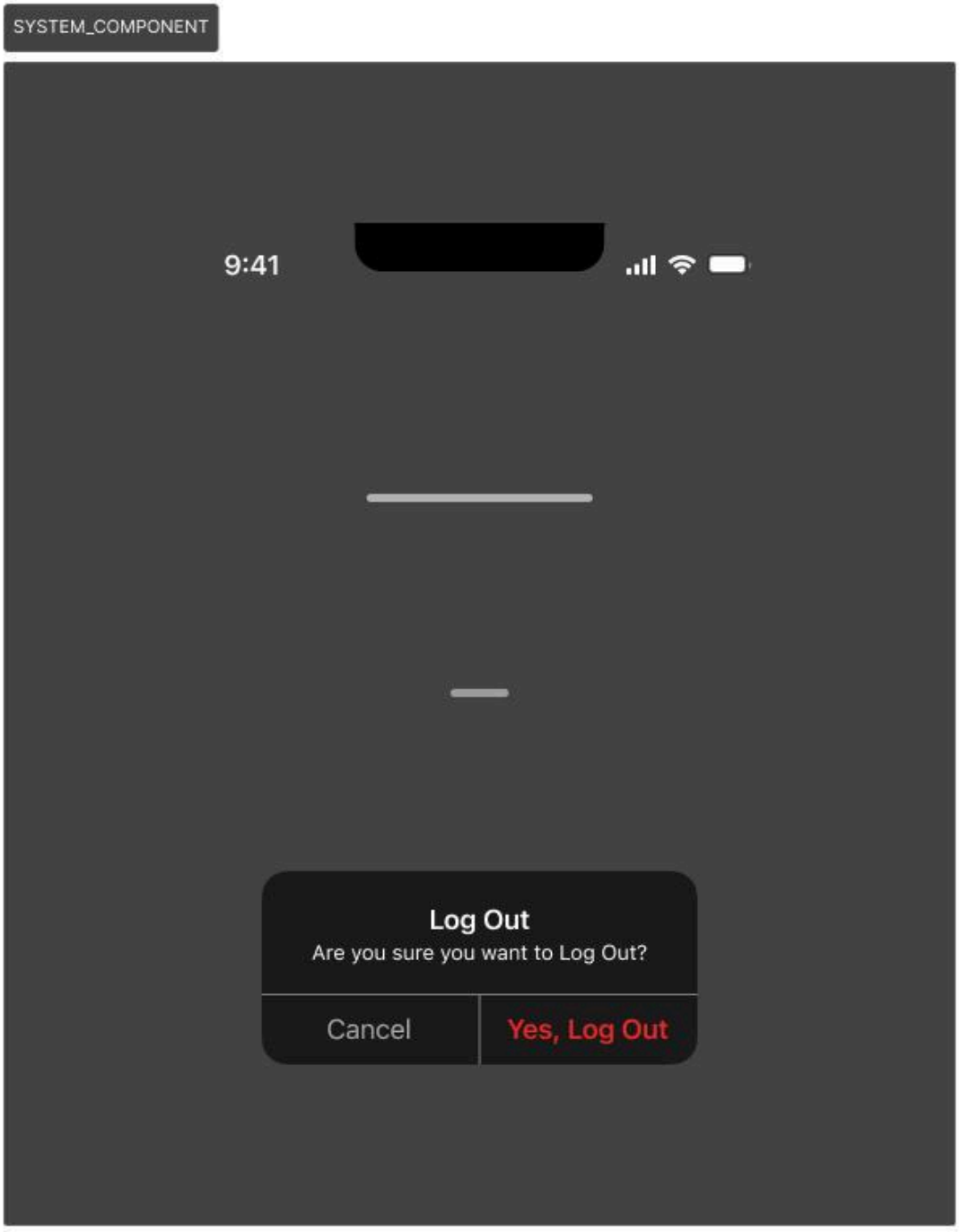


*Рис.А.1. Системні компоненти*

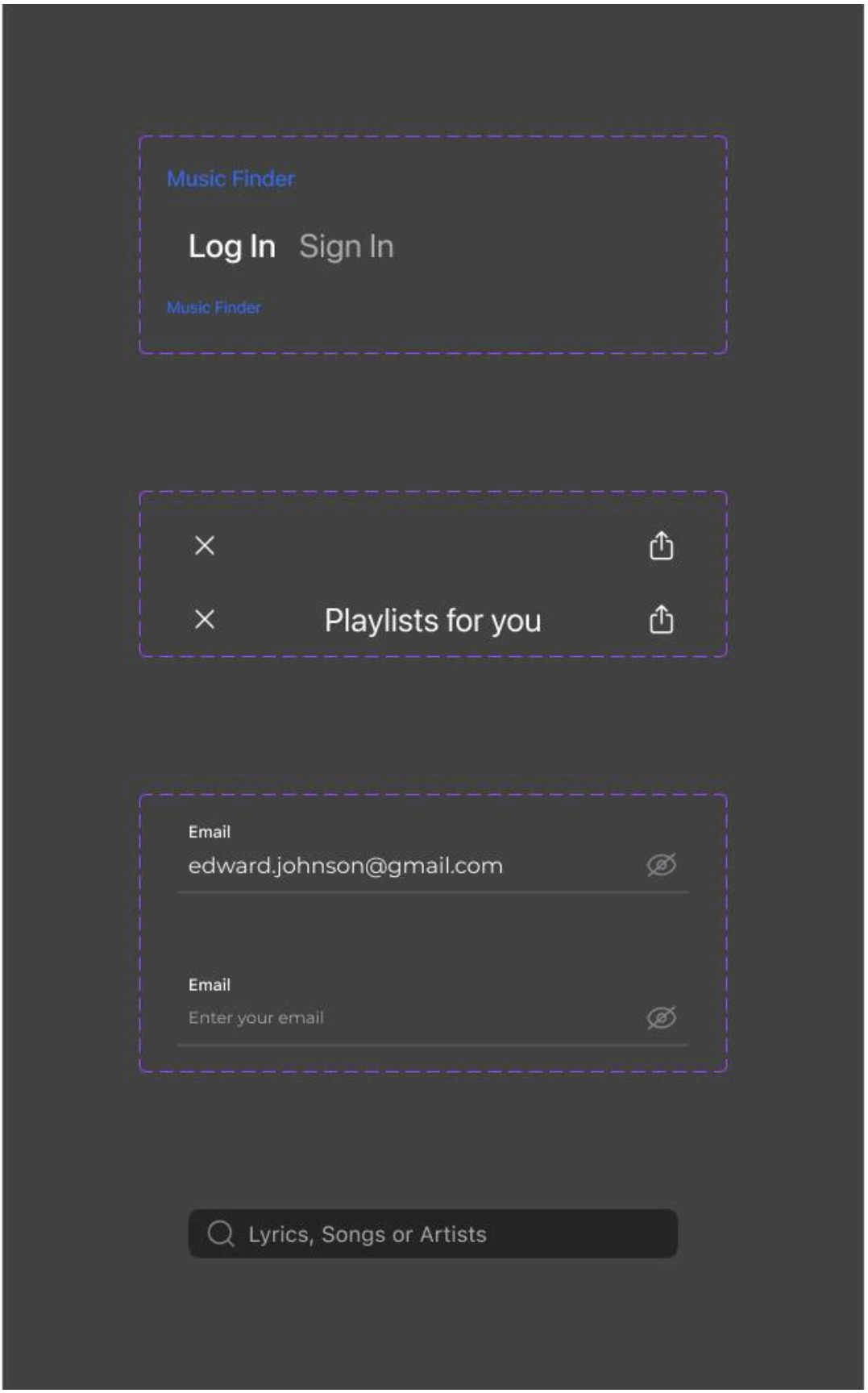


*Рис.А.2. Компоненти*

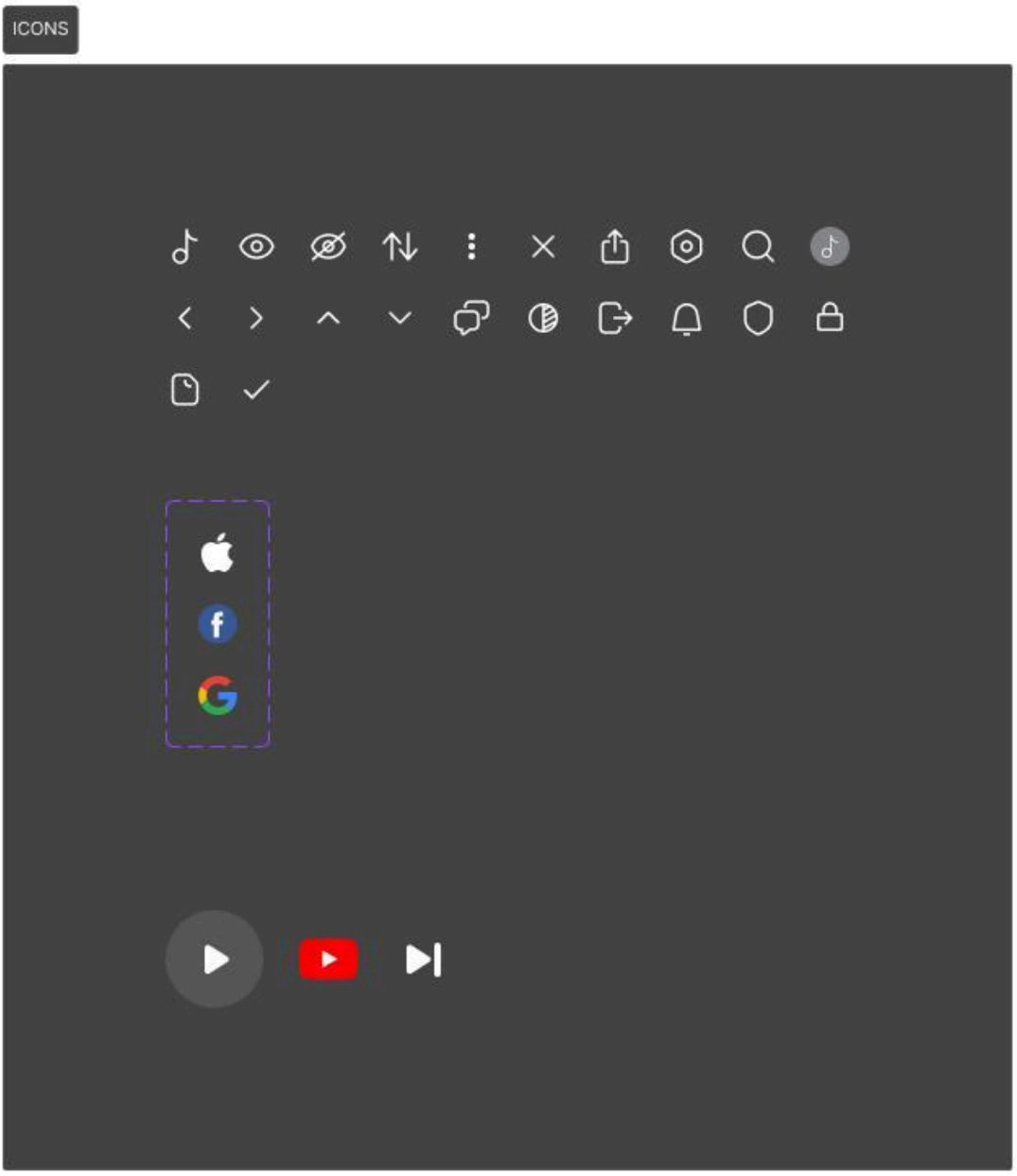


*Рис.А.3. Іконки*

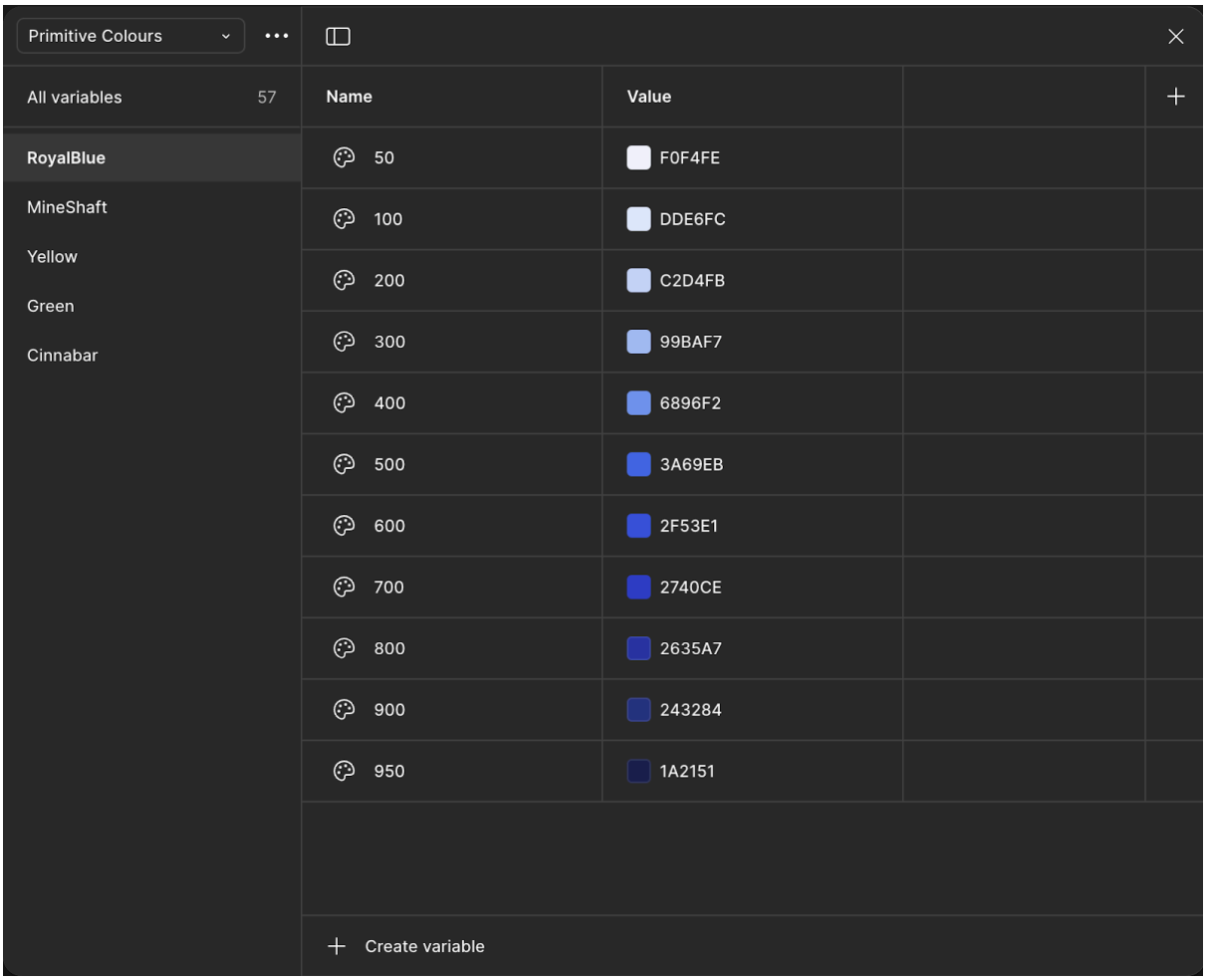


*Рис.А.5. Змінні для кольорів*

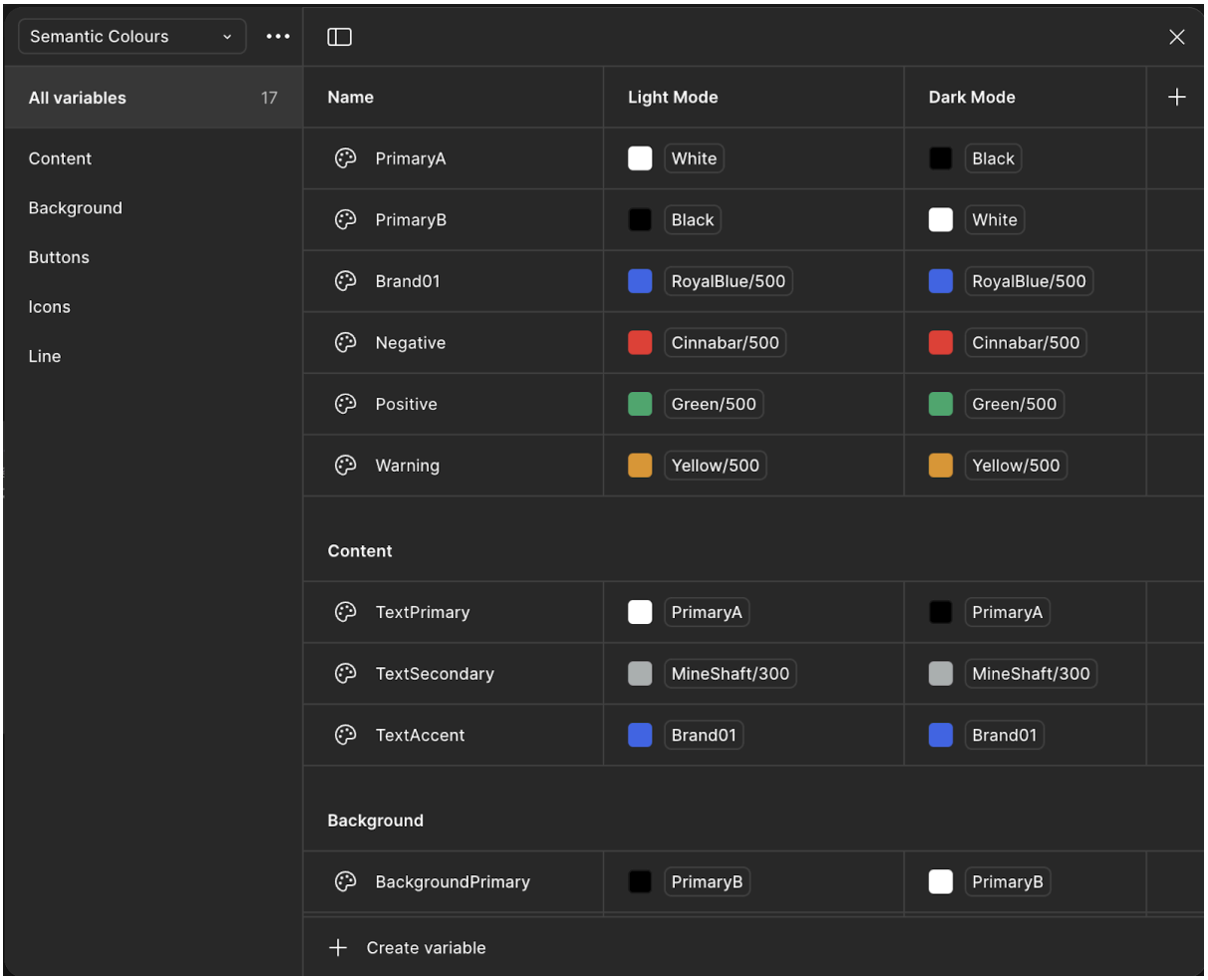


*Рис.А.6. Змінні для кольорів*

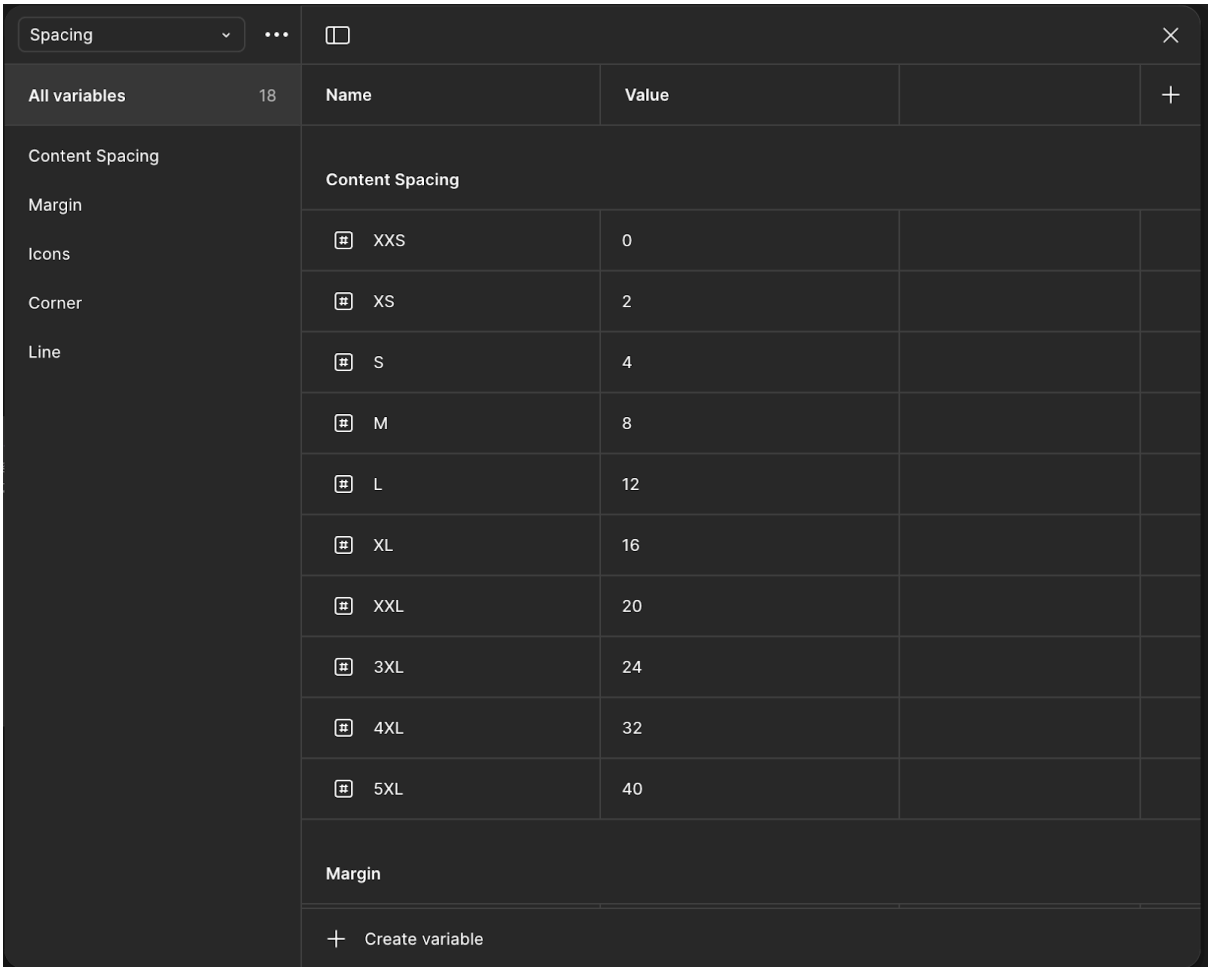


*Рис.А.7. Змінні для відступів*

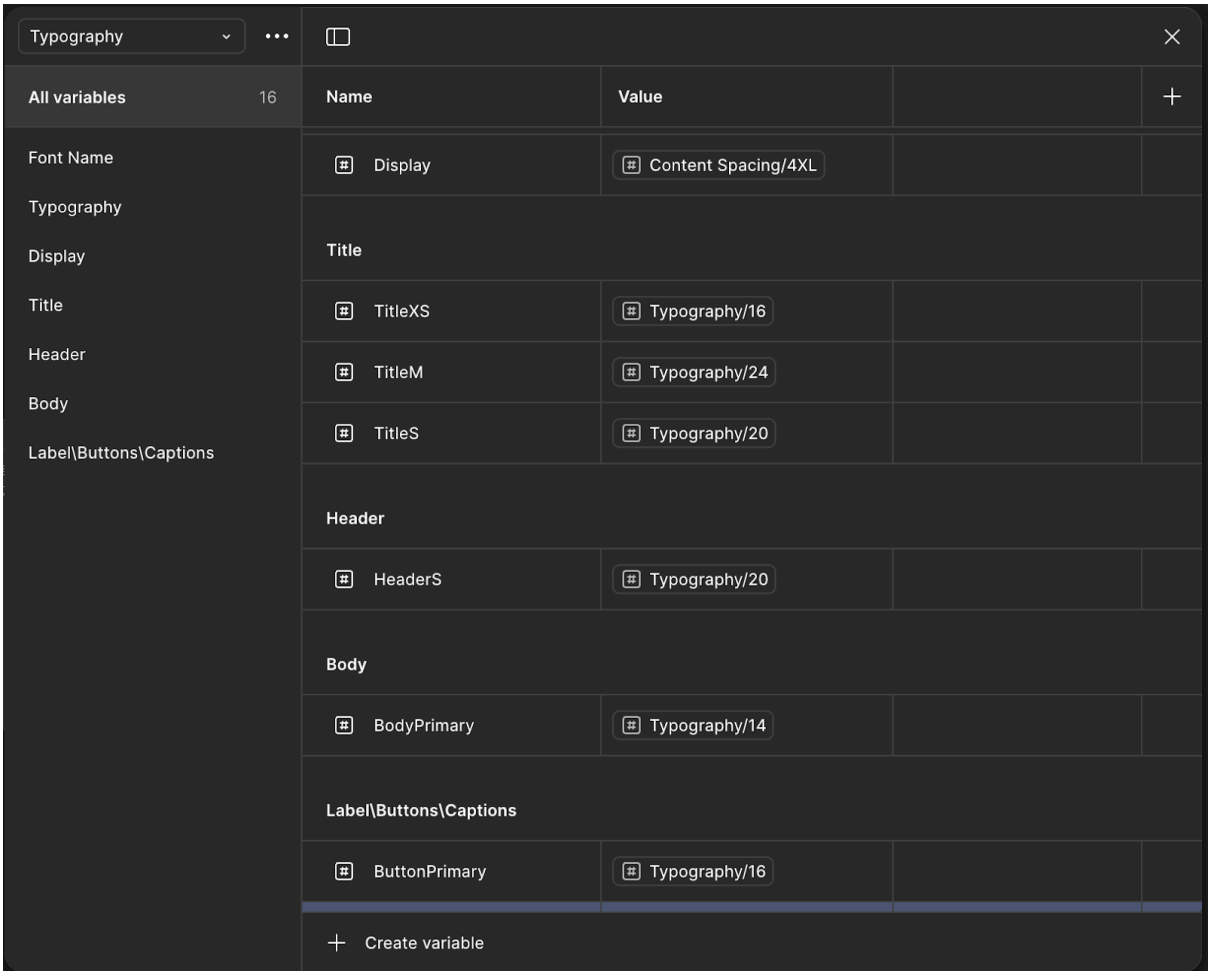


*Рис.А.8. Змінні для шрифтів*