НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)
Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Застосунок з підбору кольорових схем для інтерфейсів

Виконав студент IV курсу, групи ІТ-92
(шифр групи)

Рябко Олександр Володимирович
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач, Вітковська І. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викладач, Вітковська І. І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)

"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Рябку Олександру Володимировичу

(прізвище, ім'я, по батькові)

1. Тема проєкту Застосунок з підбору кольорових схем для інтерфейсів

керівник проєкту Вітковська І. І., ст. викладач

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 31 »квітня 2023 р. №2101-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання _____

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення. _____
2) Моделювання та конструювання програмного забезпечення. _____
3) Аналіз якості та тестування програмного забезпечення. _____
4) Впровадження та супровід програмного забезпечення. _____

5. Перелік графічного матеріалу
1) Схема структурна варіантів використань _____
2) Схема бази даних _____
3) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання	17.03.2023	
3	Постановка та формалізація задачі	18.03.2023	
4	Розробка інформаційного забезпечення	31.03.2023	
5	Алгоритмізація задачі	31.03.2023	
6	Обґрунтування вибору використаних технічних засобів	24.04.2023	
7	Розробка програмного забезпечення	01.05.2023	
8	Налагодження програми	22.05.2023	
9	Виконання графічних документів	22.05.2023	
10	Оформлення пояснювальної записки	31.05.2023	
11	Подання ДП на попередній захист	23.05.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Олександр РЯБКО
(підпис) (ініціали, прізвище)

Керівник _____ Ірина Вітковська
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 44 таблиці, 20 рисунків та 9 джерел – загалом 64 сторінки.

Дипломний проєкт присвячений застосунку з підбору кольорових схем для інтерфейсів.

Мета цього дипломного проекту полягає у покращенні процесу дизайну продуктів за допомогою розробки інтуїтивно зрозумілого мобільного застосунку. Цей застосунок буде забезпечувати автоматичний підбір оптимальних кольорових схем для користувацьких інтерфейсів, що сприятиме зниженню часу та зусиль, потрібних дизайнерам для цього процесу.

Об'єкт дослідження: Мобільний застосунок для підбору кольорових схем для інтерфейсів.

Предмет дослідження: Методи, алгоритми і технічні рішення, які використовуються для підбору, представлення та інтеграції кольорових схем в дизайні інтерфейсів мобільного застосунку.

У розділі першому досліджена предметна область, розглянуті технічні рішення і аналоги, розроблені вимоги.

Розділ другий присвячений моделюванню бізнес-процесів, створенню архітектури і конструюванню програмного забезпечення.

У третьому розділі розглядається аналіз якості програмного забезпечення та описуються процеси тестування.

У четвертому розділі розглядається розгортання і випуск програмного забезпечення.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ЗАСТОСУНОК, КОЛЬОРОВІ СХЕМИ, ДИЗАЙН ІНТЕРФЕЙСУ, МЕТОДИ ПІДБОРУ КОЛЬОРУ, АЛГОРИТМИ ПІДБОРУ КОЛЬОРУ, ТЕХНІЧНІ РІШЕННЯ ДЛЯ ДИЗАЙНУ, ІНТЕГРАЦІЯ КОЛЬОРОВИХ СХЕМ, АВТОМАТИЗОВАНИЙ ПІДБІР КОЛЬОРІВ, ПРОГРАМУВАННЯ МОБІЛЬНИХ ДОДАТКІВ, ВЕБ-РЕСУРСИ ДЛЯ ДИЗАЙНУ, СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ, UX/UI ДИЗАЙН.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 44 tables, 20 figures and 9 sources – in total 64 pages.

The purpose of this diploma project is to improve the product design process by developing an intuitively understandable mobile application. This application will provide automatic selection of optimal color schemes for user interfaces, which will reduce the time and effort required by designers for this process. The evaluation of achieving this goal will be based on user feedback, application usage statistics, and changes in designer productivity.

Research object: A mobile application for selecting color schemes for interfaces.

Research subject: Methods, algorithms, and technical solutions used for selection, representation, and integration of color schemes into the design of the mobile application's interfaces.

In the first section, the subject area is studied, technical solutions and analogs are considered, and requirements are developed.

The second section is dedicated to modelling business processes, creating architecture, and designing software.

The third section discusses the quality analysis of software and describes testing processes.

The fourth section discusses the deployment and release of software.

KEYWORDS: MOBILE APPLICATION, COLOR SCHEMES, INTERFACE DESIGN, COLOR SELECTION METHODS, COLOR SELECTION ALGORITHMS, TECHNICAL DESIGN SOLUTIONS, COLOR SCHEME INTEGRATION, AUTOMATED COLOR SELECTION, MOBILE APPLICATION PROGRAMMING, WEB RESOURCES FOR DESIGN, AUTOMATED DESIGN SYSTEMS, UX/UI DESIGN.

№ з/п	Формат	Формат	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-9222.045440.01.91	Технічне завдання	18	
3	А4	КПІ.ІТ-9222.045440.02.81	Пояснювальна записка	65	
4	А4	КПІ.ІТ-9222.045440.03.12	Текст програми	39	
5	А4	КПІ.ІТ-9222.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9222.045440.05.34	Керівництво користувача	13	
7	А3	КПІ.ІТ-9222.045440.06.99	Графічні матеріали	3	

					КПІ.ІТ-9222.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Рябко О.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Вітковська І.І.					
						КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-92	
Н.контр.		Вітковська І.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК З ПІДБОРУ КОЛЬОРОВИХ СХЕМ ДЛЯ ІНТЕРФЕЙСІВ

Технічне завдання

КПІ.ІТ-9222.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр РЯБКО

Київ – 2023

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Застосунок з підбору кольорових схем для інтерфейсів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку застосунку з підбору кольорових схем для інтерфейсів [КПІ.ІТ-9222.045440.01.91], яке використовується для автоматизації процесу вибору і інтеграції кольорових схем в дизайні інтерфейсів. Дане програмне забезпечення призначено для дизайнерів інтерфейсів, спеціалістів в галузі веб-дизайну, мобільних розробників, а також може бути використано в навчальних цілях для студентів, які вивчають дизайн і програмування.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки застосунку з підбору кольорових схем для інтерфейсів є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації та оптимізації процесу вибору кольорових схем при дизайні інтерфейсів мобільних застосунків. Це включає дизайнерів, веб-розробників, мобільних розробників та студентів відповідних напрямків.

Метою розробки є покращення процесу дизайну продуктів за допомогою розробки інтуїтивно зрозумілого мобільного застосунку. Цей застосунок буде забезпечувати автоматичний підбір оптимальних кольорових схем для користувацьких інтерфейсів, що сприятиме зниженню часу та зусиль, потрібних дизайнерам для цього процесу.

4. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1. Користувацького інтерфейсу

– візуалізація кольорових схем (рис. 4.1);

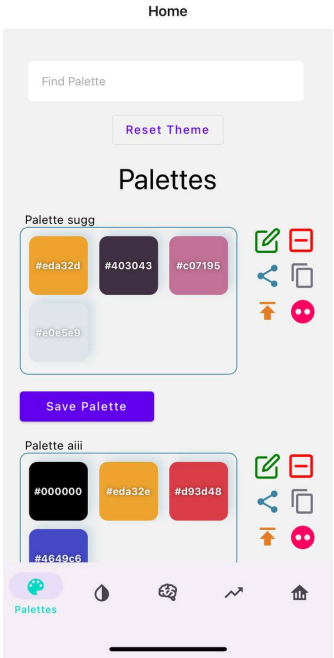


Рис. 4.1 - Візуалізація кольорових схем

– підбір кольорових схем (рис. 4.2);

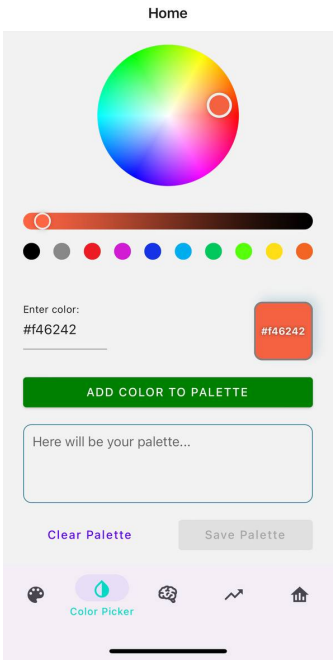


Рис. 4.2 - Підбір кольорових схем

– оповіщення користувача про будь-які події (рис. 4.3 - помилка, рис. 4.4 - копіювання кольору);

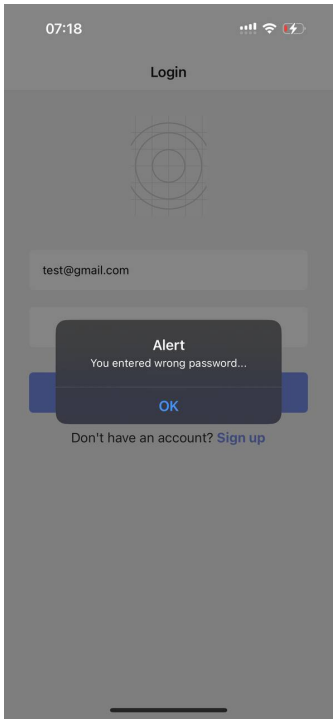


Рис. 4.3 - Оповіщення про помилку

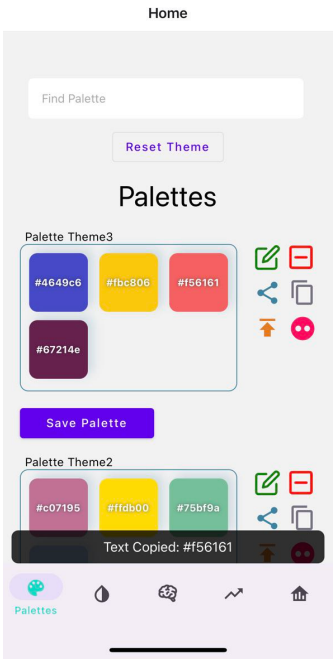


Рис. 4.4 - Оповіщення про копіювання кольору

– збереження та експорт кольорових палітр (рис. 4.5);

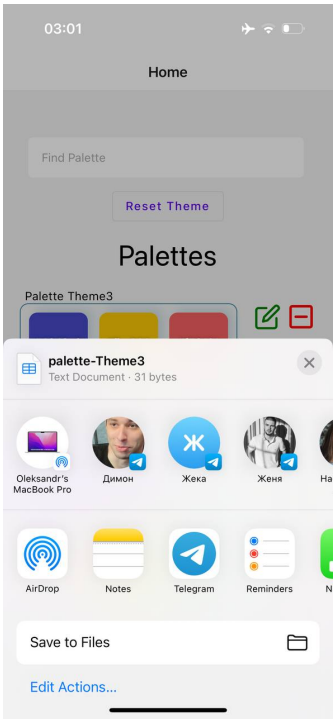


Рис. 4.5 - Збереження та експорт кольорової палітри

– регістрацію (рис. 4.6) та авторизацію користувача (рис. 4.7);

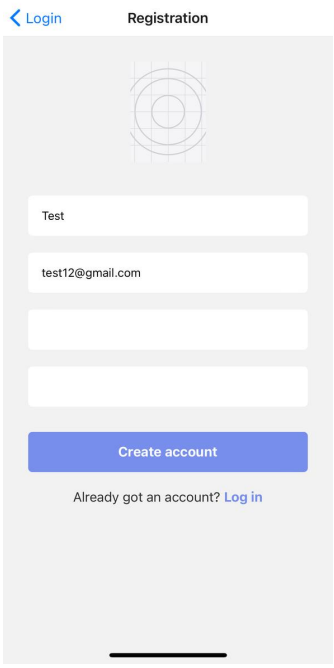


Рис. 4.6 - Регістрація користувача

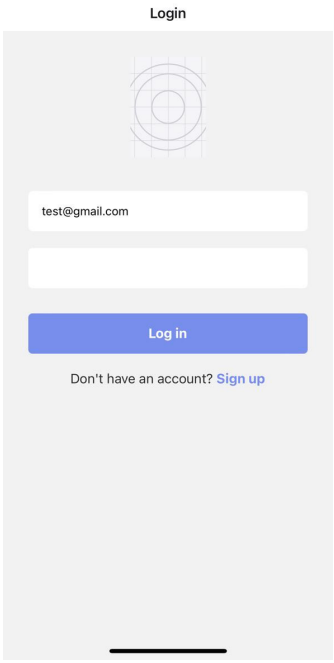


Рис. 4.7 - Авторизація користувача

– аналіз кольору (рис. 4.8):

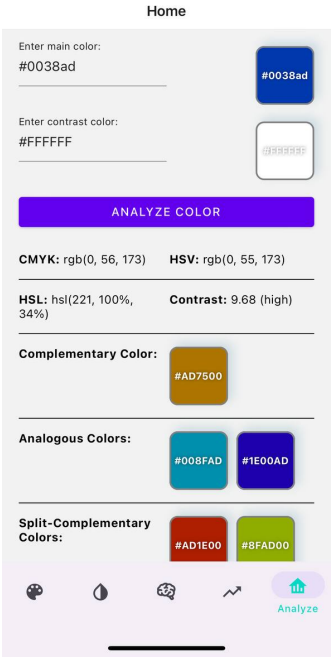


Рис. 4.7 - Аналіз кольору

– тренди кольорових палітр від інших користувачів (рис. 4.8);

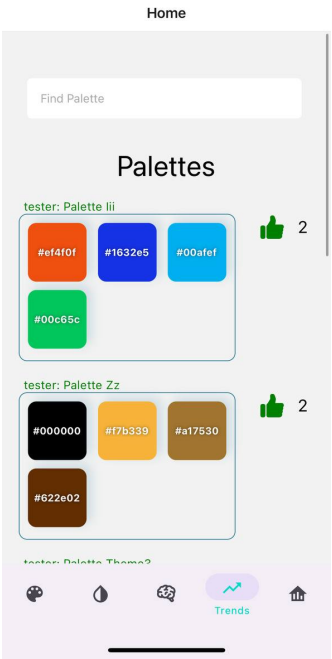


Рис. 4.8 - Аналіз кольору

– можливість застосувати кольорову палітру до інтерфейсу (рис. 4.9);

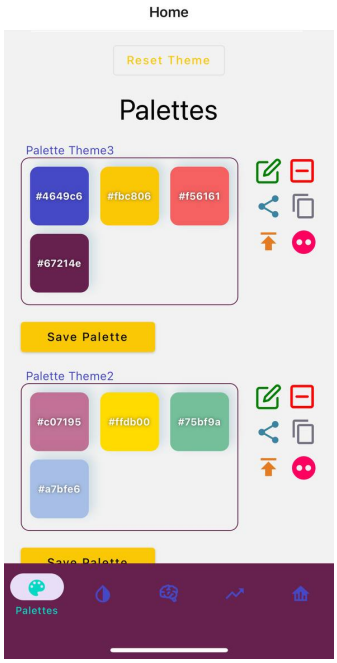


Рис. 4.9 - Інтерфейс після застосування кольорової палітри

– автоматичну генерація кольорових палітр (рис. 4.10).

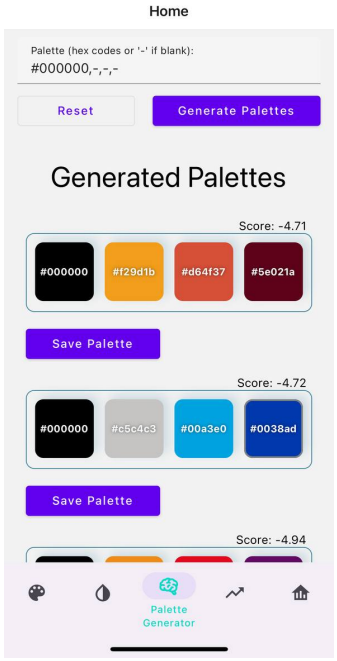


Рис. 4.10 - Автоматично згенеровані кольорові палітри

4.1.2. Для користувача:

– реєстрація користувача за допомогою пошти і пароля;
– авторизація користувача за допомогою пошти і паролю;
– створення та редагування кольорових схем;
– перегляд історії попередньо створених палітр:
– пошук кольорових схем;
– публікування кольорової схеми до трендів;
– застосування до або відміна кольорової схеми з інтерфейсу;
– копіювання окремого кольору або всієї палітри;
– експорт кольорових схем;
– додавання до або видалення кольорової схеми з історії;
– аналіз обраного кольору, включаючи можливість обрати до нього контрастний колір;
– переглядати та вподобати кольорові палітри від інших користувачів;
– генерувати за допомогою штучного інтелекту палітри кольорів за вказаним основним кольором.

4.1.4. Додаткові вимоги:

– сумісність: додаток має бути сумісний з основними мобільними операційними системами, включаючи Android та iOS; це забезпечує доступ до широкої аудиторії користувачів;
– безпека: додаток має включати відповідні механізми захисту даних користувачів, включаючи шифрування персональних даних та обмеження доступу до облікового запису користувача;
– продуктивність: додаток має забезпечувати швидку та плавну роботу, з мінімальними затримками або зависаннями;
– ергономічність: інтерфейс користувача має бути простим, інтуїтивно зрозумілим і приємним для користування, щоб сприяти використанню додатку;

– офлайн режим: додаток повинен мати можливість працювати в офлайн режимі, дозволяючи користувачам працювати над своїми кольоровими схемами без підключення до інтернету.

4.2. Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

В разі виникнення помилок або збоїв, система повинна надавати зрозумілі повідомлення про проблеми і способи їх вирішення. Повинен бути реалізований механізм збереження цілісності даних в БД при виконанні операцій модифікації.

4.3. Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1. Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2. Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4. Вимоги до складу і параметрів технічних засобів

Застосунок з підбору кольорових схем для користувацьких інтерфейсів вимагає специфічного обладнання для своєї оптимальної роботи. Оскільки річ про мобільний додаток, замість особистого комп'ютера, основні вимоги пов'язані з мобільним пристроєм користувача.

Мінімальна конфігурація технічних засобів:

– тип пристрою: Смартфон або планшет;
– операційна система: Android 7.0+ або iOS 11.0+;
– об'єм оперативної пам'яті: 2 Гб;

– внутрішня пам'ять: 100 Мб вільного місця для встановлення та зберігання даних додатка;

– підключення до мережі Інтернет: 3G, 4G, Wi-Fi.

Рекомендована конфігурація технічних засобів:

– тип пристрою: Смартфон або планшет;

– операційна система: Android 10.0+ або iOS 13.0+;

– об'єм оперативної пам'яті: 4 Гб;

– внутрішня пам'ять: 500 Мб вільного місця для встановлення та зберігання даних додатка;

– підключення до мережі Інтернет: 4G, 5G, Wi-Fi.

4.5. Вимоги до інформаційної та програмної сумісності

Мобільний додаток з підбору кольорових схем для користувацьких інтерфейсів повинен працювати під управлінням операційних систем сімейства Android версії 7.0 та вище та iOS версії 11.0 та вище.

4.5.1. Вимоги до вхідних даних

Вхідні дані повинні бути представлені в форматі JSON (JavaScript Object Notation), який складається з пар "ключ-значення". Дані користувача і вибрані ними кольорові схеми зберігаються і обробляються за допомогою цього формату.

4.5.2. Вимоги до вихідних даних

Результати повинні бути представлені в тому ж форматі JSON, що дозволяє зберігати та обробляти кольорові схеми, вибрані користувачами, для подальшого використання або модифікації в додатку.

4.5.3. Вимоги до мови розробки

Розробку виконати на мовах програмування: JavaScript, Node.js.

4.5.4. Вимоги до середовища розробки

Розробку виконати на за допомогою середовищ: XCode, VS Code та Expo Go.

4.5.5. Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді .js файлів.

4.6. Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7. Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8. Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1. Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

- пояснювальна записка;
- технічне завдання;
- текст програми;
- програма та методика тестування;
- керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

- схема структурна варіантів використання;
- схема бази даних;
- креслення вигляду екранних форм.

5.2. Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6. СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Застосунок з підбору кольорових схем для інтерфейсів

КПІ.ІТ-9222.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE – Integrated Development Environment – інтегроване середовище розробки.

API – Application programming interface, прикладний програмний Інтерфейс

SDK – Software development kit

ІТ – Інформаційні технології

БД – База даних.

ВСТУП

Мобільні додатки стають все більш важливою частиною нашого життя, оскільки вони значно спрощують різноманітні аспекти нашої повсякденної діяльності. Відповідно, актуальність розробки інтуїтивно зрозумілого мобільного додатку для автоматичного підбору кольорових схем для користувацьких інтерфейсів є досить високою. Враховуючи розширення діапазону задач, які дизайнери вирішують в сучасних умовах, цей проект може знизити час та зусилля, потрібні дизайнерам для цього процесу, і покращити якість дизайну продуктів.

У світі існують відомі тенденції стосовно використання AI і машинного навчання для підбору кольорових схем. Наприклад, компанії, як Google та Adobe, активно використовують ці технології для поліпшення досвіду своїх користувачів. Однак, не дивлячись на це, потреба в розробці інтуїтивно зрозумілих інструментів для підбору кольорових схем залишається актуальною.

Оцінка сучасного стану об'єкта розробки свідчить, що деякі завдання вже були розв'язані провідними науковими установами та організаціями. Наприклад, Adobe Color CC - це інструмент для вибору кольорових схем, який вже використовується дизайнерами по всьому світу. Однак, цей інструмент все ще вимагає певного рівня професійних знань і навичок, щоб ефективно використовувати його.

Додаток ефективно спростить ручний підбір кольорів, що раніше вимагав багато часу та зусиль. Він надасть можливість швидко створювати привабливі кольорові схеми, які відповідають дизайнерським ідеям та задовольняють потреби користувачів. Використання додатка дозволить ефективно використовувати час і ресурси, покращувати якість дизайну продуктів та створювати позитивне враження від їх використання. Користувачі зможуть швидко втілювати свої ідеї та концепції стосовно кольорових палітр, а дизайнери зосередяться на творчих аспектах своєї роботи, задовольняючи потреби своїх клієнтів.

1. АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Загальні положення

Вибір колірних схем для інтерфейсів є вирішальним аспектом дизайну програми та має значний вплив на взаємодію з користувачем. Колір відіграє важливу роль у передачі інформації, створенні настрою та покращенні зручності використання. Однак вибір відповідної колірної схеми може бути складним через суб'єктивний характер сприйняття кольору та необхідність враховувати різні чинники, такі як естетичність, доступність та ідентичність бренду [9].

Колірна гармонія та контраст є основними принципами дизайну інтерфейсу. Створення візуально приємної колірної схеми передбачає вибір кольорів, які доповнюють один одного та забезпечують достатній контраст для читабельності та чіткості. Невдалий вибір кольорів може призвести до заплутаності або надмірності інтерфейсу, що перешкоджає взаємодії та розумінню користувача.

Крім того, доступність є ключовим фактором у дизайні інтерфейсу. Дальтонізм вражає значну частину населення, тому вкрай важливо розробляти інтерфейси, які є інклюзивними та доступними для всіх користувачів. Вибір кольорів, які можуть легко розрізняти люди з вадами колірного зору та можливість самому обирати кольорову палітру для інтерфейсу, є важливими для забезпечення оптимальної взаємодії з користувачем [8].

Іншим аспектом, який слід враховувати, є культурні та психологічні кольоріві асоціації. Різні культури та люди можуть по-різному трактувати та емоційно сприймати кольори. Розуміння цих асоціацій і підбір колірних схем відповідно може допомогти викликати бажані емоції та створити гармонійний візуальний досвід.

Для вирішення проблем вибору колірних схем для інтерфейсів були розроблені різні підходи та інструменти. Принципи теорії кольорів, такі як колірне коло, можуть керувати дизайнерами у створенні збалансованих і

естетично приємних комбінацій. Генератори колірної палітри та інструменти візуалізації допомагають досліджувати різні комбінації кольорів і оцінювати їх придатність для конкретних інтерфейсів.

Загалом вибір колірних схем для інтерфейсів є багатогранним завданням, яке вимагає глибокого розуміння принципів дизайну, уподобань користувачів, міркувань доступності, культурних факторів і технологічних досягнень. Застосовуючи продуманий та інформований підхід, дизайнери можуть створювати візуально привабливі, доступні та привабливі інтерфейси, які покращують взаємодію з користувачем.

1.2. Змістовний опис і аналіз предметної області

При розробці застосунку з підбору кольорових схем для інтерфейсів важливо враховувати кілька критеріїв при виборі відповідного контейнера даних. У процесі відбору слід керуватися такими факторами:

– гнучкість: застосунок повинен забезпечувати різноманітне представлення кольорів і підтримувати різні формати, які зазвичай використовуються в дизайні інтерфейсу;

– доступність: застосунок має бути легко доступним і зручним для користувача, забезпечуючи простий інтерфейс для користувачів, щоб орієнтуватися та експериментувати з різними комбінаціями кольорів. Інтуїтивно зрозумілі елементи керування та чіткі візуальні представлення покращують взаємодію з користувачем і полегшують ефективний вибір колірної схеми.

1.3. Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку з підбору кольорових схем для інтерфейсів. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1. Аналіз допоміжних програмних засобів та засобів розробки

У процесі розробки мобільних додатків, важливою частиною є вибір відповідних інструментів, що можуть сприяти швидкій, гнучкій та ефективній розробці. Для розробки додатку на React Native [1] будуть використані наступні інструменти:

– XCode: XCode - це IDE від компанії Apple. У контексті розробки застосунку на React Native, XCode використовується переважно для збірки, тестування та відлагодження версії застосунку для iOS. Він також містить інструменти для роботи з інтерфейсом користувача, версій контролю, та включає симулятор iOS, що дозволяє виконувати додаток на віртуальному пристрої безпосередньо на робочій станції розробника;

– Visual Studio Code: Visual Studio Code (VS Code) - це відкритий та безкоштовний текстовий редактор від Microsoft. Він підтримує численні мови програмування, включаючи JavaScript і TypeScript, які використовуються у React Native. VS Code має багатий набір функцій, таких як підсвічування синтаксису, автозавершення коду, рефакторинг, вбудований термінал, Git інтеграція, а також можливість розширення за допомогою плагінів, що робить його ідеальним рішенням для розробки React Native застосунків;

– Expo Go [2]: Expo Go - це додаток, яке працює на Android і iOS пристроях, і дозволяє відкривати React Native проекти, які використовують Expo. Воно забезпечує зручний спосіб перегляду та тестування розроблюваного додатку безпосередньо на реальних пристроях без необхідності збірки сторонніми інструментами. Expo Go забезпечує доступ до всіх Expo SDK бібліотек, що дозволяє легко протестувати нові функції та зміни.

Комбінація цих інструментів забезпечує повноцінну підтримку всіх необхідних етапів розробки мобільного додатку - від написання коду до його відлагодження, тестування та випуску. XCode, Visual Studio Code та Expo Go володіють широким набором можливостей, що робить їх незамінними у роботі над проектами на React Native.

1.3.2. Аналіз відомих програмних продуктів

Після пошуку застосунків з підбору кольорових схем для інтерфейсів було відокремлено 2 застосунки, а саме:

– color wheel;

– idori.

Color wheel – це застосунок який допомагає своїм користувачам підібрати кольори за допомогою надання комплементарних кольорів, аналогічних кольорів, триад, тетрад та інших схем схеми.

Інтерфейс користувача зображено на рисунку 1.1:

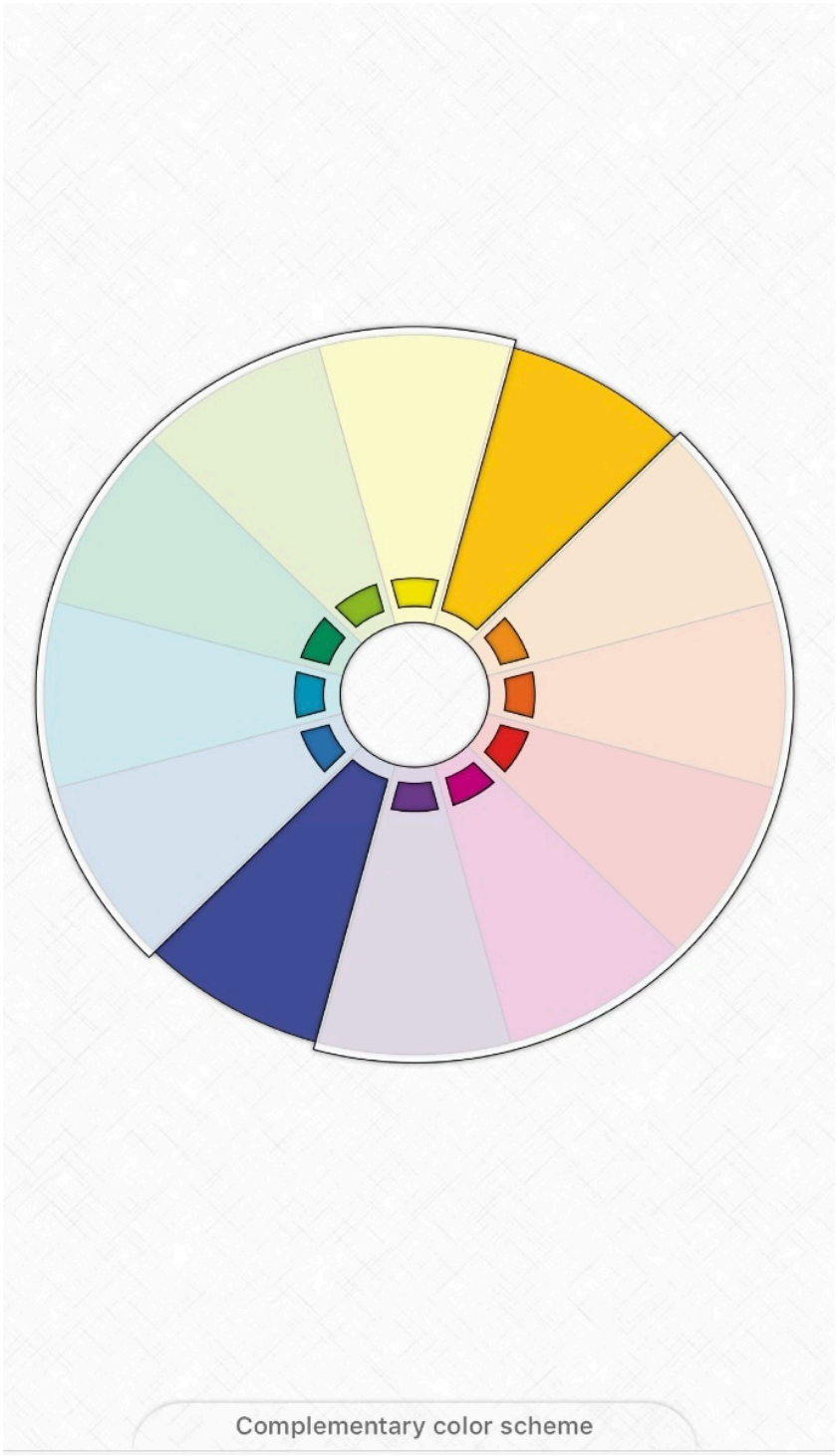


Рисунок 1.1 – Інтерфейс користувача застосунку Color wheel

Переваги:

– застосунок надає багато схем для створення інтерфейсу користувача.

Недоліки:

– застосунок має дуже обмежену функціональність;

– застосунок платний;

– не надає рекомендацій по використанню кольорів.

idori – застосунок, також надає користувачам можливість підібрати кольори, але за допомогою: зміни значень RGB, введенням HEX-значення, вибору кольору з вбудованої палетки кольорів, а також має можливість створення градієнтів. Інтерфейс користувача зображено на рисунку 1.2:

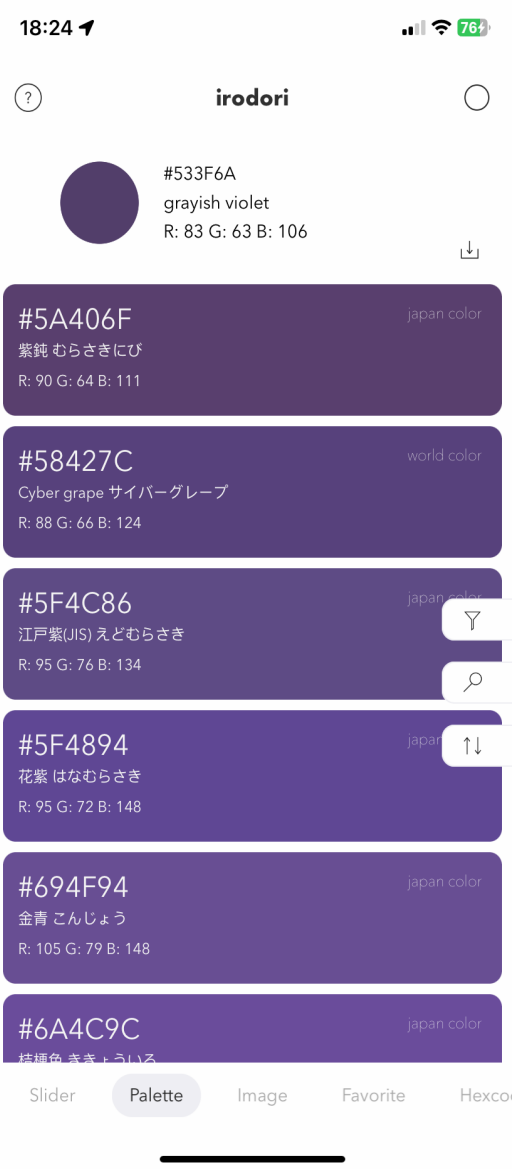


Рисунок 1.2 – Інтерфейс користувача застосунку idori

Переваги:

– можливість підбору кольорів;

– зручний та зрозумілий інтерфейс користувача;

– можливість підбору кольорів за допомогою фото.

Недоліки:

– немає трендів та рекомендацій для кольорів та кольорових схем;

– досить погана оптимізація на моделях телефонів, які трішки застарілі;

– немає порівняння контрастності.

Далі наведено таблицю порівняння:

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт(Palette Selector)	Color wheel	idori	Пояснення
Перевірка контрастності	+	-	-	Застосунок має мати можливість перевіряти контрасність кольорів
Підбір кольору	+	+	+	Застосунок надає можливість підібрати колір
Збереження даних	+	-	+	Застосунок надає можливість збереження згенерованого кольору

Продовження таблиці 1.1

Експорт кольорових схем	+	-	+	Застосунок надає можливість експортувати кольорову схему

1.4. Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є підбір кольорів, більше функцій можна побачити на рисунку 1.3:

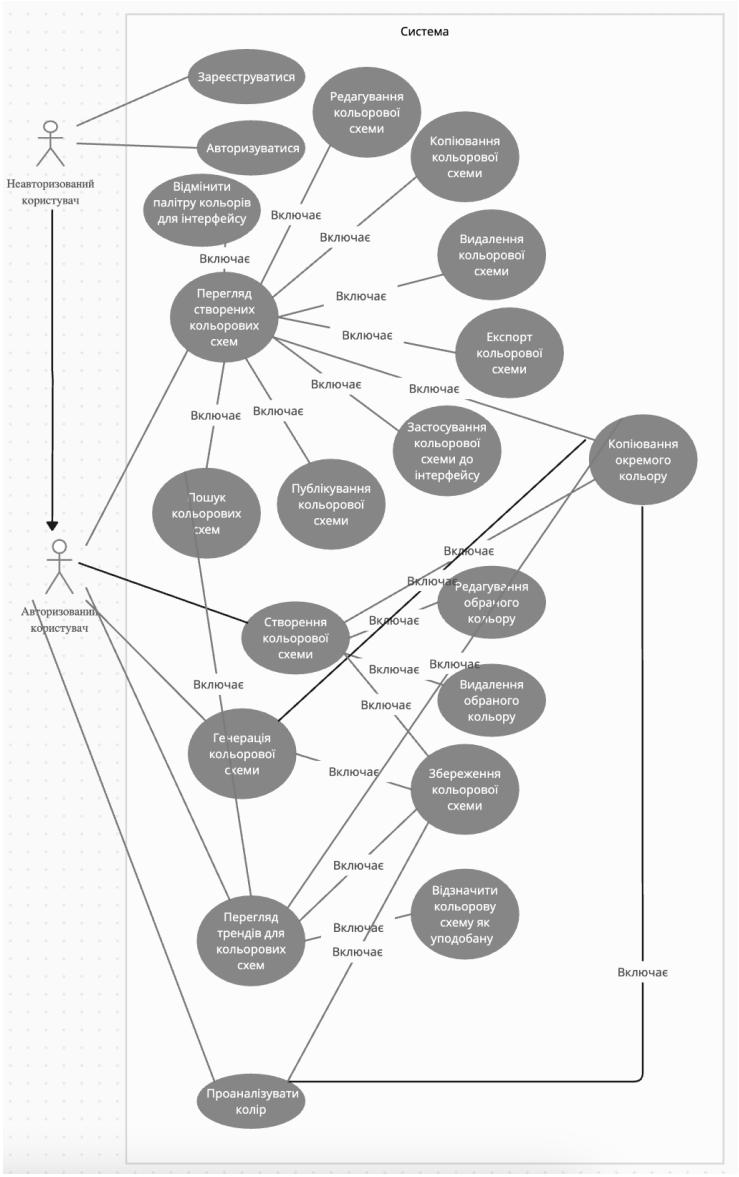


Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 - 1.9 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

Use case name	Підбір індивідуального кольору або кольорової палітри
Use case ID	UC-01
Goals	Підбір кольору або палітри
Actors	Користувач

Продовження таблиці 1.2

Trigger	Користувач бажає підібрати кольорову палітру або індивідуальний колір
Pre-conditions	-
Flow of Events	Користувач переходить на екран аналізу чи підбору кольору та обирає вподобані варіанти.
Extension	
Post-condition	Перехід на екран який відповідає за відображення кольорових палітр користувача.

Таблиця 1.3 – Варіант використання UC-2

Use case name	Підбір кольору
Use case ID	UC-02
Goals	Підбір кольору
Actors	Користувач
Trigger	Користувач бажає підібрати колір
Pre-conditions	-
Flow of Events	Користувач має змогу підібрати колір за допомогою: колірного кола, повзунків RGB та поля вводу HEX - значень
Extension	У випадку неправильного введення значень в поле HEX поле підсвічується червоним
Post-condition	Підібраний колір

Таблиця 1.4 – Варіант використання UC-3

Use case name	Завантаження кольорової схеми та її редагування
Use case ID	UC-03
Goals	Завантажити кольорову схему до застосунку
Actors	Користувач
Trigger	Користувач бажає завантажити кольорову схему до застосунку
Pre-conditions	-
Flow of Events	Після того як користувач підібрав кольорову схему та закінчив роботу з нею, він має змогу зберегти її, ввівши ім’я та натиснувши на кнопку збереження.
Extension	При помилці зберігання користувачу показується повідомлення про збій та пропонується повторити спробу
Post-condition	Перехід до головного екрану

Таблиця 1.5 – Варіант використання UC-4

Use case name	Генерація кольорів
Use case ID	UC-04
Goals	Згенерувати новий колір на основі вибраних кольорів
Actors	Застосунок
Trigger	Користувач хоче щоб застосунок допоміг йому згенерувати новий колір
Pre-conditions	-
Flow of Events	Користувач переходить на екран генерації випадкового кольору та вибирає основні кольори на основі яких застосунок згенерує колір.
Extension	-
Post-condition	Відображення згенерований колір та його дані в форматі RGB та HEX

Таблиця 1.6 – Варіант використання UC-5

Use case name	Збереження даних
Use case ID	UC-05
Goals	Зберегти дані користувача, а саме створені кольорові схеми
Actors	Користувач
Trigger	Користувач закінчив роботу над кольоровою схемою
Pre-conditions	-
Pre-conditions	
Flow of Events	Після закінчення та збереження роботи користувач може переглянути як кольорова схема виглядатиме на різних елементах
Extension	
Post-condition	Перехід до екрану збереження роботи

Таблиця 1.7 – Варіант використання UC-6

Use case name	Експорт кольорових схем
Use case ID	UC-06
Goals	Експортування створених кольорових схем
Actors	Користувач
Trigger	Користувач бажає поділитись кольоровою схемою
Pre-conditions	
Flow of Events	Після закінчення роботи з кольоровою схемою користувач має змогу експортувати її за допомогою створення файлу з розширенням .csv або .json.
Extension	При помилці експортування користувачу показується повідомлення про збій та пропонується повторити спробу
Post-condition	Перехід до головного екрану

Таблиця 1.8 – Варіант використання UC-7

Use case name	Попередній огляд
Use case ID	UC-07
Goals	Дозволити користувачу попередньо переглянути кольорову схему
Actors	Застосунок
Trigger	Користувач хоче переглянути як кольорова схема виглядатиме на різних елементах
Flow of Events	На головному екрані користувач вибирає пункт завантаження кольорової схемм до застосунку для її подальшого редагування.
Extension	
Post-condition	Створення нової кольорової схеми

Таблиця 1.9 – Варіант використання UC-8

Use case name	Перевірка котрасності
Use case ID	UC-08
Goals	Надати користувачеві змогу перевірити контрастність кольорів
Actors	Користувач
Trigger	Користувач хоче перевірити контрасність кольорів
Pre-conditions	Користувач підібрав 2 кольори
Flow of Events	Після того як користувач підібрав кольори він має змогу перевірити їх контрастність в застосунку та вирішити чи продовжувати йому роботу саме з вибраними кольорами
Extension	Якщо вибрано лише один колір користувачу відображається помилка
Post-condition	Користувач продовжує роботу з кольорами

1.4.1. Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 1.10 наведена загальна модель вимог до застосунку.

Таблиця 1.10 – Загальна модель вимог

№	Назва функціональної вимоги	Пріорітет	Ризик	ID вимоги
1	Підбір кольорових схем	1	-	FR-1
1.1	Перевірка контрастності	-	-	FR-2
1.2	Збереження та завантаження схем	-	Середній	FR-3
2	Генерація варіацій кольорів	-	-	FR-4
2.1	Імпорт та експорт кольорових схем	-	Середній	FR-5
3	Аналіз кольору	-	-	FR-6
4	Перетягування та підсвічування	-	-	FR-7
5	Рекомендації та тенденції	-	-	FR-8
6	Перегляд результатів у реальному часі	-	-	FR-9
7	Попередній перегляд на різних елементах інтерфейсу	2	-	FR-10

На рисунку 1.4 наведено матрицю трасування, а в таблицях 1.11 – 1.20 наведений опис функціональних вимог до програмного забезпечення.

Таблиця 1.11 – Функціональна вимога FR-1

Назва	Підбір кольорових схем
Опис	Застосунок повинен надати можливість створювати та переглядати кольорові схеми, які складаються з кількох взаємно гармонійних кольорів. Можливі варіанти підбору кольорових схем можуть включати комплементарні кольори, аналогічні кольори, триади, тетради та інші схеми.

Таблиця 1.12 – Функціональна вимога FR-2

Назва	Перевірка контрастності
Опис	Застосунок може надати функцію перевірки контрастності між кольорами. Це допоможе користувачам створювати кольорові схеми, які забезпечують достатню контрастність для людей з різними рівнями зорової спроможності.

Таблиця 1.13 – Функціональна вимога FR-3

Назва	Збереження та завантаження схем
Опис	Застосунок повинен забезпечувати можливість зберігати створені кольорові схеми та завантажувати їх для майбутнього використання. Користувач повинен мати можливість організовувати та категоризувати свої схеми.

Таблиця 1.14 – Функціональна вимога FR-4

Назва	Генерація варіацій кольорів
Опис	Застосунок може пропонувати функцію автоматичної генерації варіацій кольорів на основі вибраного початкового кольору або кольорової схеми. Це може бути корисно для отримання додаткових варіацій для певного кольору чи схеми.

Таблиця 1.15 – Функціональна вимога FR-5

Назва	Імпорт та експорт кольорових схем
Опис	Застосунок може надати можливість імпортувати зовнішні кольорові схеми з інших форматів, таких як HEX-коди, RGB-значення, або файлів кольорових палітр, щоб користувачі могли легко використовувати свої власні схеми. Крім того, зручною функцією може бути можливість експортувати створені кольорові схеми для використання в інших програмах або проектах.

Таблиця 1.16 – Функціональна вимога FR-6

Назва	Аналіз кольору
Опис	Застосунок може мати можливість аналізувати кольори, включаючи отримання інформації про їх HEX-коди, RGB-значення, відтінки, насиченість, яскравість тощо. Це може допомогти користувачам зрозуміти та працювати з вибраними кольорами більш ефективно.

Таблиця 1.17 – Функціональна вимога FR-7

Назва	Перетягування та підсвічування
Опис	Застосунок має забезпечити можливість перетягувати та переносити кольори між різними об'єктами або компонентами інтерфейсу. Крім того, підсвічування вибраних кольорів чи схем може полегшити їхнє використання та навігацію в рамках застосунка.

Таблиця 1.18 – Функціональна вимога FR-8

Назва	Рекомендації та тенденції
Опис	Застосунок може надати користувачам рекомендації щодо популярних кольорових схем або трендів в дизайні інтерфейсів. Це може стимулювати творчість та допомогти користувачам залишатися в тренді з актуальними кольоровими комбінаціями.

Таблиця 1.19 – Функціональна вимога FR-9

Назва	Перегляд результатів у реальному часі
Опис	Застосунок може надати можливість перегляду кольорових схем у режимі реального часу. При зміні кольору або складу схеми користувач може одразу бачити, як це впливає на вигляд інтерфейсу. Це допомагає швидко експериментувати та відстежувати зміни.

Таблиця 1.20 – Функціональна вимога FR-10

Назва	Попередній перегляд на різних елементах інтерфейсу
Опис	Застосунок може надати можливість попереднього перегляду кольорових схем на різних типах елементів інтерфейсу, таких як кнопки, текст, фони тощо. Це допомагає користувачам оцінювати, як кольорова схема виглядає на реальних елементах

Після детального опису функціональних вимог до програмного продукту, наступним кроком є їх систематизація і встановлення зв'язків між ними та відповідними випадками використання. Це допоможе забезпечити, що кожна вимога адекватно врахована та її виконання може бути простежене. На рисунку 1.4 наведена матриця трасування вимог цих взаємозв'язків:

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10
UC-1	+									
UC-2						+			+	
UC-3					+					
UC-4				+					+	
UC-5			+							
UC-6					+			+		
UC-7							+	+	+	+
UC-8		+							+	
UC-9										

Рисунок 1.4 – Матриця трасування вимог

1.4.2. Розроблення нефункціональних вимог

Для мобільного застосунку для боротьби з нікотиновою залежністю було висунуто наступні нефункціональні вимоги:

– операційна система: iOS версії 16.0 і вище та Android версії 12 і вище;

– мова інтерфейсу користувача: англійська;

– зручний інтерфейс користувача;

– застосунок має бути безкоштовним;

– можливість користування застосунком в офлайн режимі.

1.5. Постановка задачі

Мета цього дипломного проекту полягає у покращенні процесу дизайну продктів за допомогою розробки інтуїтивно зрозумілого мобільного зстосунку. Цей застосунок буде забезпечувати автоматичний підбір оптимальних кольорових схем для користувацьких інтерфейсів, що

сприятиме зниженню часу та зусиль, потрібних дизайнерам для цього процесу. Оцінка досягнення цієї мети буде базуватися на зворотному зв'язку користувачів, статистиці використання застосунку, а також на змінах в продуктивності роботи дизайнерів.

Для досягнення цієї мети, необхідно виконати наступні завдання:

– провести аналіз потреб користувачів, з'ясувавши їхні вимоги до дизайну кольорових схем;

– визначити параметри, на основі яких відбувається автоматичний підбір кольорових схем;

– розробити алоритм підбору кольорових схем, який враховує потреби користувачів і сучасні тенденції дизайну;

– проаналізувати різні платформи та технології для створення мобільного застосунку;

– розробити протоип мобільного застосунку і провести його тестування з участю майбутніх користувачів;

– на основі отриманого зворотного зв'язку вдосконалити застосунок і готовий продукт запустити в публічний доступ.

Результатом має стати високоякісний, відповідний вимогам користувачів мобільний застосунок для роботи з кольоровими схемами.

Висновки до розділу

В ході роботи над першим розділом дипломного проекту було здійснено детальний аналіз вимог до програмного забезпечення для мобільного додатку з підбору та аналізу кольорових схем.

В структурі даного розділу було визначено загальні положення та предметну область проекту. Була вивчена та проаналізована сфера дизайну, зокрема, вибір та використання кольорових схем в різноманітних сферах дизайну.

В результаті аналізу існуючих технологій та IT-проектів, було визначено основні тенденції, виклики та можливості, які необхідно врахувати під час створення нового продукту.

З урахуванням специфіки розробки мобільного додатку на React Native, було проаналізовано допоміжні програмні засоби, зокрема, XCode, Visual Studio Code та Expo Go, які забезпечують необхідну підтримку на всіх етапах розробки.

Оцінюючи відомі програмні продукти, було зроблено висновок про актуальність та перспективність розробки застосунку для кольорового аналізу, який би враховував потреби користувачів та надавав би більше можливостей для аналізу та вибору кольорових схем.

На основі усіх даних, було розроблено вимоги до програмного забезпечення. Функціональні вимоги були визначені з врахуванням потреб користувачів, тоді як нефункціональні вимоги стосувались забезпечення продуктивності, надійності та зручності використання додатку.

Таким чином, на основі проведеного аналізу, було сформульовано задачі для подальшого проектування та розробки мобільного додатку, що дозволить користувачам аналізувати, створювати та ділитися кольоровими схемами в зручному та інтуїтивно зрозумілому форматі.

2. МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Моделювання та аналіз програмного забезпечення

Для опису бізнес процесів була обрана BPMN модель.

Створення кольорової схеми користувачем (рисунок 2.1):

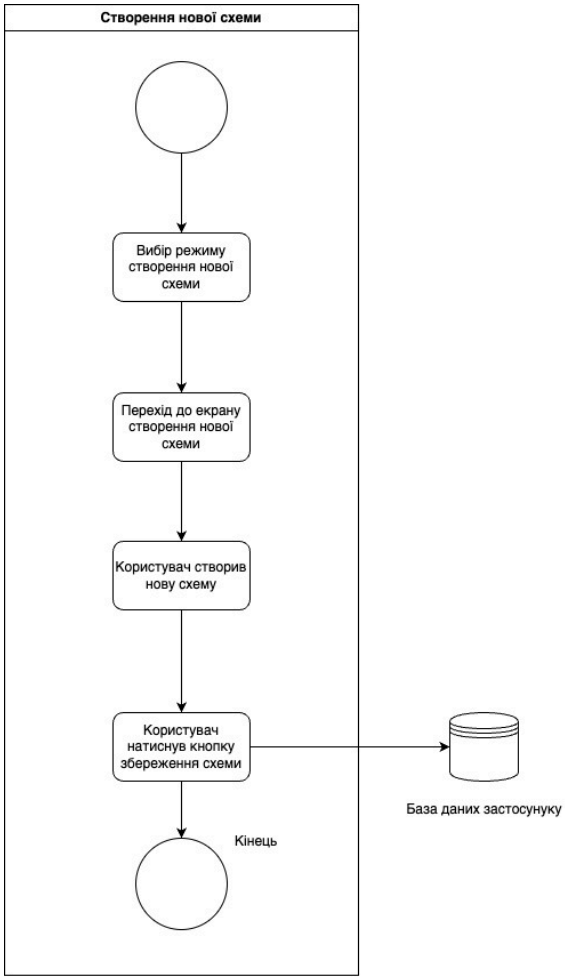


Рисунок 2.1 – Схема бізнес процесу створення кольорової схеми

Опис послідовності створення кольорової схеми користувачем:

– користувач вибирає режим створення нової схеми через User View;

– user ModelView переходить до ColorScheme View для створення;

нової схеми;

– користувач створює нову схему через ColorScheme View;

– користувач зберігає схему за допомогою ColorScheme ModelView;

– ColorScheme ModelView оновлює ColorScheme Model;

– Firebase [3] (як база даних застосунку) зберігає ці зміни;

– повернення до User View, де користувач може продовжувати використовувати застосунок або вийти.

За таким же принципом нижче наведені схеми редагування (рис. 2.2) та експортування кольорових схем (рис. 2.3):

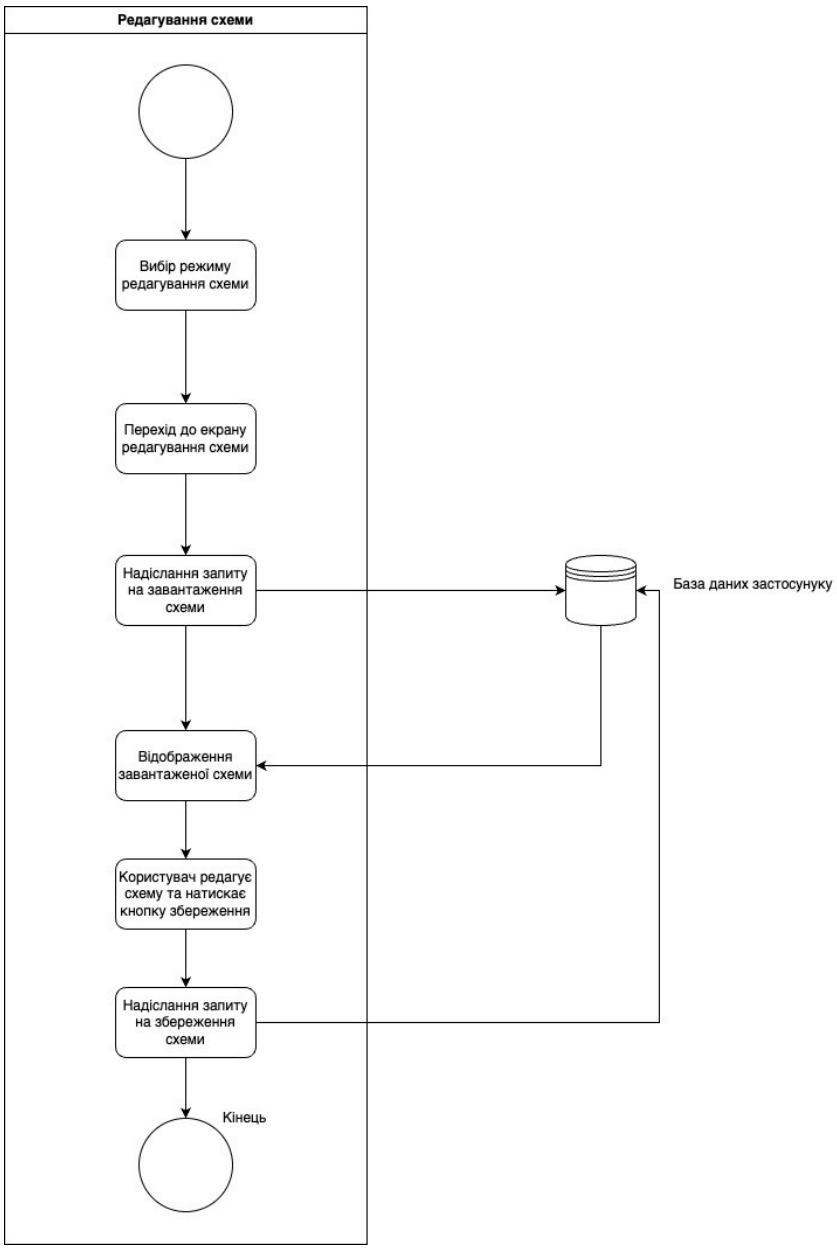


Рисунок 2.2 – Схема бізнес процесу редагування кольорової схеми

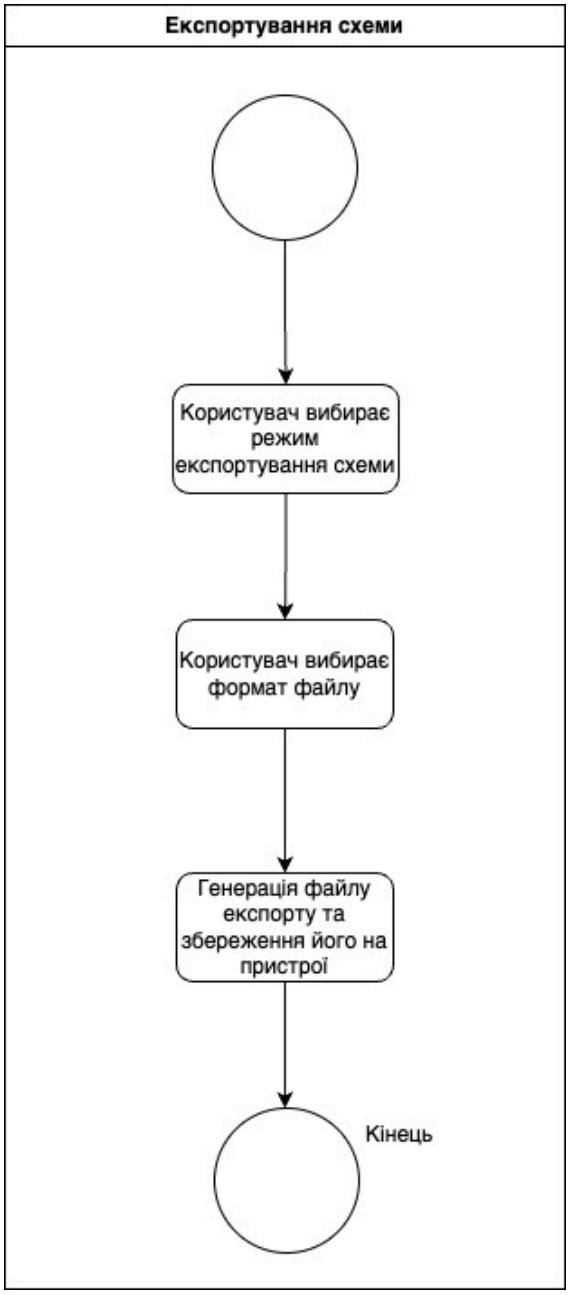


Рисунок 2.3 – Схема бізнес процесу експортування кольорової схеми

2.2. Архітектура програмного забезпечення

Мобільний застосунок було розроблено з використанням архітектури MVVM (Model-View-ViewModel), яка розділяє програму на три основні компоненти: модель, представлення та ViewModel.

Модель в архітектурі MVVM відповідає за керування даними та бізнес-логікою. Вона взаємодіє з базою даних або іншими джерелами даних для отримання та оновлення інформації. У цій архітектурі кожен модуль матиме

власну модель, яка обробляє маніпулювання даними та операції зберігання, характерні для цього модуля.

Представлення (View) відповідає за відображення користувача інтерфейсу користувача. Також View відображає інформацію, отриману з ViewModel, і взаємодіє з ViewModel для обробки даних, введених користувачем. Представлення інформує ViewModel про дії користувача або зміни, внесені до відображених даних.

ViewModel діє як посередник між моделлю та представленням. Вона отримує дані користувача від представлення та ініціює відповідні дії в моделі. Вона також отримує сповіщення від моделі про зміни в даних і відповідно оновлює представлення. ViewModel організовує потік даних і контролює поведінку програми.

У розробленій мобільній програмі кожен модуль міститиме ViewModel, яка виконує такі завдання, як отримання даних із моделі, оновлення представлення отриманими даними та обробка дій користувача. ViewModel відповідає за координацію взаємодії між моделлю та представленням.

Дотримуючись архітектури MVVM, мобільний додаток розділяє проблеми та сприяє модульному дизайну. Це забезпечує спрощене обслуговування, повторне використання коду та гнучкість у додаванні нових функцій або внесенні змін до існуючих функцій.

Архітектура застосунку продемонстрована на рисунку 2.4:

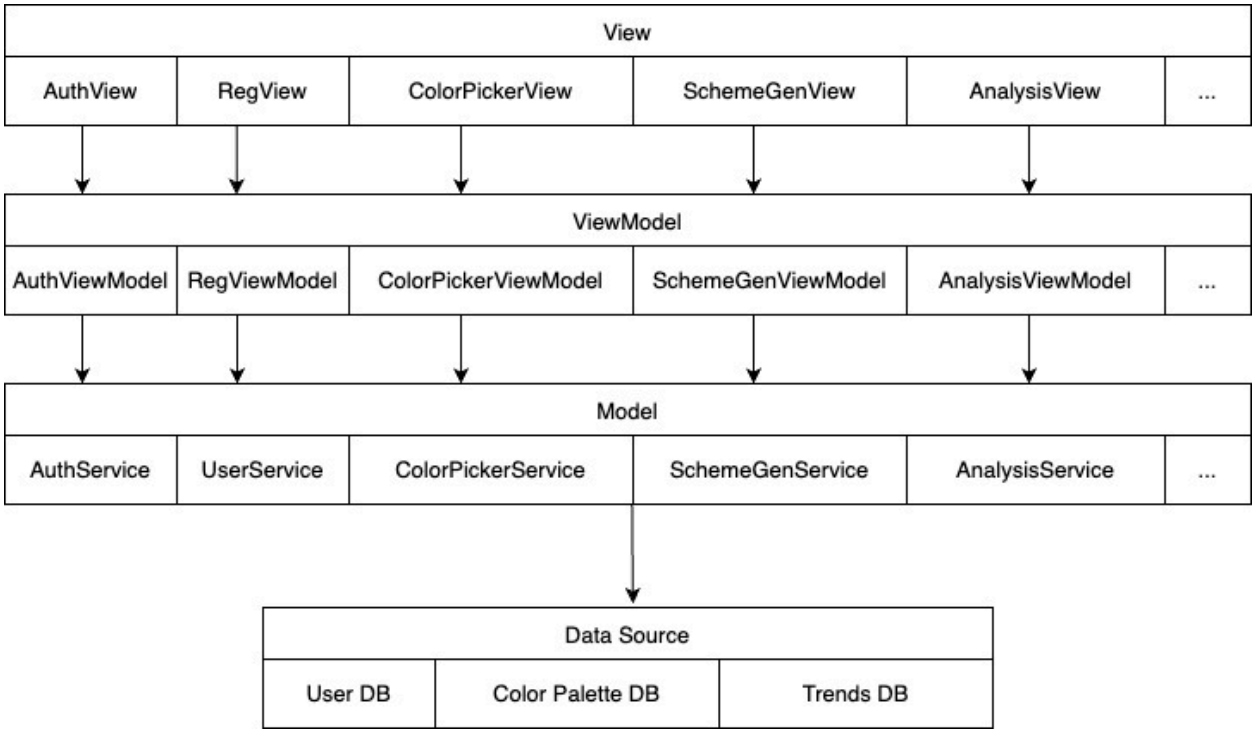


Рисунок 2.4 – Взаємодія між компонентами MVVM для різних частин застосунку

2.3. Конструювання програмного забезпечення

Клієнтську частину застосунку буде розділено на 8 окремих модулів:

- AuthentificationModule
- RegistrationModule
- ColorPickerModule
- ColorSchemeGeneratorModule
- ColorAnalysisModule
- ColorPaletteManagerModule
- ExportImportModule
- TrendsModule

Нижче наведено короткий опис кожного модуля, його призначення та короткий опис методів

AuthentificationModule надає користувачу доступ до автентифікації та отримання даних з бази даних.

Таблиця 2.1 – Опис методів модулю “AuthentificationModule”

login	Приймає ім'я користувача та пароль і повертає об'єкт користувача при успішній автентифікації
logout	Видаляє сеанс користувача
getUserData	повертає дані авторизованого користувача

RegistrationModule – це модуль який надає користувачу змогу зареєструватись в застосунку використовуючи електронну пошту та пароль.

Таблиця 2.2 – Опис методів модулю “RegistrationModule”

register	Приймає ім'я користувача, електронну пошту та пароль і створює новий аккаунт

ColorPickerModule – цей модуль дає змогу користувачу вибирати кольори для створення схем.

Таблиця 2.3 – Опис методів модулю “ColorPickerModule”

pickColor	Дає користувачу можливість вибрати колір

ColorSchemeGeneratorModule – цей модуль створює схеми кольорів на основі вибраних кольорів

Таблиця 2.4 – Опис методів модулю “ColorSchemeGeneratorModule”

generateScheme	Приймає основний колір і генерує схему кольорів

ColorAnalysisModule – цей модуль аналізує кольори та надає додаткову інформацію, таку як контрастність.

Таблиця 2.5 – Опис методів модулю “ColorAnalysisModule”

analyzeColor	Приймає колір та повертає аналіз цього кольору

ColorPaletteManagerModule – цей модуль дозволяє користувачу зберігати, завантажувати та управляти кольоровими схемами.

Таблиця 2.6 – Опис методів модулю “ColorPaletteManagerModule”

savePalette	Приймає назву палітри та кольорову схему для зберігання
loadPalette	Приймає назву палітри та повертає відповідну кольорову схему
deletePalette	Приймає назву палітри і видаляє її
getAllPalettes	Повертає всі збережені палітри користувача

ExportImportModule – цей модуль надає змогу імпортувати та експортувати кольорові схеми.

Таблиця 2.7 – Опис методів модулю “ExportImportModule”

exportPalette	Приймає назву палітри та експортує її в зазначений формат (наприклад, JSON або CSV)
importPalette	Приймає файл палітри та імпортує його в додаток

TrendsModule – цей модуль надає користувачу змогу переглянути та зберегти популярні палітри серед інших користувачів.

Таблиця 2.8 – Опис методів модулю “TrendsModule”

getTrends	Повертає популярні палітри, відсортовані по кількістю вподобань
likeTrendPalette	Відмітити палітру як вподобану
saveTrendPalette	Зберегти палітру до своєї колекції

2.3.1. Вибір інструментів для написання клієнтської частини застосунку

Для написання клієнтської частини мобільного застосунку було обрано React Native - мову програмування JavaScript, що використовує фреймворк React [4]. При цьому, для створення користувацького інтерфейсу застосунку в розробників є вибір з декількох бібліотек:

– React Native Material [6]: це стандартна бібліотека компонентів, яка входить до складу React Native. Вона включає в себе компоненти, такі як View, Text, Button тощо. Розробники можуть використовувати ці компоненти для створення користувацького інтерфейсу застосунку;

– React Native Paper [5]: це бібліотека UI компонентів, яка містить попередньо створені та налаштовані компоненти для швидкого створення інтерфейсу застосунку;

– React Navigation [7]: ця бібліотека дозволяє легко створювати навігацію між екранами в застосунку.

Розробники можуть використовувати ці бібліотеки окремо або разом, в залежності від вимог до інтерфейсу застосунку. Важливо зазначити, що React Native має імперативний стиль програмування, де розробник керує процесом виконання програми, визначаючи послідовність кроків.

В той же час, React Native також підтримує декларативний стиль програмування за допомогою React Hooks. React Hooks - це функції, що дозволяють розробникам використовувати стан та інші особливості React без написання класових компонентів.

Також для обробки асинхронних запитів в React Native застосовується бібліотека Redux або Context API вбудоване в React.

Redux: це бібліотека для управління станом застосунку. Вона дозволяє розробникам писати застосунки, що поводяться послідовно в різних середовищах та легкі в тестуванні. Redux підтримує асинхронну логіку, яка дозволяє розробникам визначати зв'язку між діями, які викликаються в компонентах, і змінами стану, що зберігаються в сховищі Redux.

Context API: це API, яке входить до складу React і дозволяє розробникам передавати дані безпосередньо між компонентами без потреби в пропусканні пропсів через проміжні компоненти. Це особливо корисно для даних, які потрібні багатьом компонентам на різних рівнях.

Крім того, для створення інтерфейсу користувача можна використовувати бібліотеку React Native Paper, яка надає набір предефінованих компонентів, стилізованих відповідно до принципів Material Design.

Згадані бібліотеки дають можливість створювати високоякісний користувацький інтерфейс та управляти динамічною поведінкою застосунку, використовуючи єдиний код для різних платформ.

2.3.2.Вибір інструментів для написання серверної частини застосунку

У якості серверної частини для даного застосунку, розробник обрав Firebase. Це рішення, що пропонує великий спектр служб, відповідно до вимог проекту:

– Firebase Firestore - це NoSQL база даних у реальному часі, яка дозволяє зберігати та синхронізувати дані між користувачами, відповідаючи потребам цього проекту;

– Firebase Storage пропонує масштабоване та безпечне місце для зберігання користувацького контенту, такого як зображення та відео;

– Firebase Functions дозволяє створювати і використовувати серверні функції для обробки подій в реальному часі;

– Firebase Hosting є відмінним рішенням для швидкого та надійного розгортання веб-додатків;

– Firebase Cloud Messaging пропонує можливість надсилати повідомлення та сповіщення користувачам.

Використовуючи Firebase, розробник отримує можливість зосередитися на клієнтській частині розробки, маючи зручний та потужний інструментарій для серверної частини. Така комплексна підтримка дозволяє ефективно оптимізувати процес розробки та забезпечує швидкість впровадження потрібних функцій та служб.

2.3.3. База даних

Firebase пропонує гнучкий та користувацький інтерфейс, який дозволяє легко зберігати та управляти даними, він ідеально підходить для роботи з React Native.

Ця платформа забезпечує миттєве оновлення даних у режимі реального часу завдяки її NoSQL структурі, що становить велику вагу для додатків, які вимагають негайних оновлень.

Firebase також надає масштабованість, що дозволяє без проблем адаптуватися до зростання додатку, та інтегровані рішення для автентифікації, резервного копіювання, аналітики та повідомлень.

Тому, використовуючи Firebase, можна зосередитися на оптимізації функціональності додатку, замість управління серверною інфраструктурою.

Зважаючи на ці ключові аспекти Firebase був обраний в якості бази даних для застосунку.

Схему бази даних зображено на рисунку 2.5:

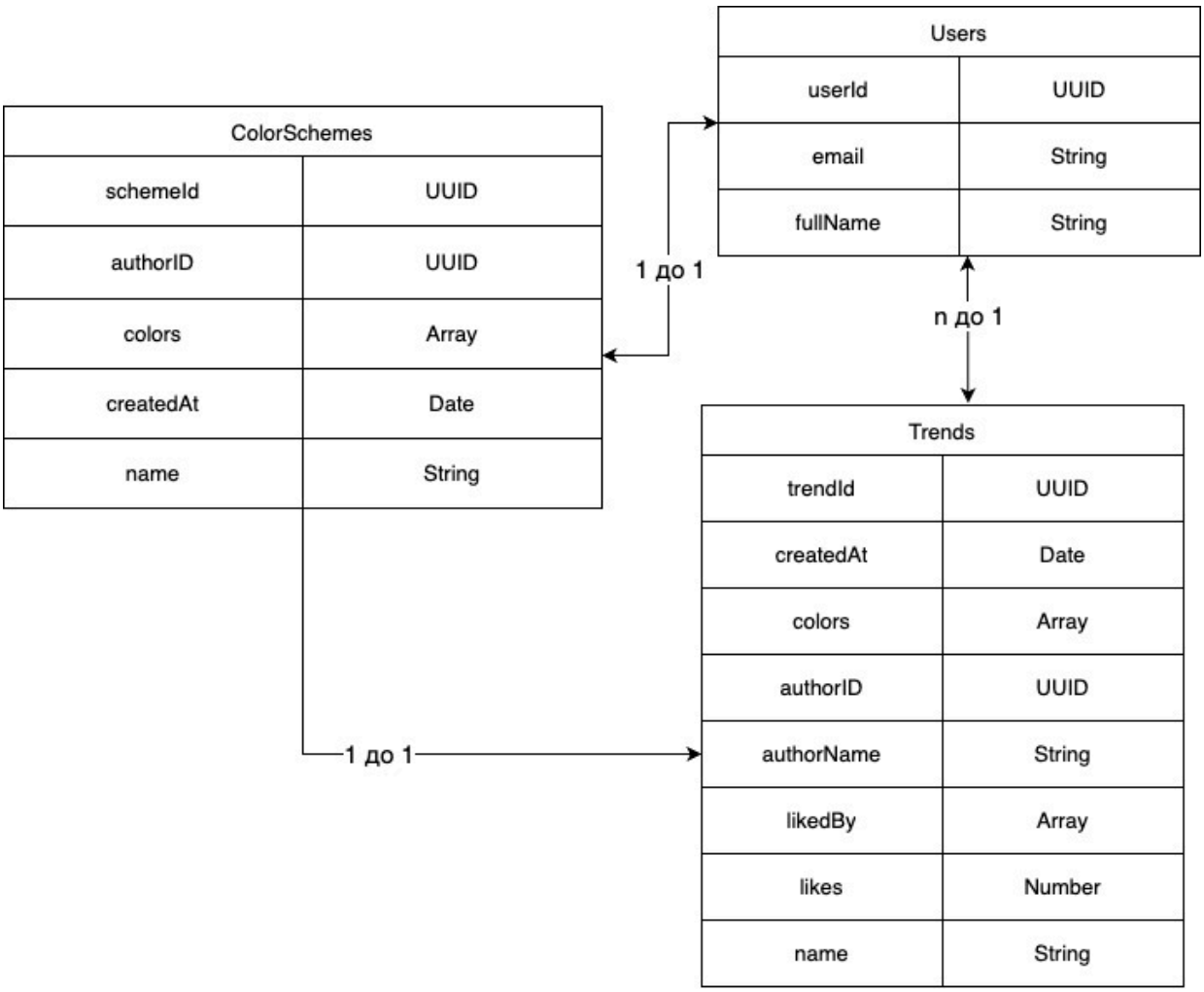


Рисунок 2.5 – Схема бази даних

Детальний опис користувацьких сутностей бази даних наведено у наступних таблицях:

Таблиця 2.9 – Опис таблиці “Users”

Назва таблиці	Користувачі (Users):
Призначення	Відповідає за відображення даних про користувача
Опис атрибутів таблиці	
userId	Відповідає за збереження унікального ідентифікатора користувача
email	Відповідає за збереження пошти користувача
fullName	Повне ім’я користувача

Таблиця 2.10 – Опис таблиці “ColorSchemes”

Назва таблиці	Кольорові схеми (ColorSchemes)
Призначення	Відповідає за опис та збереження кольорових схем
Опис атрибутів таблиці	
schemeId	Унікальний ідентифікатор кольорової схеми
authorId	Унікальний ідентифікатор користувача, що створив схему
colors	Масив кольорів, що входять до
createdAt	Дата створення схеми
name	Назва схеми

Таблиця 2.12 – Опис таблиці “Trends”

Назва таблиці	Тенденції (Trends)
Призначення	Відповідає за відображення та збереження трендів з кольорових схем
Опис атрибутів таблиці	
trendId	Унікальний ідентифікатор тенденції
createdAt	Дата створення тенденції
authorId	Унікальний ідентифікатор користувача, що створив тенденцію
colors	Масив кольорів, пов’язаних з цією тенденцією
authorName	Ім’я користувача, що створив тенденцію
likedBy	Масив унікальних ідентифікаторів користувачів, які вподобали відповідну тенденцією
likes	Кількість вподобань для відповідної тенденції
Name	Назва палітри, на основі якої була опублікована тенденція

2.4. Аналіз безпеки даних

Безпека даних є надзвичайно важливою для забезпечення захисту та конфіденційності даних користувачів.

Одним з ключових аспектів є використання HTTPS (Hypertext Transfer Protocol Secure) для зв’язку між застосунком та сервером. Завдяки використанню HTTPS усі передані дані шифруються, запобігаючи несанкціонованому доступу або перехопленню конфіденційної інформації під час передачі. Це шифрування гарантує, що дані залишаються конфіденційними та безпечними.

Крім того, застосунок не отримує файли cookie та не зберігає конфіденційні дані користувача. Такий підхід мінімізує ризик витоку даних і несанкціонованого доступу до особистої інформації.

Крім того, було впроваджено систему автентифікації firebase. Це дозволяє нам реалізацію багатофакторної, з можливістю збереження даних, автентифікації користувача в шифрованому вигляді.

Загалом, використання HTTPS для безпечного зв’язку, уникаючи зберігання конфіденційних даних користувача, дотримуючись методів безпечного кодування та впроваджуючи механізми автентифікації Firebase достатньо для того щоб захистити дані користувачів.

Висновки до розділу

У другому розділі дипломного проекту було зосереджено увагу на моделюванні та конструюванні програмного забезпечення для мобільного додатку з підбору та аналізу кольорових схем.

Розглянуто детальне моделювання та аналіз програмного забезпечення, у результаті якого були розроблені моделі бізнес-процесів, що відображають роботу програми з точки зору користувача.

В рамках дослідження архітектури програмного забезпечення, було вибрано модель MVVM для структуризації програми. Використання такої моделі забезпечує гнучкість, легкість обслуговування та масштабованість мобільних застосунків. Вона дозволяє ефективно відокремити логіку інтерфейсу користувача та бізнес-логіку, спрощуючи розробку, тестування та підтримку програмного забезпечення. Використання MVVM є важливим аспектом для успішної реалізації сучасних мобільних додатків.

В процесі конструювання програмного забезпечення, було вирішено поділити клієнтську частину на вісім модулів, кожен з яких відповідає за свій власний розділ функціональності, що дозволяє працювати над кожним з них незалежно та реалізовувати зміни без впливу на інші модулі.

Розглянуто вибір інструментів для написання клієнтської частини застосунку, було визначено, що React Native є оптимальним вибором для реалізації даного проєкту через його високу продуктивність, гнучкість та можливість використовувати однаковий код для різних платформ.

Була вибрана Firebase як база даних через її безпечність, швидкість та легкість інтеграції з React Native.

Аналіз безпеки даних виявив, що Firebase надає достатній рівень захисту інформації, проте додаткові заходи з безпеки повинні бути реалізовані на рівні додатку.

В цілому, у другому розділі було зроблено важливий крок на шляху до реалізації додатку, поклавши основу для його подальшого розроблення та імплементації.

3. АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Аналіз якості ПЗ

Аналіз якості програмного забезпечення є важливим етапом перед фіналізацією розробки. Для оцінки якості було використано декілька інструментів: Web Vitals і Lighthouse для оцінки веб-продуктивності.

– Web Vitals: Це метрики, які вимірюють користувацький досвід на веб-сторінках, такі як час завантаження, швидкість відгуку та стабільність. Використання Web Vitals дозволяє оцінити продуктивність застосунку та виявити можливі проблеми, які можуть впливати на користувацький досвід;

– Lighthouse: Цей інструмент надає детальну оцінку веб-сторінок з погляду продуктивності, доступності, найкращих практик та SEO. Використовуючи Lighthouse, можна отримати оцінку якості веб-додатку та рекомендації щодо покращення.

На рисунках 3.1 - 3.3 наведені звіти з цих інструментів:

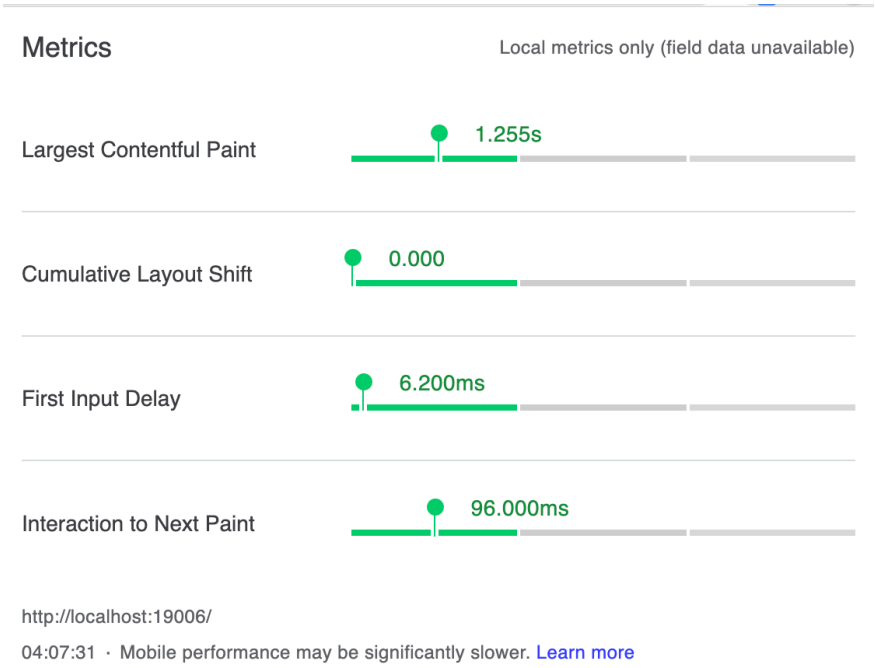


Рисунок 3.1 – Метрики Web Vitals

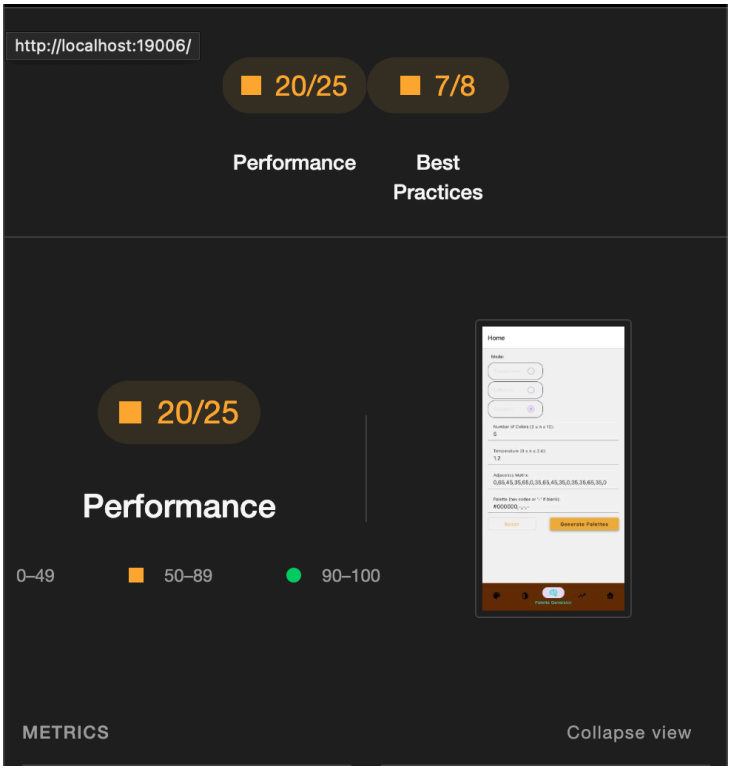


Рисунок 3.2 – Метрики Lighthouse через замірювання методом Timespan (проміжок часу)

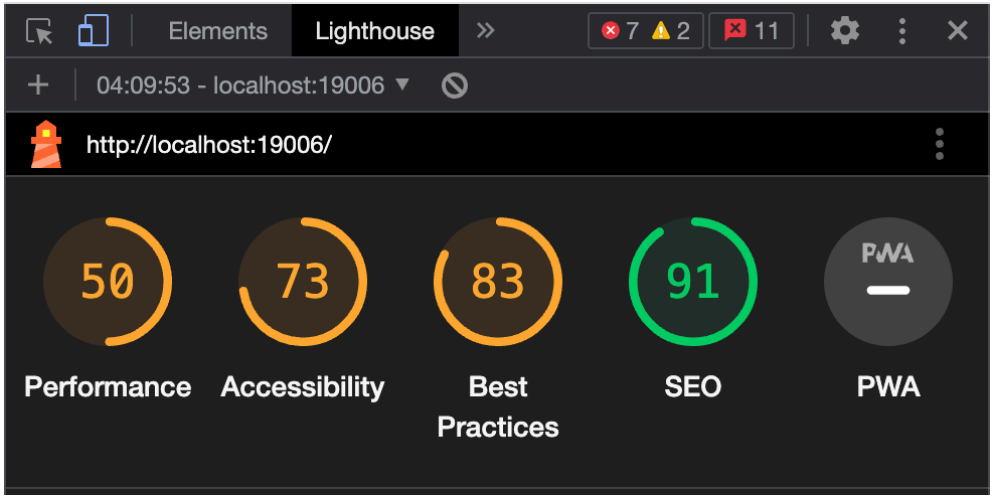


Рисунок 3.3 – Метрики Lighthouse через замірювання методом Navigation (навігація)

Загальний аналіз Web Vitals та Lighthouse показує, що програмне забезпечення має добрі показники продуктивності та якості. Згідно з Web Vitals, час завантаження першого зображення (LCP) становить 1.255 секунди, що є в прийнятних межах. Показники кумулятивної стабільності макету (CLS) та часу відгуку першої взаємодії (FID) також показують добрі результати зі значеннями 0 та 6.2 мс відповідно. Час бездіяльності мережі (ItNP) складає 96 мс, що також є прийнятним показником.

Оцінка Lighthouse показує добрі результати в різних аспектах. З погляду продуктивності (Perfomance), застосунок отримав 20 балів з 25 можливих. Щодо найкращих практик (Best practices), було отримано 7 балів з 8. Оцінки в навігаційних категоріях також є задовільними, з перфомансом на рівні 50, доступністю - 73, найкращими практиками - 83 та SEO - 91.

На підставі цих результатів можна зробити висновок, що програмне забезпечення має хорошу якість та продуктивність. Проте, є деякі можливості для покращення, зокрема в аспекті найкращих практик та доступності. Рекомендації Lighthouse можуть бути використані для вдосконалення продукту та підвищення якості користувацького досвіду.

3.2. Опис процесів тестування

В даній роботі було виконане мануальне тестування функціональної частини програмного забезпечення. Опис відповідних тестів наведено у таблицях 3.1 – 3.13.

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації

Продовження таблиці 3.1

Вхідні данні	Повне ім’я, електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	Вводяться ім'я, коректна електронна пошта (не зареєстрована раніше) і пароль (6 символів). Натискається кнопка реєстрації.
Очікуваний результат	Реєстрація успішна, користувач додається до системи та переходить на сторінку збережених палітр.
Фактичний результат	Реєстрація успішна, користувач додається до системи та переходить на сторінку збережених палітр.

Таблиця 3.2 – Тест 1.2

Тест	Відображення помилки реєстрації користувача
Модуль	Реєстрація користувача
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні данні	Ім’я, електронна пошта, пароль, підтвердження паролю
Опис проведення	Вводяться ім’я, некоректна електронна пошта, пароль від 6 символів. Натискається кнопка реєстрації.
Очікуваний результат	Реєстрація невдала, користувач не додається у систему і відображається повідомлення про використання неправильної адреси.
Фактичний результат	Реєстрація невдала, користувач не додається у систему і відображається повідомлення про використання неправильної адреси.

Таблиця 3.3 – Тест 1.3

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні данні	Електронна пошта, пароль
Опис проведення	Вводиться правильна електронна пошта та пароль, який містить не менше 6 символів.
Очікуваний результат	Користувач успішно авторизується і переходить до головного екрану застосунку.
Фактичний результат	Користувач успішно авторизується і переходить до головного екрану застосунку.

Таблиця 3.4 – Тест 1.4

Тест	Введення неправильних даних для авторизації
Модуль	Авторизація користувача
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні данні	неправильна електронна пошта або пароль
Опис проведення	Вводиться некоректна електронна пошта або пароль.
Очікуваний результат	Авторизація користувача не вдалася, і користувач отримує повідомлення з вказанням причини відмови.
Фактичний результат	Авторизація користувача не вдалася, і користувач отримує повідомлення з вказанням причини відмови.

Таблиця 3.5 – Тест 1.5

Тест	Відображення списку палітр
Модуль	Кольорові палітри
Номер тесту	1.5
Початковий стан системи	Користувач знаходиться на головному екрані застосунку
Вхідні данні	-
Опис проведення	Перевірка, чи відображаються усі доступні палітри користувача на головному екрані.
Очікуваний результат	Відображення списку усіх доступних палітр користувача на головному екрані.
Фактичний результат	Відображення списку усіх доступних палітр користувача на головному екрані.

Таблиця 3.6 – Тест 1.6

Тест	Створення нової палітри
Модуль	Вибір кольору
Номер тесту	1.6
Початковий стан системи	Користувач знаходиться на екрані вибору кольору
Вхідні данні	Назва палітри, список кольорів
Опис проведення	Користувач вводить назву палітри та обирає кольори для палітри.
Очікуваний результат	Палітра успішно створена та відображається у списку палітр користувача.
Фактичний результат	Палітра успішно створена та відображається у списку палітр користувача.

Таблиця 3.7 – Тест 1.7

Тест	Видалення палітри
Модуль	Кольорові палітри
Номер тесту	1.7
Початковий стан системи	Користувач знаходиться на головному екрані застосунку зі списком палітр
Вхідні данні	-
Опис проведення	Користувач вибирає палітру для видалення.
Очікуваний результат	Палітра успішно видаляється і більше не відображається у списку палітр користувача.
Фактичний результат	Палітра успішно видаляється і більше не відображається у списку палітр користувача.

Таблиця 3.8 – Тест 1.8

Тест	Редагування палітри
Модуль	Вибір кольору
Номер тесту	1.8
Початковий стан системи	Користувач знаходиться на екрані вибору кольору
Вхідні данні	Назва палітри, список кольорів
Опис проведення	Користувач вибирає палітру для редагування та змінює її назву та/або список кольорів.
Очікуваний результат	Палітра успішно редагується та відображається зі зміненими даними у списку палітр користувача.
Фактичний результат	Палітра успішно редагується та відображається зі зміненими даними у списку палітр користувача.

Таблиця 3.9 – Тест 1.9

Тест	Пошук палітри
Модуль	Кольорові палітри
Номер тесту	1.9
Початковий стан системи	Користувач знаходиться на головному екрані застосунку зі списком палітр
Вхідні данні	Ключове слово для пошуку палітри
Опис проведення	У поле вводиться ключове слово, яке буде містити назва палітри.
Очікуваний результат	Відображаються результати пошуку, що відповідають ключовому слову.
Фактичний результат	Відображаються результати пошуку, що відповідають ключовому слову.

Таблиця 3.10 – Тест 1.10

Тест	Правильне заповнення форми генерації палітр
Модуль	Генерація кольорових палітр
Номер тесту	1.10
Початковий стан системи	Користувач знаходиться на екрані генерації палітр
Вхідні данні	Кількість кольорів, тип палітри, матриця суміжності, попередньо відомі кольори
Опис проведення	У відповідні поля вводяться правильні дані для генерації палітри та натискається кнопка генерації.
Очікуваний результат	Палітра генерується з вказаною кількістю кольорів та відповідає вибраному типу.
Фактичний результат	Палітра генерується з вказаною кількістю кольорів та відповідає вибраному типу.

Таблиця 3.11 – Тест 1.11

Тест	Коректний колір для аналізу
Модуль	Аналіз кольорів
Номер тесту	1.11
Початковий стан системи	Користувач знаходиться на екрані аналізу кольорів
Вхідні данні	Коректний колір для аналізу у форматі hex
Опис проведення	У поле вводиться коректний колір для аналізу у форматі hex та натискається кнопка аналізу.
Очікуваний результат	Відображаються дані про аналіз коректного кольору.
Фактичний результат	Відображаються дані про аналіз коректного кольору.

Таблиця 3.12 – Тест 1.12

Тест	Неправильне заповнення форми генерації палітр
Модуль	Генерація кольорових палітр
Номер тесту	1.12
Початковий стан системи	Користувач знаходиться на екрані генерації палітр
Вхідні данні	Некоректні дані для генерації палітри
Опис проведення	У відповідні поля вводяться некоректні дані для генерації палітри та натискається кнопка генерації.
Очікуваний результат	Відображається повідомлення про неправильне заповнення форми.
Фактичний результат	Відображається повідомлення про неправильне заповнення форми.

Таблиця 3.13 – Тест 1.13

Тест	Некоректний колір для аналізу
Модуль	Аналіз кольорів
Номер тесту	1.13
Початковий стан системи	Користувач знаходиться на екрані аналізу кольорів
Вхідні данні	Некоректний колір для аналізу
Опис проведення	У поле вводиться некоректний колір для аналізу та натискається кнопка аналізу.
Очікуваний результат	Відображається повідомлення про некоректний колір.
Фактичний результат	Відображається повідомлення про некоректний колір.

3.3. Опис контрольного прикладу

Далі ми проведемо контрольний приклад, в якому описані кроки виконання тестування та наведені ілюстрації до кожного з кроків. Починаємо з запуску застосунку, після чого ми потрапляємо на екран авторизації. На рисунку 3.4 представлений зовнішній вигляд екрану авторизації.

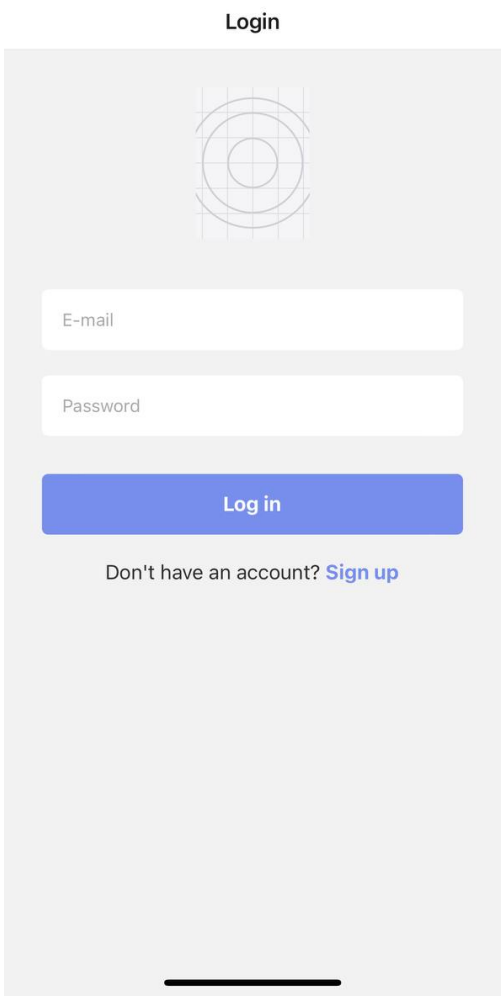


Рис. 3.4 – Зовнішній вигляд екрану авторизації

Далі, для тестування, введемо неправильні дані в поле електронної пошти та натиснемо кнопку авторизації. Згідно з рисунком 3.5, застосунок відображає повідомлення про невірний пароль користувача:

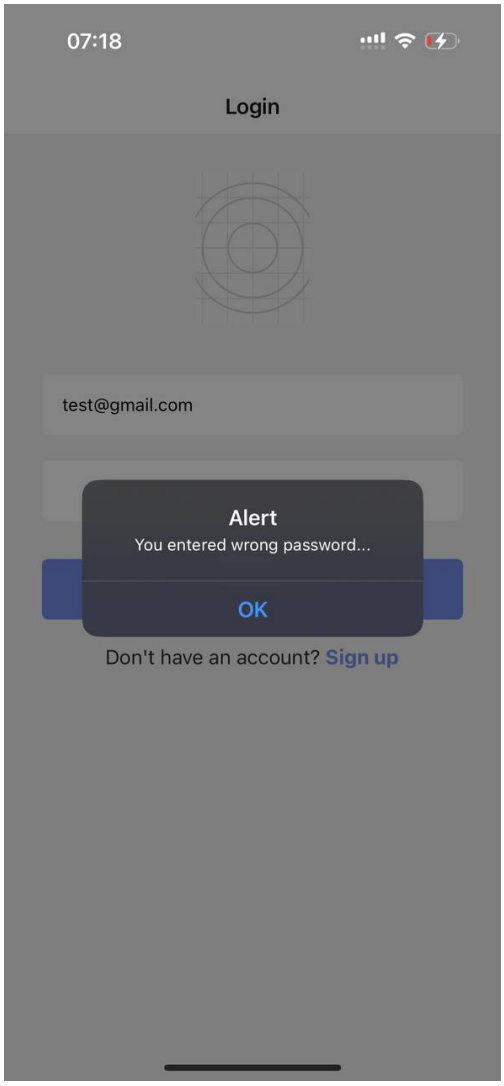


Рис. 3.5 – Повідомлення про те, що пароль був введений невірно

Потім спробуємо ввести неправильну електронну пошту. Згідно з рисунком 3.6, застосунок робить кнопку авторизації неактивною, якщо користувач ввів неправильну електронну пошту:

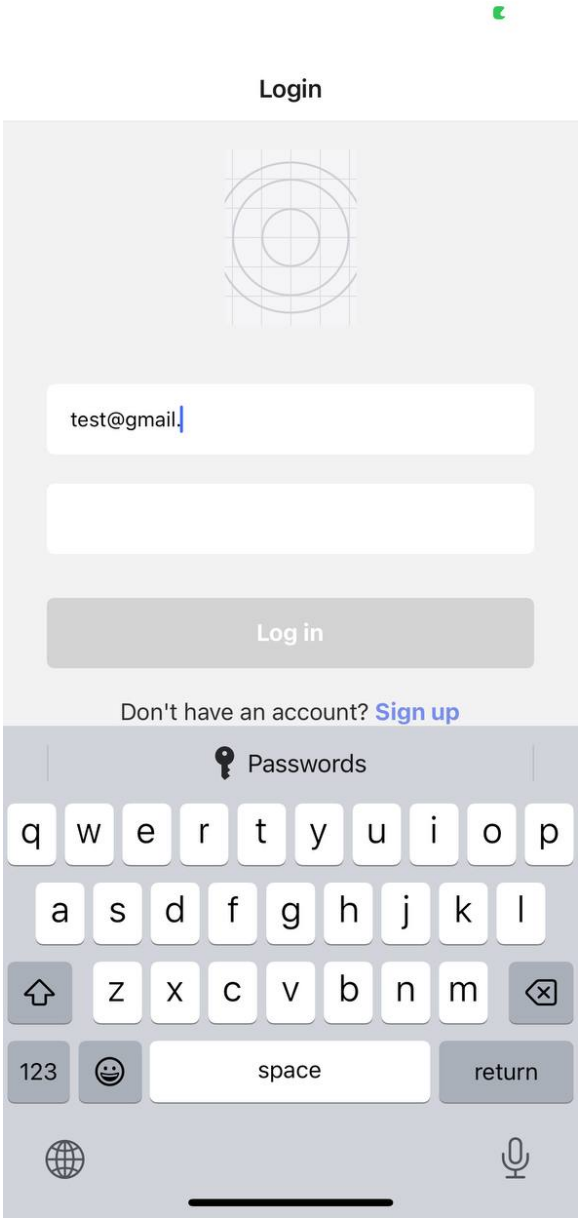


Рис. 3.6 – Кнопка авторизації стає сірою, якщо пошта введена невірно

Тепер перейдемо до екрану реєстрації. Користувач повинен ввести наступні дані: електронну пошту, пароль та підтвердження паролю. Якщо користувач неправильно введе пароль другий раз, застосунок покаже повідомлення про неправильне підтвердження паролю та відмовить у реєстрації., як показано на рисунку 3.7:

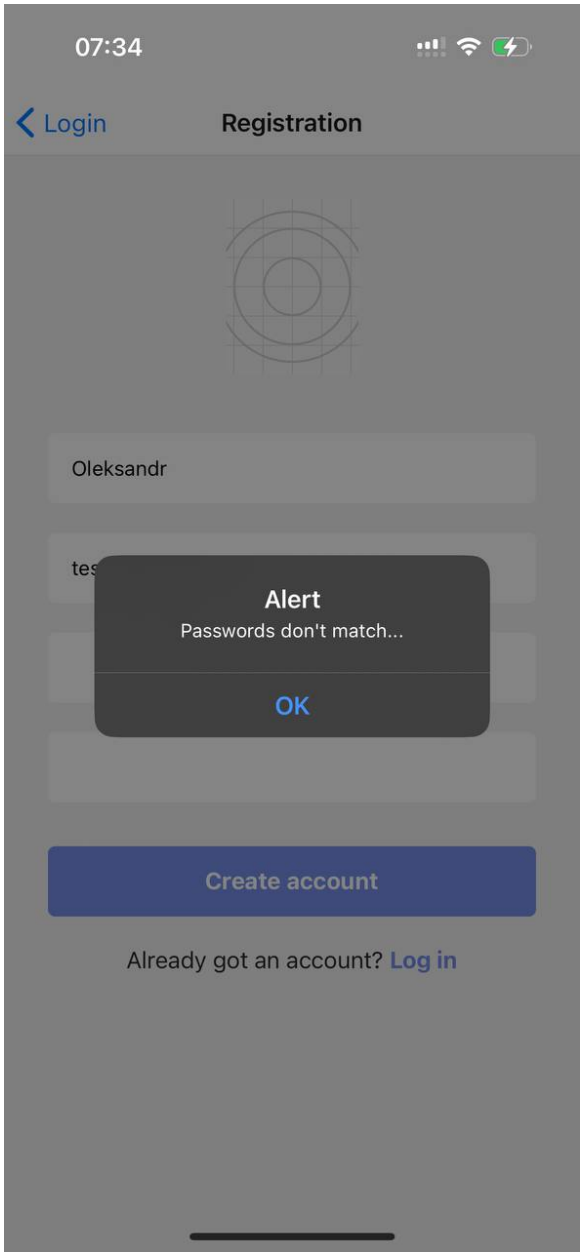


Рис. 3.7 – Повідомлення про те, що пароль був введений невірно

Роздивимось ще два випадки: неправильно введений колір та неправильно заповнена форма.

– випадок неправильно введеного кольору: після вибору кольору для аналізу, користувач вводить неправильний колір. Застосунок відображає повідомлення про неправильний колір та не проводить аналіз (рис. 3.8);

– випадок невірно заповненої форми: Користувач вводить неправильні дані у форму генерації кольорових палітр (наприклад, неправильний тип

схеми, неправильні налаштування тощо). Застосунок відображає повідомлення про неправильно заповнену форму та не генерує палітру (рис. 3.9).

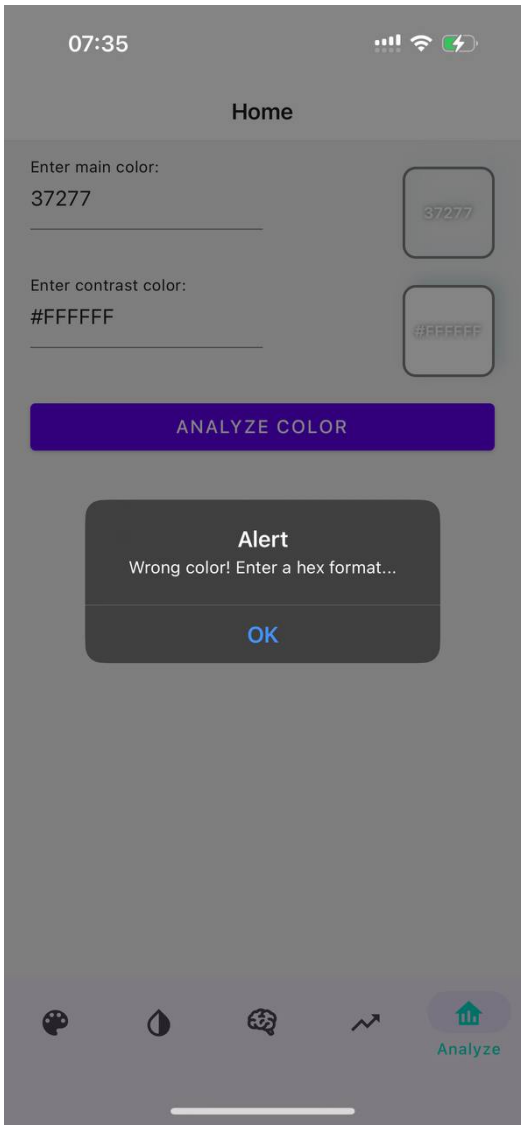


Рис. 3.8 – Повідомлення про те, що колір був введений в невірному форматі

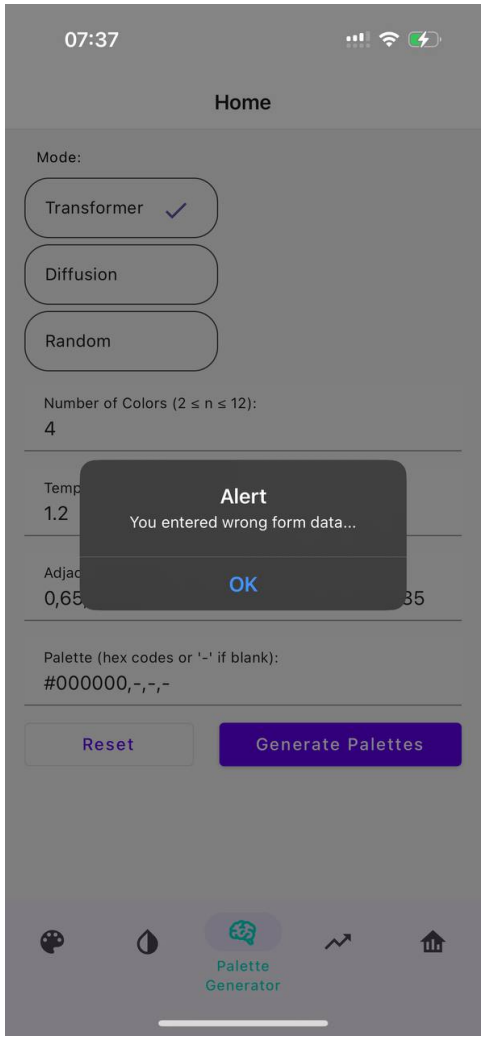


Рис. 3.9 – Повідомлення про те, що в форму генерації палітр були введені невірні дані

Розглянемо також процес зміни кольорової палітри інтерфейсу. Користувачу запропоновано обрати одну зі збережених кольорових схем, які можуть визначити загальний вигляд інтерфейсу. На рисунку 3.10 зображений інтерфейс за замовчуванням, а на рисунку 3.11 - після того як користувач застосував до нього першу із збережених палітр:

Рис. 3.10 – Інтерфейс за замовчуванням

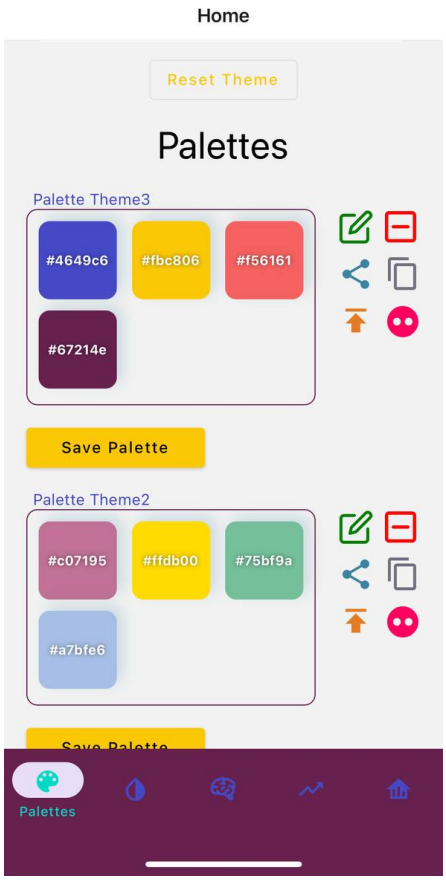


Рис. 3.11 – Інтерфейс після застосування кольорової палітри

Висновки до розділу

В ході дослідження теми "Мобільний застосунок з підбору кольорових схем для користувацьких інтерфейсів" було проведено аналіз якості програмного забезпечення та описано процеси тестування.

Аналіз якості ПЗ здійснювався за допомогою інструментів Web Vitals та Lighthouse. Використання Web Vitals дозволило оцінити веб-продуктивність з погляду швидкості завантаження сторінок, стабільності та відгуку на дії користувача. Lighthouse надавав звіти з питань продуктивності, доступності, дотримання найкращих практик та оптимізації для пошукових систем.

Загальний результат аналізу свідчив про хорошу продуктивність та відповідність найкращим практикам.

Процеси тестування включали мануальне тестування функціональної частини програмного забезпечення. Було проведено тести на реєстрацію та авторизацію користувача, валідацію даних, пошук палітри, форму генерації палітри та введення кольору для аналізу. Кожен тест описувався початковим станом системи, вхідними даними, описом проведення тесту та очікуваним та фактичним результатами.

Загалом, на основі проведеного аналізу якості ПЗ та процесів тестування можна зробити висновок про успішну реалізацію мобільного застосунку з підбору кольорових схем для користувацьких інтерфейсів. Результати аналізу свідчать про задовільну продуктивність та відповідність найкращим практикам, а проведені тести підтверджують коректну роботу функціональних модулів застосунку. Таким чином, розроблений застосунок може бути використаний для зручного та ефективного підбору кольорових схем у користувацьких інтерфейсах.

4. ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Розгортання програмного забезпечення

Розгортання програмного забезпечення (ПЗ) на React Native та Expo включає кілька кроків. Процес може бути розбитий на підготовчу фазу, збірку проекту, тестування, і наостанок - розгортання.

Підготовка до розгортання:

– необхідне програмне забезпечення (Node.js, npm/yarn, Expo CLI) має бути встановлене. Якщо ці програми відсутні, їх можна встановити, перейшовши за відповідними посиланнями;

– репозиторій клонується за допомогою git clone;

– залежності проекту встановлюються за допомогою npm install або yarn.

Збірка проекту:

– для збірки проекту використовується команда expo build:android для Android-додатка або expo build:ios для iOS-додатка;

– Expo запитує про режим збірки: archive для генерації .ipa (iOS) або .apk (Android) файлів для розповсюдження або simulator для генерації файлів, які можна використовувати на емуляторі.

Тестування:

– для тестування додатка на емуляторі використовується expo start;

– також можна провести ручні тести або автоматичні тести, наприклад, з використанням Jest.

Розгортання:

– для розгортання додатка, .apk або .ipa файл завантажується до магазину додатків (Google Play Store для Android, App Store для iOS);

– весь необхідний мета-матеріал (опис, скріншоти тощо) подається, а також проходить процес рецензування.

Ці кроки забезпечують повний процес розгортання ПЗ для React Native та Expo. Зазначимо, що для розгортання додатків у магазинах додатків, можливо, доведеться оплатити певні збори та/або пройти процес перевірки.

4.2. Підтримка програмного забезпечення

Для підтримки застосунка, створеного на React Native та Expo, важливо розглянути специфічні аспекти, пов'язані з цими технологіями.

Виправлення помилок та операційна підтримка:

– для виявлення та виправлення помилок можна використовувати інструменти відлагодження та моніторингу, такі як Flipper або Reactotron;

– спілкування з користувачами може відбуватися через різні канали, включаючи службу підтримки в додатку, email, соціальні мережі або телефонну підтримку.

Технічна підтримка та оновлення безпеки:

– регулярно слід оновлювати Expo SDK та інші бібліотеки, що використовуються в проекті, для забезпечення актуальності застосунку та його безпеки;

– для виявлення та виправлення можливих проблем з продуктивністю можна використовувати профілювальники React Native.

Стратегічна підтримка та додавання нових функцій:

– основуючись на відгуках користувачів та аналізі даних про використання додатку, можна розробляти та реалізовувати нові функції;

– Expo регулярно випускає нові можливості, що можуть бути інтегровані в застосунок.

Регулярні оновлення:

– для надання користувачам останніх оновлень та виправлень помилок, регулярно публікуються нові версії додатка в магазинах додатків;

– з Expo Updates можна розгорнути оновлення безпосередньо на пристрої користувачів, минувши процес рецензування магазином додатків.

Цей підхід до підтримки застосунка на React Native та Expo забезпечує регулярні оновлення, виправлення помилок та покращення для користувачів.

Висновки до розділу

Розгортання та підтримка програмного забезпечення - два критично важливі аспекти в життєвому циклі будь-якого застосунка. Вони впливають на успіх продукту, задоволеність користувачів і, врешті решт, на репутацію розробників або компанії.

Розгортання програмного забезпечення, створеного за допомогою React Native та Expo, потребує певного розуміння екосистеми цих інструментів. Це включає встановлення необхідного середовища, збірку проекту, тестування та кінцеве розгортання додатка у відповідних магазинах додатків. Під час цього процесу, важливо враховувати вимоги безпеки, зокрема, регулярно оновлювати залежності та проходити процеси рецензування, вимогані магазинами додатків.

Підтримка застосунка, створеного на React Native та Expo, також вимагає особливої уваги. Вона включає виправлення помилок, використовуючи відповідні інструменти відлагодження, надання підтримки користувачам, регулярне оновлення безпеки та інтеграцію нових можливостей, що надаються екосистемою Expo. Регулярні оновлення застосунку, що надаються користувачам, важливі для забезпечення високої якості та актуальності продукту.

У цілому, процес розгортання та підтримки програмного забезпечення на React Native та Expo є важливим і складним завданням, яке вимагає чіткого планування та виконання. Проте, враховуючи всі вимоги та найкращі практики, розробники можуть забезпечити стабільний, високоякісний та безпечний застосунок, що задовольнятиме потреби користувачів.

ВИСНОВКИ

В результаті виконання дипломного проєкту було спроєктовано мобільний додаток з підбору та аналізу кольорових схем на платформі React Native. Для розробки використовувались комбінація різних програмних засобів, зокрема XCode та Visual Studio Code, а також Firebase в якості бази даних.

Найважливіші наукові й практичні результати роботи включають:

– розроблення алгоритму підбору кольорових схем, враховуючи потреби користувачів і сучасні тенденції дизайну;

– успішне створення прототипу мобільного застосунку, його тестування та вдосконалення на основі зворотного зв'язку користувачів;

– відповідність отриманих результатів сучасному рівню наукових і технічних знань.

Ступінь впровадження та можливі галузі або сфери використання результатів роботи:

– розроблений мобільний застосунок може бути використаний у сфері дизайну продуктів та інтерфейсів;

– впровадження застосунку сприятиме зниженню часу та зусиль, необхідних для підбору кольорових схем, та покращить продуктивність роботи дизайнерів.

Наукова, науково-технічна, соціально-економічна значущість роботи:

– розроблений алгоритм підбору кольорових схем на основі аналізу потреб користувачів та сучасних тенденцій дизайну має наукову значущість у вдосконаленні процесу дизайну продуктів;

– застосування мобільного застосунку може мати соціально-економічний вплив, забезпечуючи більш ефективну роботу дизайнерів та скорочуючи час, необхідний для розробки продуктів.

Модифіковано:

– було модифіковано алгоритм підбору кольорових схем з метою забезпечення кращої точності та збалансованості відповідно до вимог користувачів та сучасних тенденцій дизайну;

– було вдосконалено параметри підбору кольорових схем, враховуючи додаткові фактори, такі як контрастність, співвідношення кольорів та їх відповідність різним типам інтерфейсів.

Набуло подальший розвиток:

– було розширено можливості застосунку шляхом інтеграції з онлайн-сервісами та платформами для спільної роботи та обміну кольоровими схемами з іншими дизайнерами;

– було покращено аналіз зворотного зв'язку в застосунку, що дозволяє збирати більш детальну інформацію про використання та задоволення користувачів, що в подальшому сприятиме поліпшенню продуктивності та якості дизайну.

Стан вирішення усіх поставлених в дипломному проєктуванні задач:

– усі поставлені в дипломному проєктуванні задачі були успішно вирішені. Було проведено аналіз потреб користувачів, розроблено алгоритм підбору кольорових схем, створено прототип мобільного застосунку, проведено його тестування з участю користувачів та вдосконалено на основі отриманого зворотного зв’язку;

– результати тестування та аналізу ефективності підтвердили успішну реалізацію мобільного застосунку з підбору кольорових схем.

Отже, дипломна робота має значний науковий, науково-технічний та соціально-економічний вплив, а результати її виконання можуть бути успішно впроваджені у сфері дизайну продуктів та інтерфейсів. Продовження досліджень у даній тематиці є доцільним з метою подальшого вдосконалення алгоритмів підбору кольорових схем, розширення функціональності застосунку та його адаптації для інших платформ та галузей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Getting Started with React Native [Електронний ресурс]. - Режим доступу: https://reactnative.dev/docs/getting-started. - Дата звернення: 01.06.2023.

2) Getting Started with Expo Go [Електронний ресурс]. - Режим доступу: https://docs.expo.dev/get-started/expo-go/. - Дата звернення: 01.06.2023.

3) Official Firebase Documentation [Електронний ресурс]. - Режим доступу: https://firebase.google.com/docs. - Дата звернення: 01.06.2023.

4) React Reference [Електронний ресурс]. - Режим доступу: https://react.dev/reference/react. - Дата звернення: 01.06.2023.

5) Getting Started with React Native Paper [Електронний ресурс]. - Режим доступу: https://callstack.github.io/react-native-paper/docs/guides/getting-started. - Дата звернення: 01.06.2023.

6) Getting Started with React Native Material [Електронний ресурс]. - Режим доступу: https://www.react-native-material.com/docs/getting-started. - Дата звернення: 01.06.2023.

7) Getting Started with React Navigation [Електронний ресурс]. - Режим доступу: https://reactnavigation.org/docs/getting-started. - Дата звернення: 01.06.2023.

8) Tidwell, J. "Designing Interfaces: Patterns for Effective Interaction Design". // Interface Design. - 2005. - Vol. 1. - P. 85-100.

9) Anderson, S. "Seductive Interaction Design: Creating Playful, Fun, and Effective User Experiences". // User Experience Design. - 2011. - Vol. 1. - P. 50-65.

ДОДАТКИ

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015394125

Дата перевірки:
02.06.2023 14:27:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 14:53:31 EEST

ID користувача:
76913

Назва документа: IT-92_Рябко_ПЗ

Кількість сторінок: 69 **Кількість слів:** 8105 **Кількість символів:** 62401 **Розмір файлу:** 2.38 MB **ID файлу:** 1015058518

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.1%
Схожість

Найбільша схожість: 2.37% з джерелом з Бібліотеки (ID файлу: 1015058486)

2.37% Джерела з Інтернету	113	Сторінка 71
6.02% Джерела з Бібліотеки	150	Сторінка 72

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	637
Підозріле форматування	11 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК З ПІДБОРУ КОЛЬОРОВИХ СХЕМ ДЛЯ ІНТЕРФЕЙСІВ

Текст програми

КПІ.ІТ-9222.045440.03.13

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр РЯБКО

Київ – 2023

Файл app.json

```
{
  "expo": {
    "name": "diploma-pallete-app",
    "slug": "diploma-plallete-app",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "userInterfaceStyle": "light",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#ffffff"
    },
    "assetBundlePatterns": [
      "**/*"
    ],
    "ios": {
      "supportsTablet": true
    },
    "android": {
      "adaptiveIcon": {
        "foregroundImage": "./assets/adaptive-icon.png",
        "backgroundColor": "#ffffff"
      }
    },
    "web": {
      "favicon": "./assets/favicon.png"
    },
    "extra": {
      "eas": {
        "projectId": "37b4c275-3f92-49ed-850a-930372d35ba3"
      }
    },
    "owner": "olimpki"
  }
```

Файл App.js

```
import 'react-native-gesture-handler';
import React, { useEffect, useState } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { decode, encode } from 'base-64';
import { ActivityIndicator, Flex } from '@react-native-material/core';
import { ClickOutsideProvider } from 'react-native-click-outside';
import { Provider as PaperProvider } from 'react-native-paper';
import { LoginScreen, HomeScreen, RegistrationScreen } from './src/screens';
import {
  auth,
  db,
  doc,
  getDoc,
  onAuthStateChanged,
} from './src/firebase/config';
import { ThemeProvider } from './src/contexts/themeContext';

if (!global.btoa) {
  global.btoa = encode;
}
if (!global.atob) {
  global.atob = decode;
}
```

```
const Stack = createStackNavigator();

export default function App() {
  const [loading, setLoading] = useState(true);
  const [user, setUser] = useState(null);

  useEffect(() => {
    onAuthStateChanged(auth, (user) => {
      if (user) {
        const userRef = doc(db, 'users', user.uid);
        getDoc(userRef)
          .then((document) => {
            const userData = document.data();
            setLoading(false);
            setUser(userData);
          })
          .catch((error) => {
            alert(error);
            setLoading(false);
          });
      } else {
        setLoading(false);
      }
    });
  }, []);

  if (loading) {
    return (
      <Flex justify='center' items='center' fill>
        <ActivityIndicator size='large' />
      </Flex>
    );
  }

  return (
    <ClickOutsideProvider>
      <ThemeProvider>
        <PaperProvider>
          <NavigationContainer>
            <Stack.Navigator>
              {user ? (
                <Stack.Screen name='Home'>
                  {(props) => <HomeScreen {...props} user={user} />}
                </Stack.Screen>
              ) : (
                <>
                  <Stack.Screen name='Login' component={LoginScreen} />
                  <Stack.Screen
                    name='Registration'
                    component={RegistrationScreen}
                  />
                </>
              )}
            </Stack.Navigator>
          </NavigationContainer>
        </PaperProvider>
      </ThemeProvider>
    </ClickOutsideProvider>
  );
}
```

Файл index.js

```
export { default as LoginScreen } from './LoginScreen/LoginScreen';

export { default as HomeScreen } from './HomeScreen/HomeScreen';

export { default as RegistrationScreen } from './RegistrationScreen/
RegistrationScreen';
```

Файл HomeScreen.js

```
import { Provider } from '@react-native-material/core';
import { MaterialCommunityIcons } from '@expo/vector-icons';
import { createMaterialBottomTabNavigator } from '@react-navigation/material-
bottom-tabs';
import ColorPickerScreen from '../ColorPickerScreen/ColorPickerScreen';
import SavedPalettesScreen from '../SavedPalettesScreen/SavedPalettesScreen';
import PaletteGenerator from '../PaletteGenerator/PaletteGenerator';
import TrendsScreen from '../TrendsScreen/TrendsScreen';
import ColorAnalysisScreen from '../ColorAnalysisScreen/ColorAnalysisScreen';
import { useCustomTheme } from '../../contexts/themeContext';

const Tab = createMaterialBottomTabNavigator();

export default function HomeScreen({ user }) {
  const { theme } = useCustomTheme();

  return (
    <Provider theme={theme}>
      <Tab.Navigator
        initialRouteName='Palettes'
        activeColor={theme.palette.secondary?.main || '#390D7F'}
        inactiveColor={theme.typography.caption?.color}
        barStyle={{ backgroundColor: theme.palette.border?.main }}
        shifting
      >
        <Tab.Screen
          name='Palettes'
          options={{
            tabBarLabel: 'Palettes',
            tabBarIcon: ({ color }) => (
              <MaterialCommunityIcons name='palette' color={color}
size={26} />
            ),
          }}
        >
          {(props) => (
            <SavedPalettesScreen
              {...props}
              userId={user.id}
              userName={user.fullName}
            />
          )}
        </Tab.Screen>
        <Tab.Screen
          name='Color Picker'
          options={{
            tabBarLabel: 'Color Picker',
            tabBarIcon: ({ color }) => (
              <MaterialCommunityIcons
                name='invert-colors'
                color={color}
                size={26}
              />
            ),
          }}
```

```
        >
          {(props) => <ColorPickerScreen {...props} userId={user.id} />}
        </Tab.Screen>
        <Tab.Screen
          name='Palette Generator'
          options={{
            tabBarLabel: 'Palette Generator',
            tabBarIcon: ({ color }) => (
              <MaterialCommunityIcons name='brain' color={color} size={26} />
            ),
          }}
        >
          {(props) => <PaletteGenerator {...props} userId={user.id} />}
        </Tab.Screen>
        <Tab.Screen
          name='Trends'
          options={{
            tabBarLabel: 'Trends',
            tabBarIcon: ({ color }) => (
              <MaterialCommunityIcons
                name='trending-up'
                color={color}
                size={26}
              />
            ),
          }}
        >
          {(props) => <TrendsScreen {...props} userId={user.id} />}
        </Tab.Screen>
        <Tab.Screen
          name='Analyze'
          options={{
            tabBarLabel: 'Analyze',
            tabBarIcon: ({ color }) => (
              <MaterialCommunityIcons
                name='home-analytics'
                size={26}
                color={color}
              />
            ),
          }}
        >
          {(props) => <ColorAnalysisScreen {...props} userId={user.id} />}
        </Tab.Screen>
      </Tab.Navigator>
    </Provider>
  );
}
```

Файл ColorAnalysisScreen.js

```
import React, { useState } from 'react';
import { View } from 'react-native';
import Color from 'color';
import { Button, Flex, Text, TextInput } from '@react-native-material/core';
import ColorSwatch from '../../components/ColorSwatch/ColorSwatch';
import { ScrollView } from 'react-native-gesture-handler';
import usePaletteActions from '../../hooks/usePaletteActions';
import Notification from '../../components/Notification/Notification';
import { generateAIPalettes } from '../../helpers/generateAIPalette';
import ColorPalette from '../../components/ColorPallete/ColorPallete';
import styles from './styles';

const getContrastStatus = (contrast) => {
```

```
  if (contrast < 2.5) {
    return 'very low';
  }

  if (contrast >= 2.5 && contrast < 4.5) {
    return 'low';
  }

  if (contrast >= 4.5 && contrast < 7.5) {
    return 'medium';
  }

  return 'high';
};

export default function ColorAnalysisScreen({ userId }) {
  const [mainColor, setMainColor] = useState('');
  const [contrastColor, setContrastColor] = useState('#FFFFFF');

  const [analytics, setAnalytics] = useState();

  const [complementaryColor, setComplementaryColor] = useState('');
  const [analogousColors, setAnalogousColors] = useState([]);
  const [splitComplementaryColors, setSplitComplementaryColors] =
useState([]);

  const [contrastColors, setContrastColors] = useState([]);

  const [loading, setLoading] = useState(false);
  const [aiPalletes, setAIPalettes] = useState([]);

  const { onCopy, notification } = usePaletteActions({});

  const analyzeColor = async () => {
    try {
      setLoading(true);

      const color = Color(mainColor, 'hex');
      setAnalytics({
        main: color,
        contrast: Color(contrastColor, 'hex'),
      });

      const hsl = color.hsl().object();

      // Calculate complementary color
      const complementary = Color({
        h: (hsl.h + 180) % 360,
        s: hsl.s,
        l: hsl.l,
      });
      setComplementaryColor(complementary.hex());

      // Calculate analogous colors
      const analogous1 = Color({
        h: (hsl.h - 30 + 360) % 360,
        s: hsl.s,
        l: hsl.l,
      });
      const analogous2 = Color({ h: (hsl.h + 30) % 360, s: hsl.s, l:
hsl.l });
      setAnalogousColors([analogous1.hex(), analogous2.hex()]);

      // Calculate split-complementary colors
      const splitComplementary1 = Color({
        h: (hsl.h + 150) % 360,
```

```
      s: hsl.s,
      l: hsl.l,
    });
    const splitComplementary2 = Color({
      h: (hsl.h + 210) % 360,
      s: hsl.s,
      l: hsl.l,
    });
    setSplitComplementaryColors([
      splitComplementary1.hex(),
      splitComplementary2.hex(),
    ]);

    const contrastList = [];

    for (let i = 0; i < 6; i++) {
      const contrastingColor = Color({
        h: (hsl.h + i * 60) % 360,
        s: hsl.s,
        l: 100 - hsl.l,
      });
      contrastList.push(contrastingColor.hex());
    }

    setContrastColors(contrastList);

    const data = await generateAIPalettes(
      'diffusion',
      4,
      1.2,
      3,
      [
        '0',
        '65',
        '45',
        '35',
        '65',
        '0',
        '35',
        '65',
        '45',
        '35',
        '0',
        '35',
        '35',
        '65',
        '35',
        '0',
      ],
      [color.hex(), '-', '-', '-']
    );
    setAIPalettes(data);
  } catch (error) {
    alert('Wrong color! Enter a hex format...');
  } finally {
    setLoading(false);
  }
};

return (
  <ScrollView>
    <View style={styles.container}>
      <Flex direction='row' justify='between' mb={20}>
        <TextInput
          style={styles.colorInput}
          label='Enter main color:'
```

```
            value={mainColor}
            onChangeText={setMainColor}
            variant='standard'
          />
          <ColorSwatch active={true} color={mainColor} />
        </Flex>

        <Flex direction='row' justify='between' mb={20}>
          <TextInput
            style={styles.colorInput}
            label='Enter contrast color:'
            value={contrastColor}
            onChangeText={setContrastColor}
            variant='standard'
          />
          <ColorSwatch active={true} color={contrastColor} />
        </Flex>

        <Button
          title='Analyze Color'
          onPress={analyzeColor}
          loading={loading}
        />

        {analytics && (
          <Flex
            wrap='wrap'
            justify='between'
            direction='row'
            style={styles.statisticsContainer}
            mt={30}
          >
            <Text variant='body2' style={styles.statItem}>
              <Text style={styles.boldText}>CMYK:</Text>{' '}
              {analytics.main.cmyk().round().string()}
            </Text>
            <Text variant='body2' style={styles.statItem}>
              <Text style={styles.boldText}>HSV:</Text>{' '}
              {analytics.main.hsv().round().string()}
            </Text>
            <View style={styles.divider} />
            <Text variant='body2' style={styles.statItem}>
              <Text style={styles.boldText}>HSL:</Text>{' '}
              {analytics.main.hsl().round().string()}
            </Text>
            <Text variant='body2' style={styles.statItem}>
              <Text style={styles.boldText}>Contrast:</Text>{' '}
              {analytics.main.contrast(Color(analytics.contrast)).toFixed(2)}
(
              {getContrastStatus(
                analytics.main.contrast(Color(analytics.contrast))
              )}
              )
            </Text>
            <View style={styles.divider} />

            <Text variant='body2' style={styles.statItem}>
              <Text style={styles.boldText}>Complementary Color:</Text>
            </Text>
            <View style={styles.statItem}>
              <ColorSwatch
                active={true}
                color={complementaryColor}
                handlePress={() => onCopy(complementaryColor)}
              />
            </View>
```

```
              <View style={styles.divider} />

              <Text variant='body2'>
                <Text style={styles.boldText}>Analogous Colors:</Text>
              </Text>
              <Flex direction='row' style={styles.statItem}>
                {analogousColors.map((color, index) => (
                  <ColorSwatch
                    key={index}
                    active={true}
                    color={color}
                    handlePress={() => onCopy(color)}
                  />
                ))}
              </Flex>
              <View style={styles.divider} />

              <Text variant='body2' style={styles.statItem}>
                <Text style={styles.boldText}>Split-Complementary Colors:</
Text>
              </Text>
              <Flex direction='row' style={styles.statItem}>
                {splitComplementaryColors.map((color, index) => (
                  <ColorSwatch
                    key={index}
                    active={true}
                    color={color}
                    handlePress={() => onCopy(color)}
                  />
                ))}
              </Flex>
              <View style={styles.divider} />

              <Text variant='body2' style={styles.statItem}>
                <Text style={styles.boldText}>Contrast Colors:</Text>
              </Text>
              <Flex direction='row' style={styles.statItem}>
                {contrastColors.map((color, index) => (
                  <ColorSwatch
                    key={index}
                    active={true}
                    color={color}
                    handlePress={() => onCopy(color)}
                  />
                ))}
              </Flex>
              <View style={styles.divider} />

              <Text variant='body2' style={styles.statItem}>
                <Text style={styles.boldText}>Suggested AI palettes:</Text>
              </Text>
              <Flex style={{ width: '100%' }}>
                {aiPalletes.map((item, index) => (
                  <ColorPalette
                    key={`${item.score}-${index}}`}
                    canCreate={true}
                    colors={item.palette}
                    spacing={20}
                    userId={userId}
                  />
                ))}
              </Flex>
            </Flex>
          )}
          {notification && (
            <Notification message={`Text Copied: ${notification}`} />
```

```
        )}
      </View>
    </ScrollView>
  );
}
```

Файл ColorAnalysisScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
    paddingHorizontal: 20,
    paddingVertical: 20,
  },
  colorInput: {
    width: '50%',
  },
  statisticsContainer: {
    rowGap: 15,
  },
  statItem: {
    maxWidth: '49%',
    width: '49%',
    textAlign: 'left',
    gap: 10,
    flexShrink: 1,
    flexWrap: 'wrap',
  },
  boldText: {
    fontWeight: 'bold',
    fontSize: 14,
  },
  divider: {
    height: 1,
    width: '100%',
    backgroundColor: 'black',
  },
});
```

Файл ColorPickerScreen.js

```
import React, { useState, useRef, useEffect } from 'react';
import { ScrollView, View } from 'react-native';
import ColorPicker from 'react-native-wheel-color-picker';
import { Button, Flex, TextInput } from '@react-native-material/core';
import ColorPalette from '../../components/ColorPallete/ColorPallete';
import styles from './styles';
import ColorSwatch from '../../components/ColorSwatch/ColorSwatch';
import Notification from '../../components/Notification/Notification';
import usePaletteActions from '../../hooks/usePaletteActions';
import Color from 'color';

export default function ColorPickerScreen({ userId, route }) {
  const palleteInfo = route.params?.palleteInfo || {};
  const [colorPallete, setColorPallete] = useState(palleteInfo.colors || []);
  const [currentColor, setCurrentColor] = useState('#000000');
  const [colorInput, setColorInput] = useState('#000000');
  const [selectedColorIdx, setSelectedColorIdx] = useState();

  const { onCopy, notification } = usePaletteActions({});
```

```
const onColorChange = (color) => {
  setCurrentColor(color);
  setColorInput(color);
};

const onCommitColor = () => {
  if (selectedColorIdx === undefined) {
    if (!colorPallete.includes(currentColor)) {
      setColorPallete([...colorPallete, currentColor]);
    }
  } else {
    const duplicate = [...colorPallete];
    duplicate[selectedColorIdx] = currentColor;
    setColorPallete(duplicate);
  }
};

const onColorRemove = (idx) => {
  const duplicate = [...colorPallete];
  duplicate.splice(idx, 1);
  if (selectedColorIdx === idx) {
    setSelectedColorIdx(undefined);
  } else if (selectedColorIdx > idx) {
    setSelectedColorIdx(selectedColorIdx - 1);
  }
  setColorPallete(duplicate);
};

const onColorSelect = (idx) => {
  if (idx === selectedColorIdx) {
    setSelectedColorIdx(undefined);
  } else {
    setSelectedColorIdx(idx);
  }
};

useEffect(() => {
  try {
    const parsedColor = Color(colorInput);
    hex = parsedColor.hex();
    setCurrentColor(hex);
  } catch (error) {}
}, [colorInput]);

useEffect(() => {
  setColorPallete(palleteInfo.colors || []);
}, [route]);

return (
  <ScrollView>
    <View style={styles.pickerContainer}>
      <ColorPicker
        color={currentColor}
        onColorChangeComplete={onColorChange}
        thumbSize={30}
        sliderSize={20}
        noSnap={true}
        row={false}
        palette={palleteInfo.colors}
      />
    </View>
    <View style={styles.palleteContainer}>
      <Flex direction='row' justify='between' items='center'>
        <TextInput
          style={styles.colorInput}
          label='Enter color:'
```

```
            value={colorInput}
            onChangeText={setColorInput}
            variant='standard'
          />
          <ColorSwatch
            active={true}
            color={currentColor}
            handlePress={() => onCopy(currentColor)}
          />
        </Flex>
        <Button
          title={
            selectedColorIdx === undefined
              ? 'Add color to palette'
              : 'Update selected color'
          }
          color='green'
          onPress={onCommitColor}
        />
        <ColorPalette
          canCreate
          colors={colorPallete}
          onColorRemove={onColorRemove}
          onColorSelect={onColorSelect}
          onClear={() => setColorPallete([])}
          userId={userId}
          id={palleteInfo.id}
          name={palleteInfo.name}
          activeColor={selectedColorIdx}
        />
        {notification && (
          <Notification message={`Text Copied: ${notification}`} />
        )}
      </View>
    </ScrollView>
  );
}
```

Файл ColorPickerScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  pickerContainer: {
    paddingHorizontal: 24,
  },
  palleteContainer: {
    marginTop: 50,
    paddingHorizontal: 24,
    gap: 20,
  },
  colorInput: {
    width: 100,
  },
});
```

Файл LoginScreen.js

```
import React, { useState } from 'react';
import { Image, Text, TextInput, TouchableOpacity, View } from 'react-
native';
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view';
```

```
import styles from './styles';
import {
  auth,
  db,
  doc,
  getDoc,
  signInWithEmailAndPassword,
} from '../../firebase/config';

const regex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w\w+)+$/;

export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const onFooterLinkPress = () => {
    navigation.navigate('Registration');
  };

  const onLoginPress = async () => {
    try {
      const resp = await signInWithEmailAndPassword(auth, email, password);
      const uid = resp.user.uid;
      const userRef = doc(db, 'users', uid);

      const firestoreDocument = await getDoc(userRef);
      if (firestoreDocument.exists()) {
        const user = firestoreDocument.data();
        navigation.navigate('Home', { user });
      } else {
        alert('User does not exist anymore.');
        return;
      }
    } catch (error) {
      if (error.code === 'auth/wrong-password') {
        alert('You entered wrong password...');
      } else {
        alert('User not found...');
      }
    }
  };

  return (
    <View style={styles.container}>
      <KeyboardAwareScrollView
        style={{ flex: 1, width: '100%' }}
        keyboardShouldPersistTaps='always'
      >
        <Image
          style={styles.logo}
          source={require('../../../assets/icon.png')}
        />
        <TextInput
          style={styles.input}
          placeholder='E-mail'
          placeholderTextColor='#aaaaaa'
          onChangeText={(text) => setEmail(text)}
          value={email}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TextInput
          style={styles.input}
          placeholderTextColor='#aaaaaa'
          secureTextEntry
          placeholder='Password'
```

```
          onChangeText={(text) => setPassword(text)}
          value={password}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TouchableOpacity
          style={[
            styles.button,
            { backgroundColor: !regex.test(email) ? 'lightgrey' :
'#788eec' },
          ]}
          onPress={() => onLoginPress()}
          disabled={!regex.test(email)}
        >
          <Text style={styles.buttonTitle}>Log in</Text>
        </TouchableOpacity>
        <View style={styles.footerView}>
          <Text style={styles.footerText}>
            Don't have an account?{' '}
            <Text onPress={onFooterLinkPress} style={styles.footerLink}>
              Sign up
            </Text>
          </Text>
        </View>
      </KeyboardAwareScrollView>
    </View>
  );
}
```

Файл LoginScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
  },
  title: {},
  logo: {
    flex: 1,
    height: 120,
    width: 90,
    alignSelf: 'center',
    margin: 30,
  },
  input: {
    height: 48,
    borderRadius: 5,
    overflow: 'hidden',
    backgroundColor: 'white',
    marginTop: 10,
    marginBottom: 10,
    marginLeft: 30,
    marginRight: 30,
    paddingLeft: 16,
  },
  button: {
    marginLeft: 30,
    marginRight: 30,
    marginTop: 20,
    height: 48,
    borderRadius: 5,
    alignItems: 'center',
```

```
    justifyContent: 'center',
  },
  buttonTitle: {
    color: 'white',
    fontSize: 16,
    fontWeight: 'bold',
  },
  footerView: {
    flex: 1,
    alignItems: 'center',
    marginTop: 20,
  },
  footerText: {
    fontSize: 16,
    color: '#2e2e2d',
  },
  footerLink: {
    color: '#788eec',
    fontWeight: 'bold',
    fontSize: 16,
  },
});
```

Файл PaletteGenerator.js

```
import React, { useRef, useState } from 'react';
import { View, ScrollView } from 'react-native';
import { RadioButton } from 'react-native-paper';
import { Flex, Text, TextInput, Button } from '@react-native-material/core';
import ColorPalette from '../../components/ColorPallete/ColorPallete';
import { generateAIPalettes } from '../../helpers/generateAIPalette';
import styles from './styles';

export default function PaletteGenerator({ route, navigation, userId }) {
  const [mode, setMode] = useState('transformer');
  const [numColors, setNumColors] = useState('4');
  const [temperature, setTemperature] = useState('1.2');
  const [adjacency, setAdjacency] = useState([
    '0',
    '65',
    '45',
    '35',
    '65',
    '0',
    '35',
    '65',
    '45',
    '35',
    '0',
    '35',
    '35',
    '65',
    '35',
    '0',
  ]);
  const [palette, setPalette] = useState(['#000000', '-', '-', '-']);
  const [generatedPalettes, setGeneratedPalettes] = useState([]);
  const [loading, setLoading] = useState(false);
  const [selected, setSelected] = useState({});
  const scrollViewRef = useRef(null);

  const handleReset = () => {
    setMode('transformer');
    setNumColors('4');
```

```
    setTemperature('1.2');
    setAdjacency([
      '0',
      '65',
      '45',
      '35',
      '65',
      '0',
      '35',
      '65',
      '45',
      '35',
      '0',
      '35',
      '35',
      '65',
      '35',
      '0',
    ]);
    setPalette(['#000000', '-', '-', '-']);
    setGeneratedPalettes([]);
  };

  const handleSubmit = async () => {
    if (!validateForm()) {
      return;
    }

    try {
      setLoading(true);
      const data = await generateAIPalettes(
        mode,
        numColors,
        temperature,
        10,
        adjacency,
        palette
      );
      setGeneratedPalettes(data);
    } catch (error) {
      alert('You entered wrong form data...');
    } finally {
      setLoading(false);
    }
  };

  const validateForm = () => {
    if (
      numColors > 12 ||
      numColors < 2 ||
      temperature < 0 ||
      temperature > 2.4
    ) {
      alert('Please fill in all required fields.');
      return false;
    }

    if (!(adjacency.length > 1)) {
      alert('Please provide a valid adjacency matrix.');
      return false;
    }

    if (!(palette.length > 1)) {
      alert('Please provide a valid palette.');
      return false;
    }
```

```
    return true;
  };

  const onColorRemove = (paletteIdx) => {
    return (colorIdx) => {
      const { color, palette } = selected;
      const duplicate = JSON.parse(JSON.stringify(generatedPalettes));
      duplicate[paletteIdx].palette.splice(colorIdx, 1);
      if (paletteIdx === palette && color === colorIdx) {
        setSelected({});
      } else if (paletteIdx === palette && color > colorIdx) {
        setSelected({
          palette,
          color: color - 1,
        });
      }
      setGeneratedPalettes(duplicate);
    };
  };

  const onColorSelect = (paletteIdx) => {
    return (colorIdx) => {
      const { color, palette } = selected;
      if (paletteIdx === palette && colorIdx === color) {
        setSelected({});
      } else {
        setSelected({
          palette: paletteIdx,
          color: colorIdx,
        });
      }
    };
  };

  return (
    <ScrollView
      ref={scrollViewRef}
      onContentSizeChange={() =>
        scrollViewRef.current.scrollTo({
          y: 500,
          animated: true,
        })
      }
    >
      <View style={styles.container}>
        <Text variant='caption' style={{ marginLeft: 10 }}>
          Mode:
        </Text>
        <RadioButton.Group
          onValueChange={(value) => setMode(value)}
          value={mode}
        >
          <View style={styles.radioGroup}>
            <RadioButton.Item
              style={styles.radioGroupItem}
              labelVariant='bodyMedium'
              label='Transformer'
              value='transformer'
            />
            <RadioButton.Item
              style={styles.radioGroupItem}
              labelVariant='bodyMedium'
              label='Diffusion'
              value='diffusion'
            />
```

```
      <RadioButton.Item
        style={styles.radioGroupItem}
        labelVariant='bodyMedium'
        label='Random'
        value='random'
      />
    </View>
  </RadioButton.Group>
  <TextInput
    style={styles.input}
    label='Number of Colors (2 ≤ n ≤ 12):'
    value={numColors}
    onChangeText={(value) => setNumColors(value)}
    keyboardType='numeric'
  />
  <TextInput
    style={styles.input}
    placeholder='Enter temperature'
    label='Temperature (0 ≤ n ≤ 2.4):'
    value={temperature}
    onChangeText={(value) => setTemperature(value)}
    keyboardType='numeric'
  />
  <TextInput
    style={styles.input}
    label='Adjacency Matrix:'
    value={adjacency.join(',')}
    onChangeText={(value) => setAdjacency(value.split(','))}
  />
  <TextInput
    style={styles.input}
    label="Palette (hex codes or '-' if blank):"
    value={palette.join(',')}
    onChangeText={(value) => setPalette(value.split(','))}
  />
  <Flex direction='row' justify='space-between'>
    <Button
      variant='outlined'
      uppercase={false}
      onPress={handleReset}
      style={{ width: 140 }}
      title='Reset'
    />
    <Button
      loading={loading}
      uppercase={false}
      onPress={handleSubmit}
      style={{ width: 200 }}
      title='Generate Palettes'
    />
  </Flex>
  {generatedPalettes.length > 0 && (
    <>
      <Text variant='h4' style={styles.palettesTitle}>
        Generated Palettes
      </Text>
      <View style={styles.listContainer}>
        {generatedPalettes.map((item, index) => (
          <React.Fragment key={`${item.score}-${index}}`}>
            <Text
              variant='body2'
              style={{ marginLeft: 'auto', marginRight: 6 }}
            >
              Score: {item.score.toFixed(2)}
            </Text>
            <ColorPalette
```

```
                    canCreate={true}
                    colors={item.palette}
                    spacing={20}
                    userId={userId}
                    onColorRemove={onColorRemove(index)}
                    onColorSelect={onColorSelect(index)}
                    activeColor={
                      selected.palette === index ? selected.color : undefined
                    }
                  />
                </React.Fragment>
              ))}
            </View>
          </>
        )}
      </View>
    </ScrollView>
  );
}
```

Файл PaletteGenerator/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    gap: 10,
  },
  radioGroup: {
    flexDirection: 'column',
    gap: 5,
  },
  radioGroupItem: {
    width: 160,
    height: 50,
    borderBottomWidth: 1,
    borderTopWidth: 1,
    borderLeftWidth: 1,
    borderRightWidth: 1,
    borderColor: '#48464B',
    borderRadius: 20,
  },
  palettesTitle: {
    marginTop: 30,
    marginBottom: 10,
    marginLeft: 'auto',
    marginRight: 'auto',
  },
  listContainer: {
    padding: 10,
  },
});
```

Файл RegistrationScreen.js

```
import React, { useState } from 'react';
import { Image, Text, TextInput, TouchableOpacity, View } from 'react-
native';
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view';
import {
```

```
  db,
  createUserWithEmailAndPassword,
  collection,
  auth,
  setDoc,
  doc,
} from '../../firebase/config';
import styles from './styles';

export default function RegistrationScreen({ navigation }) {
  const [fullName, setFullName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');

  const onFooterLinkPress = () => {
    navigation.navigate('Login');
  };

  const onRegisterPress = async () => {
    if (password !== confirmPassword) {
      alert("Passwords don't match...");
      return;
    }
    try {
      const resp = await createUserWithEmailAndPassword(auth, email,
password);
      const uid = resp.user.uid;
      const data = {
        id: uid,
        email,
        fullName,
      };
      const usersRef = collection(db, 'users');
      await setDoc(doc(usersRef, uid), data);

      navigation.navigate('Home', { user: data });
    } catch (error) {
      alert(error);
    }
  };

  return (
    <View style={styles.container}>
      <KeyboardAwareScrollView
        style={{ flex: 1, width: '100%' }}
        keyboardShouldPersistTaps='always'
      >
        <Image
          style={styles.logo}
          source={require('../../../assets/icon.png')}
        />
        <TextInput
          style={styles.input}
          placeholder='Full Name'
          placeholderTextColor='#aaaaaa'
          onChangeText={(text) => setFullName(text)}
          value={fullName}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TextInput
          style={styles.input}
          placeholder='E-mail'
          placeholderTextColor='#aaaaaa'
          onChangeText={(text) => setEmail(text)}
```

```
          value={email}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TextInput
          style={styles.input}
          placeholderTextColor='#aaaaaa'
          secureTextEntry
          placeholder='Password'
          onChangeText={(text) => setPassword(text)}
          value={password}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TextInput
          style={styles.input}
          placeholderTextColor='#aaaaaa'
          secureTextEntry
          placeholder='Confirm Password'
          onChangeText={(text) => setConfirmPassword(text)}
          value={confirmPassword}
          underlineColorAndroid='transparent'
          autoCapitalize='none'
        />
        <TouchableOpacity style={styles.button} onPress={onRegisterPress}>
          <Text style={styles.buttonTitle}>Create account</Text>
        </TouchableOpacity>
        <View style={styles.footerView}>
          <Text style={styles.footerText}>
            Already got an account?{' '}
            <Text onPress={onFooterLinkPress} style={styles.footerLink}>
              Log in
            </Text>
          </Text>
        </View>
      </KeyboardAwareScrollView>
    </View>
  );
}
```

Файл RegistrationScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
  },
  title: {},
  logo: {
    flex: 1,
    height: 120,
    width: 90,
    alignSelf: 'center',
    margin: 30,
  },
  input: {
    height: 48,
    borderRadius: 5,
    overflow: 'hidden',
    backgroundColor: 'white',
    marginTop: 10,
    marginBottom: 10,
```

```
    marginLeft: 30,
    marginRight: 30,
    paddingLeft: 16,
  },
  button: {
    backgroundColor: '#788eec',
    marginLeft: 30,
    marginRight: 30,
    marginTop: 20,
    height: 48,
    borderRadius: 5,
    alignItems: 'center',
    justifyContent: 'center',
  },
  buttonTitle: {
    color: 'white',
    fontSize: 16,
    fontWeight: 'bold',
  },
  footerView: {
    flex: 1,
    alignItems: 'center',
    marginTop: 20,
  },
  footerText: {
    fontSize: 16,
    color: '#2e2e2d',
  },
  footerLink: {
    color: '#788eec',
    fontWeight: 'bold',
    fontSize: 16,
  },
});
```

Файл SavedPalettesScreen.js

```
import React, { useEffect, useState } from 'react';
import { ScrollView, TextInput, View } from 'react-native';
import { Button, Text } from '@react-native-material/core';
import {
  collection,
  db,
  onSnapshot,
  orderBy,
  query,
  where,
} from '../../firebase/config';
import ColorPalette from '../../components/ColorPallete/ColorPallete';
import { useCustomTheme } from '../../contexts/themeContext';
import styles from './styles';

export default function SavedPalettesScreen({ userId, userName }) {
  const [search, setSearch] = useState('');
  const [palettes, setPalettes] = useState([]);
  const { resetTheme } = useCustomTheme();

  const palettesRef = collection(db, 'palletes');

  useEffect(() => {
    const q = query(
      palettesRef,
      where('authorID', '==', userId),
      orderBy('createdAt', 'desc')
```

```
    );

    const unsubscribe = onSnapshot(
      q,
      (querySnapshot) => {
        const newPalettes = [];
        querySnapshot.forEach((doc) => {
          const palette = doc.data();
          palette.id = doc.id;
          newPalettes.push(palette);
        });
        setPalettes(newPalettes);
      },
      (error) => {
        console.log(error);
      }
    );

    return () => {
      unsubscribe();
    };
  }, []);

  return (
    <ScrollView>
      <View style={styles.container}>
        <View style={styles.formContainer}>
          <TextInput
            style={styles.input}
            placeholder='Find Palette'
            placeholderTextColor='#aaaaaa'
            onChangeText={setSearch}
            value={search}
            underlineColorAndroid='transparent'
            autoCapitalize='none'
          />
        </View>
        <Button
          style={styles.resetButton}
          title='Reset Theme'
          variant='outlined'
          uppercase={false}
          onPress={resetTheme}
        />
        <Text variant='h4'>Palettes</Text>
        {palettes.length > 0 && (
          <View style={styles.listContainer}>
            {palettes
              .filter((palette) =>
                palette.name.toLowerCase().includes(search.toLowerCase())
              )
              .map((item) => (
                <React.Fragment key={item.colors.toString()}>
                  <Text variant='body2' style={{ marginLeft: 6 }}>
                    Palette {item.name}
                  </Text>
                  <ColorPalette
                    canEdit
                    canCreate={true}
                    colors={item.colors}
                    spacing={20}
                    id={item.id}
                    name={item.name}
                    userId={userId}
                    userName={userName}
                  />
```

```
                </React.Fragment>
              ))}
            </View>
          )}
        </View>
      </ScrollView>
    );
}
```

Файл SavedPalettesScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
  },
  formContainer: {
    flexDirection: 'row',
    height: 80,
    marginTop: 40,
    flex: 1,
    paddingTop: 10,
    paddingBottom: 10,
    paddingLeft: 30,
    paddingRight: 30,
    justifyContent: 'center',
    alignItems: 'center',
  },
  input: {
    height: 48,
    borderRadius: 5,
    overflow: 'hidden',
    backgroundColor: 'white',
    paddingLeft: 16,
    flex: 1,
    marginRight: 5,
  },
  button: {
    height: 47,
    borderRadius: 5,
    backgroundColor: '#788eec',
    width: 80,
    alignItems: 'center',
    justifyContent: 'center',
  },
  buttonText: {
    color: 'white',
    fontSize: 16,
  },
  listContainer: {
    padding: 20,
  },
  entityContainer: {
    marginTop: 16,
    borderBottomColor: '#cccccc',
    borderBottomWidth: 1,
    paddingBottom: 16,
  },
  entityText: {
    fontSize: 20,
    color: '#333333',
  },
```

```
  footerView: {
    flex: 1,
    alignItems: 'center',
    marginTop: 20,
  },
  footerLink: {
    color: '#788eec',
    fontWeight: 'bold',
    fontSize: 16,
  },
  resetButton: {
    marginBottom: 20,
  },
});
```

Файл TrendsScreen.js

```
import React, { useEffect, useState } from 'react';
import { ScrollView, TextInput, View } from 'react-native';
import { Text } from '@react-native-material/core';
import {
  collection,
  db,
  onSnapshot,
  orderBy,
  query,
} from '../../firebase/config';
import ColorPalette from '../../components/ColorPallete/ColorPallete';
import styles from './styles';

export default function TrendsScreen({ userId }) {
  const [search, setSearch] = useState('');
  const [palettes, setPalettes] = useState([]);

  const trendsRef = collection(db, 'trends');

  useEffect(() => {
    const q = query(
      trendsRef,
      orderBy('likes', 'desc'),
      orderBy('createdAt', 'desc')
    );

    const unsubscribe = onSnapshot(
      q,
      (querySnapshot) => {
        const newPalettes = [];
        querySnapshot.forEach((doc) => {
          const palette = doc.data();
          palette.id = doc.id;
          newPalettes.push(palette);
        });
        setPalettes(newPalettes);
      },
      (error) => {
        console.log(error);
      }
    );

    return () => {
      unsubscribe();
    };
  }, []);
```

```
  return (
    <ScrollView>
      <View style={styles.container}>
        <View style={styles.formContainer}>
          <TextInput
            style={styles.input}
            placeholder='Find Palette'
            placeholderTextColor='#aaaaaa'
            onChangeText={setSearch}
            value={search}
            underlineColorAndroid='transparent'
            autoCapitalize='none'
          />
        </View>
        <Text variant='h4'>Palettes</Text>
        {palettes.length > 0 && (
          <View style={styles.listContainer}>
            {palettes
              .filter(
                (palette) =>
                  palette.name.toLowerCase().includes(search.toLowerCase())
||
                  palette.authorName
                    .toLowerCase()
                    .includes(search.toLowerCase())
              )
              .map((item) => (
                <React.Fragment key={`${item.authorID}-${item.name}`}>
                  <Text
                    variant='body2'
                    color={userId === item.authorID ? 'green' : undefined}
                    style={{ marginLeft: 6 }}
                  >
                    {item.authorName}: Palette {item.name}
                  </Text>
                  <ColorPalette
                    canLike={true}
                    canEdit={false}
                    canCreate={userId !== item.authorID}
                    colors={item.colors}
                    spacing={20}
                    id={item.id}
                    userId={userId}
                    name={item.name}
                    likes={item.likes}
                    likedBy={item.likedBy}
                    author={item.authorID}
                  />
                </React.Fragment>
              ))}
          </View>
        )}
      </View>
    </ScrollView>
  );
}
```

Файл TrendsScreen/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    flex: 1,
```

```
    alignItems: 'center',
  },
  formContainer: {
    flexDirection: 'row',
    height: 80,
    marginTop: 40,
    marginBottom: 20,
    flex: 1,
    paddingTop: 10,
    paddingBottom: 10,
    paddingLeft: 30,
    paddingRight: 30,
    justifyContent: 'center',
    alignItems: 'center',
  },
  input: {
    height: 48,
    borderRadius: 5,
    overflow: 'hidden',
    backgroundColor: 'white',
    paddingLeft: 16,
    flex: 1,
    marginRight: 5,
  },
  button: {
    height: 47,
    borderRadius: 5,
    backgroundColor: '#788eec',
    width: 80,
    alignItems: 'center',
    justifyContent: 'center',
  },
  buttonText: {
    color: 'white',
    fontSize: 16,
  },
  listContainer: {
    padding: 20,
  },
  entityContainer: {
    marginTop: 16,
    borderBottomColor: '#cccccc',
    borderBottomWidth: 1,
    paddingBottom: 16,
  },
  entityText: {
    fontSize: 20,
    color: '#333333',
  },
  footerView: {
    flex: 1,
    alignItems: 'center',
    marginTop: 20,
  },
  footerLink: {
    color: '#788eec',
    fontWeight: 'bold',
    fontSize: 16,
  },
});
```

Файл usePaletteActions.js

```
import { useRef, useState } from 'react';
```

```
import { Keyboard } from 'react-native';
import * as FileSystem from 'expo-file-system';
import * as Sharing from 'expo-sharing';
import * as Clipboard from 'expo-clipboard';
import { unparse } from 'papaparse';
import {
  addDoc,
  collection,
  db,
  deleteDoc,
  doc,
  serverTimestamp,
  setDoc,
} from '../firebase/config';
import { useCustomTheme } from '../contexts/themeContext';

export default function usePaletteActions({
  palleteName,
  userId,
  id,
  colors,
  name,
  author,
  userName,
  likedBy,
  likes,
}) {
  const notificationTimer = useRef();
  const [notification, setNotification] = useState('');
  const [isConfirmation, setConfirmation] = useState(false);

  const { updateTheme } = useCustomTheme();

  const palletesRef = collection(db, 'palletes');
  const trendsRef = collection(db, 'trends');

  const onSavePallete = async () => {
    try {
      const timestamp = serverTimestamp();
      const data = {
        name: palleteName,
        authorID: userId,
        createdAt: timestamp,
        colors,
      };
      if (!id) {
        await addDoc(palletesRef, data);
      } else {
        await setDoc(doc(palletesRef, id), data);
      }
    } catch (error) {
      console.log(error);
    }

    Keyboard.dismiss();
    setConfirmation(false);
  };

  const onDeletePallete = async () => {
    await deleteDoc(doc(db, 'palletes', id));
  };

  const onConvertAndShareCSVPalette = async () => {
    try {
      const csvData = unparse([colors]);
      const filePath = `${FileSystem.cacheDirectory}palette-${name}.csv`;
```

```
      await FileSystem.writeAsStringAsync(filePath, csvData, {
        encoding: FileSystem.EncodingType.UTF8,
      });

      // Share the CSV file
      await Sharing.shareAsync(filePath, {
        mimeType: 'text/csv',
        dialogTitle: `Share Palette ${name}`,
      });
    } catch (error) {
      alert(error);
    }
  };

  const onCopy = async (color) => {
    setNotification(undefined);
    clearTimeout(notificationTimer.current);
    // const copiedText = await Clipboard.getStringAsync();
    // if (copiedText === color) {
    //   return;
    // }
    if (color) {
      await Clipboard.setStringAsync(color);
      setNotification(color);
    } else {
      await Clipboard.setStringAsync(colors.join(', '));
      setNotification(colors.join(', '));
    }

    // Hide the notification after a certain duration
    const timer = setTimeout(() => {
      setNotification(undefined);
    }, 2000);

    notificationTimer.current = timer;
  };

  const onPublish = async () => {
    try {
      const timestamp = serverTimestamp();
      const data = {
        name: palleteName,
        authorID: userId,
        authorName: userName,
        createdAt: timestamp,
        likes,
        colors,
        likedBy,
      };
      await setDoc(
        doc(trendsRef, `${userId}-${name}-${colors.join('-')}`),
        data
      );
    } catch (error) {
      console.log(error);
    }

    setConfirmation(false);
  };

  const onLike = async () => {
    try {
      const data = {};
      if (likedBy.includes(userId)) {
        data.likes = likes - 1;
        data.likedBy = likedBy.filter((id) => id !== userId);
```

```
      } else {
        data.likes = likes + 1;
        data.likedBy = [...likedBy, userId];
      }
      await setDoc(
        doc(trendsRef, `${author}-${name}-${colors.join('-')}`),
        data,
        { merge: true }
      );
    } catch (error) {
      console.log(error);
    }

    setConfirmation(false);
  };

  const onApplyTheme = () => {
    updateTheme(...colors);
  };

  return {
    onConvertAndShareCSVPalette,
    onCopy,
    onDeletePallete,
    onLike,
    onPublish,
    onSavePallete,
    onApplyTheme,
    setConfirmation,
    notification,
    isConfirmation,
  };
}
```

Файл generateAIPalette.js

```
const aiGeneratorUrl = 'https://api.huemint.com/color';

export const generateAIPalettes = async (
  mode,
  numColors,
  temperature,
  numResults,
  adjacency,
  palette
) => {
  try {
    const body = {
      mode,
      num_colors: parseInt(numColors),
      temperature: parseFloat(temperature),
      num_results: numResults,
      adjacency,
      palette,
    };

    const resp = await fetch(aiGeneratorUrl, {
      method: 'POST',
      mode: 'cors',
      cache: 'no-cache',
      credentials: 'same-origin',
      headers: {
        'Content-Type': 'application/json',
      },
```

```
      redirect: 'follow',
      referrerPolicy: 'no-referrer',
      body: JSON.stringify(body),
    });

    const data = await resp.json();

    return data.results;
  } catch (error) {
    throw new Error(error.message);
  }
};
```

Файл firebase/config.js

```
import { initializeApp } from 'firebase/app';
import {
  getAuth,
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
} from 'firebase/auth';
import {
  getFirestore,
  collection,
  doc,
  setDoc,
  getDoc,
  query,
  orderBy,
  limit,
  where,
  onSnapshot,
  serverTimestamp,
  addDoc,
  deleteDoc,
} from '@firebase/firestore';
import { Platform } from 'react-native';

const firebaseConfig = {
  apiKey: 'AIzaSyBTFHjEpMJ7wZGM--hxjr8MrNnq8pZU6zc',
  authDomain: 'color-pallete-app-e1a21.firebaseapp.com',
  databaseURL: 'https://your-database-name.firebaseio.com',
  projectId: 'color-pallete-app-e1a21',
  storageBucket: 'color-pallete-app-e1a21.appspot.com',
  messagingSenderId: '817401085747',
  appId:
    Platform.OS === 'android'
      ? '1:817401085747:android:614d35cf911d1f2287c846'
      : '1:817401085747:ios:025e2f6634d12cab87c846',
};

const app = initializeApp(firebaseConfig);
const auth = getAuth();

// Initialize Cloud Firestore and get a reference to the service
const db = getFirestore(app);

export {
  app,
  auth,
  db,
  doc,
```

```
  collection,
  setDoc,
  getDoc,
  query,
  orderBy,
  limit,
  where,
  onSnapshot,
  addDoc,
  deleteDoc,
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  onAuthStateChanged,
  serverTimestamp,
};
```

Файл themeContext.js

```
import React, { useContext, createContext, useState } from 'react';
import { defaultTheme } from '@react-native-material/core';

const ThemeContext = createContext();

export const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState(defaultTheme);

  const updateTheme = (text, primary, background, border, secondary, error)
=> {
    const def = defaultTheme.palette;
    const textColor = text
      ? {
          caption: { ...defaultTheme.typography.caption, color: text },
          body2: { ...defaultTheme.typography.body2, color: text },
        }
      : {
          caption: defaultTheme.typography.caption,
          body2: defaultTheme.typography.body2,
        };
    const primaryColor = primary
      ? { main: primary, on: def.primary.on }
      : def.primary;
    const secondaryColor = secondary
      ? { main: secondary, on: def.secondary.on }
      : def.secondary;
    const backgroundColor = background
      ? { main: background, on: def.background.on }
      : def.background;
    const borderColor = border
      ? { main: border, on: border }
      : {
          primary: '#C6B18E',
          on: '#3C84A1',
        };
    const errorColor = error ? { main: error, on: def.error.on } : def.error;
    const updatedTheme = {
      ...defaultTheme,
      typography: {
        ...defaultTheme.typography,
        ...textColor,
      },
      palette: {
        ...defaultTheme.palette,
        primary: primaryColor,
        secondary: secondaryColor,
```

```
        background: backgroundColor,
        border: borderColor,
        error: errorColor,
      },
    };
    setTheme(updatedTheme);
  };

  const resetTheme = () => {
    const def = defaultTheme.palette;
    const textColor = {
      caption: defaultTheme.typography.caption,
      body2: defaultTheme.typography.body2,
    };
    const primaryColor = def.primary;
    const secondaryColor = def.secondary;
    const backgroundColor = def.background;
    const borderColor = {
      primary: '#C6B18E',
      on: '#3C84A1',
    };
    const errorColor = def.error;
    const updatedTheme = {
      ...defaultTheme,
      typography: {
        ...defaultTheme.typography,
        ...textColor,
      },
      palette: {
        ...defaultTheme.palette,
        primary: primaryColor,
        secondary: secondaryColor,
        background: backgroundColor,
        border: borderColor,
        error: errorColor,
      },
    };
    setTheme(updatedTheme);
  };

  return (
    <ThemeContext.Provider value={{ theme, updateTheme, resetTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

export const useCustomTheme = () => {
  const context = useContext(ThemeContext);
  if (context === undefined) {
    throw new Error('useCustomTheme must be used within a ThemeProvider');
  }
  return context;
};
```

Файл Notification.js

```
import { Text, View } from 'react-native';
import styles from './styles';
import { Portal } from '@react-native-material/core';

export default function Notification({ message }) {
  return (
    <Portal>
```

```
      <View style={styles.container}>
        <Text style={styles.message}>{message}</Text>
      </View>
    </Portal>
  );
}
```

Файл Notification/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  container: {
    backgroundColor: 'rgba(0, 0, 0, 0.8)',
    borderRadius: 8,
    padding: 10,
    position: 'absolute',
    bottom: 114,
    right: 10,
    left: 10,
  },
  message: {
    color: '#fff',
    fontSize: 16,
    textAlign: 'center',
  },
});
```

Файл ColorSwatch.js

```
import React from 'react';
import { View, TouchableOpacity, Text } from 'react-native';
import { AntDesign } from '@expo/vector-icons';
import styles from './styles';

export default function ColorSwatch({
  active,
  color,
  onColorRemove,
  handlePress,
  handlePressIn,
  handlePressOut,
  isPressed = false,
}) {
  return (
    <View style={styles.swatchContainer}>
      <TouchableOpacity
        disabled={!handlePress}
        style={[
          styles.colorSwatch,
          {
            backgroundColor: color,
            ...(active
              ? {
                  borderWidth: 2,
                  borderColor: 'grey',
                }
              : {}),
          },
        ]}
        onPress={handlePress}
        onPressIn={handlePressIn}
        onPressOut={handlePressOut}
```

```
      >
        <Text style={styles.colorHint}>{color}</Text>
      </TouchableOpacity>
      {isPressed && onColorRemove && (
        <TouchableOpacity
          style={styles.removeSwatchIcon}
          onPress={onColorRemove}
        >
          <AntDesign name='closecircleo' size={24} color='red' />
        </TouchableOpacity>
      )}
    </View>
  );
}
```

Файл ColorPallete.js

```
import React, { useRef, useState } from 'react';
import { View, TouchableOpacity, Text, Dimensions } from 'react-native';
import { useNavigation } from '@react-navigation/native';
import {
  Button,
  Dialog,
  DialogActions,
  DialogContent,
  DialogHeader,
  Flex,
  Stack,
  TextInput,
} from '@react-native-material/core';
import {
  Octicons,
  AntDesign,
  Entypo,
  Feather,
  MaterialCommunityIcons,
  MaterialIcons,
  Ionicons,
} from '@expo/vector-icons';
import { useClickOutside } from 'react-native-click-outside';
import styles from './styles';
import Notification from '../Notification/Notification';
import ColorSwatch from '../ColorSwatch/ColorSwatch';
import usePaletteActions from '../../hooks/usePaletteActions';

function ColorPalette({
  colors,
  onColorSelect,
  onColorRemove,
  onClear,
  userId,
  userName,
  canLike = false,
  canCreate = false,
  canEdit = false,
  spacing,
  id,
  name = '',
  activeColor,
  likes = 0,
  likedBy = [],
  author,
}) {
  const navigation = useNavigation();
```

```
const [isPressed, setIsPressed] = useState(false);
const [palleteName, setPalleteName] = useState(name);
const pressTimer = useRef();
const paletteRef = useClickOutside(() => setIsPressed(false));

const {
  onConvertAndShareCSVPalette,
  onCopy,
  onDeletePallete,
  onLike,
  onPublish,
  onSavePallete,
  onApplyTheme,
  setConfirmation,
  notification,
  isConfirmation,
} = usePaletteActions({
  palleteName,
  userId,
  id,
  colors,
  name,
  author,
  userName,
  likedBy,
  likes,
});

const handlePressIn = () => {
  if (!canEdit) {
    const timer = setTimeout(() => {
      setIsPressed(true);
    }, 800);

    pressTimer.current = timer;
  }
};

const handlePressOut = () => {
  if (!canEdit) {
    clearTimeout(pressTimer.current);
  }
};

const handlePress = (idx) => {
  if (onColorSelect) {
    onColorSelect(idx);
  }
  onCopy(colors[idx]);
};

return (
  <Flex direction='column' style={{ marginBottom: spacing }}>
    <Flex direction='row'>
      <View
        style={[
          styles.paletteContainer,
          {
            width:
              canEdit || canLike
                ? Dimensions.get('window').width - 134
                : '100%',
          },
        ]}
        ref={paletteRef}
      >
```

```
        {colors.length > 0 ? (
          colors.map((color, index) => (
            <ColorSwatch
              key={color}
              color={color}
              active={activeColor === index}
              onColorRemove={
                onColorRemove ? () => onColorRemove(index) : undefined
              }
              handlePress={() => handlePress(index)}
              handlePressIn={handlePressIn}
              handlePressOut={handlePressOut}
              isPressed={isPressed}
            />
          ))
        ) : (
          <Text style={styles.placeholderText}>
            Here will be your palette...
          </Text>
        )}
      </View>
      {canLike && (
        <View style={styles.likeContainer}>
          <TouchableOpacity style={styles.icon} onPress={onLike}>
            <AntDesign
              name='like1'
              size={32}
              color={likedBy.includes(userId) ? 'green' : 'grey'}
            />
          </TouchableOpacity>
          <Text style={styles.likeLabel}>{likes}</Text>
        </View>
      )}
      {canEdit && (
        <View style={styles.iconsContainer}>
          <TouchableOpacity
            style={styles.icon}
            onPress={() =>
              navigation.jumpTo('Color Picker', {
                palleteInfo: { id, colors, name },
              })
            }
          >
            <Feather name='edit' size={32} color='green' />
          </TouchableOpacity>
          <TouchableOpacity style={styles.icon} onPress={onDeletePallete}>
            <Octicons name='diff-removed' size={32} color='red' />
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.icon}
            onPress={onConvertAndShareCSVPalette}
          >
            <Entypo name='share' size={32} color='#3C84A1' />
          </TouchableOpacity>
          <TouchableOpacity style={styles.icon} onPress={() => onCopy()}>
            <MaterialCommunityIcons
              name='content-copy'
              size={32}
              color='#767181'
            />
          </TouchableOpacity>
          <TouchableOpacity style={styles.icon} onPress={onPublish}>
            <MaterialIcons name='publish' size={32} color='#E57C23' />
          </TouchableOpacity>
          <TouchableOpacity style={styles.icon} onPress={onApplyTheme}>
            <Ionicons name='md-logo-flickr' size={32} color='#FF0060' />
```

```
            </TouchableOpacity>
          </View>
        )}
      </Flex>
      {canCreate && (
        <Flex direction='row' justify='space-between'>
          {onClear && (
            <Button
              style={styles.saveButton}
              variant='text'
              title='Clear Palette'
              uppercase={false}
              onPress={onClear}
            />
          )}
          <Button
            style={styles.saveButton}
            title='Save Palette'
            uppercase={false}
            disabled={!(colors.length > 0)}
            onPress={() => setConfirmation(true)}
          />
        </Flex>
      )}
      <Dialog visible={isConfirmation} onDismiss={() =>
setConfirmation(false)}>
        <DialogHeader title='Enter palette name' />
        <DialogContent>
          <Stack spacing={2}>
            <TextInput
              label='Poster-palette...'
              variant='outlined'
              value={palleteName}
              onChangeText={(text) => setPalleteName(text)}
            />
          </Stack>
        </DialogContent>
        <DialogActions>
          <Button
            title='Cancel'
            compact
            variant='text'
            onPress={() => setConfirmation(false)}
          />
          <Button
            title='Save'
            compact
            variant='text'
            disabled={!(palleteName.length > 0)}
            onPress={onSavePallete}
          />
        </DialogActions>
      </Dialog>
      {notification && (
        <Notification message={`Text Copied: ${notification}`} />
      )}
    </Flex>
  );
}

export default React.memo(ColorPalette);
```

Файл ColorPallete/styles.js

```
import { StyleSheet } from 'react-native';

export default StyleSheet.create({
  paletteContainer: {
    flexDirection: 'row',
    flexWrap: 'wrap',
    flexGrow: 1,
    justifyContent: 'space-between',
    // columnGap: 10,
    rowGap: 10,
    paddingVertical: 10,
    paddingHorizontal: 10,
    borderWidth: 1,
    borderRadius: 10,
    borderColor: '#3C84A1',
    minHeight: 94,
  },
  likeContainer: {
    flexDirection: 'row',
    justifyContent: 'center',
    alignItems: 'center',
    flexWrap: 'wrap',
    gap: 15,
    marginLeft: 20,
    width: 74,
    height: '100%',
  },
  likeLabel: {
    fontSize: 20,
  },
  iconsContainer: {
    flexDirection: 'row',
    justifySelf: 'flex-end',
    alignItems: 'flex-start',
    flexWrap: 'wrap',
    gap: 10,
    marginLeft: 20,
    width: 74,
  },
  icon: {
    width: 32,
    height: 32,
  },
  placeholderText: {
    fontSize: 16,
    color: '#2e2e2d',
    opacity: 0.7,
  },
  saveButton: {
    marginTop: 20,
    width: 160,
    alignSelf: 'center',
  },
});
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК З ПІДБОРУ КОЛЬОРОВИХ СХЕМ ДЛЯ ІНТЕРФЕЙСІВ

Програма та методика тестування

КПІ.ІТ-9222.045440.05.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр РЯБКО

Київ – 2023

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Об'єктом випробування є розроблений мобільний додаток для підбору кольорових схем для користувацьких інтерфейсів. Додаток було розроблено для платформ Android та iOS.

2. МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи мобільного додатку відповідно до визначених функціональних вимог;
- перевірка надійності збереження даних у сервісі Firebase;
- перевірка сумісності мобільного додатку з актуальними версіями операційних систем Android та iOS;
- виявленн я та документування будь-яких проблем, помилок або недоліків для їх подальшого виправлення;
- перевірка зручності та інтуїтивності користувацького інтерфейсу додатку.

3. МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- системне тестування – перевіряється усе програмне забезпечення в цілому;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;
- тестування «чорної скриньки» – об’єктом тестування тут є функції присутні у програмі. Перевіряється коректність вихідних даних при заданих вхідних;
- тестування «білої скриньки» – об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування мобільного застосунку з підбору кольорових схем для користувацьких інтерфейсів проводиться вручну. Метою тестування є виявлення можливих помилок і недоліків в роботі програмного забезпечення, перевірка зручності використання, а також аналіз сумісності застосунку з різними браузерами та операційними системами. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування, що передбачає перевірку відповідності реалізованих функцій заявленим у функціональних вимогах;
- статичне тестування коду, при якому проводиться аналіз якості коду та дотримання стандартів програмування;
- тестування сумісності веб-версії застосунку з різними браузерами, а також на різних версіях окремих браузерів;
- перевірка роботи консольної версії застосунку на різних операційних системах;
- тестування інтерфейсу користувача з метою виявлення можливих проблем у взаємодії користувача і програми;
- тестування зручності використання, що передбачає аналіз зручності роботи з програмою для кінцевого користувача.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК З ПІДБОРУ КОЛЬОРОВИХ СХЕМ ДЛЯ ІНТЕРФЕЙСІВ

Керівництво користувача

КПІ.ІТ-9222.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр РЯБКО

Київ – 2023

ЗМІСТ

1. ПРИЗНАЧЕННЯ ПРОГРАМИ

"Color Scheme Selector" - це мобільний застосунок для підбору та управління кольоровими схемами для користувацьких інтерфейсів. Цей застосунок призначений для дизайнерів, програмістів, а також для будь-якого користувача, який хоче експериментувати з кольором в своїх проектах.

Основна версія додатка включає підтримку як Android, так і iOS платформ. Інсталяційна версія додатка містить всі необхідні компоненти та бібліотеки для забезпечення його повноцінного функціонування.

Репозиторій додатка містить весь вихідний код, а також документацію, скрипти для збірки та встановлення та інші ресурси, необхідні для розробки, тестування та випуску нових версій додатка.

2. ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1. Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

Мінімальні вимоги:

– тип пристрою: Смартфон або планшет;
– операційна система: Android 7.0+ або iOS 11.0+;
– об'єм оперативної пам'яті: 2 Гб;
– внутрішня пам'ять: 100 Мб вільного місця для встановлення та оновлення додатка;
– підключення до мережі Інтернет: 3G, 4G, Wi-Fi.

Рекомендовані вимоги:

– тип пристрою: Смартфон або планшет;
– операційна система: Android 10.0+ або iOS 13.0+;
– об'єм оперативної пам'яті: 4 Гб;
– внутрішня пам'ять: 500 Мб вільного місця для встановлення та зберігання даних додатка;
– підключення до мережі Інтернет: 4G, 5G, Wi-Fi.

2.2. Завантаження застосунку

На даний момент, для встановлення застосунку "Color Scheme Selector", користувачу необхідно здійснити ручне встановлення через відповідний APK-файл. Процедура встановлення включає наступні кроки:

– потрібно завантажити APK-файл на мобільний пристрій за допомогою наступного посилання: https://apkfab.com/diplomapalleteapp/com.olimpki.diplomaplalleteapp/apk?h=1c39668cc7917000b573cf3aa2bffd001fcfafbaa558c0f936fd0536963fa35c;
– після завантаження APK-файлу на пристрій, використовуйте будь-який доступний інсталятор для встановлення застосунку на ваш пристрій.

Також треба звернути увагу, що в залежності від налаштувань безпеки пристрою, може знадобитися дозволити встановлення додатків з невідомих джерел.

Отримати доступ до веб-додатку можна перейшовши за посиланням https://color-pallete-app-e1a21.web.app.

2.3. Перевірка коректної роботи

Для перевірки коректності процесу встановлення та розгортання програмного забезпечення "Color Scheme Selector", користувач повинен виконати наступні кроки:

– завантажити додаток з відповідного магазину додатків (Google Play для Android, App Store для iOS);

– виконати процедуру встановлення за інструкціями магазину додатків;

– запустити додаток, натиснувши на відповідний значок на екрані свого пристрою;

– переконатись, що головний екран додатка відображається коректно, без помилок або збоїв;

– перевірити, чи можна створити нову кольорову схему і зберегти її;

– перевірити, чи з'являється збережена схема в розділі “Palettes”;

– закрити додаток і знову його відкрити, перевірити, чи збереглися всі збережені схеми.

Якщо всі вказані кроки виконані успішно без помилок, то можна вважати, що процес встановлення та розгортання програмного забезпечення пройшов успішно.

3. ВИКОНАННЯ ПРОГРАМИ

Розпочавши свою роботу з додатком, користувач спочатку має зареєструватися (рис. 3.1), ввівши свою електронну адресу та створивши пароль. Однак, якщо він вже зареєстрований, потрібно авторизуватися (рис. 3.2), використовуючи свою електронну пошту та пароль:

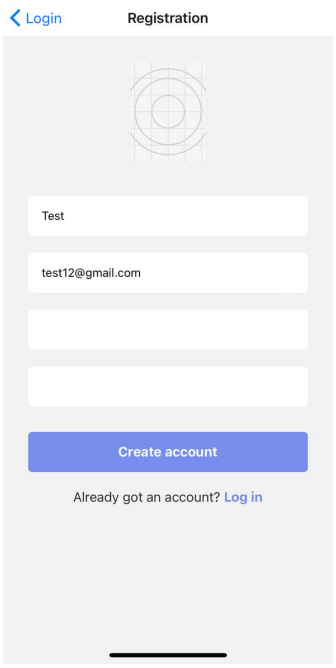


Рис. 3.1 – Регістрація

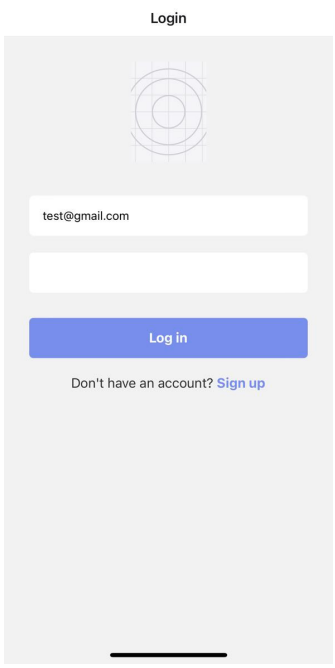


Рис. 3.2 - Авторизація

Після входу до системи, користувач має доступ до основного інтерфейсу програми, де він може створювати нові кольорові схеми. Це включає вибір окремих кольорів за допомогою колеса кольорів або введення коду кольору вручну (рис. 3.3). Після вибору кольорів, користувач може їх видаляти, затримавши палець на кольорі впродовж 2 секунд (рис. 3.4), або редагувати (рис. 3.5):

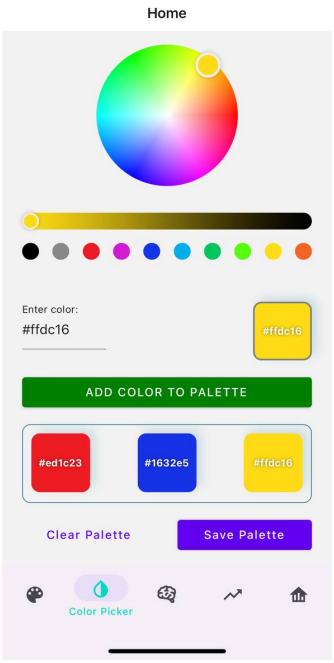


Рис. 3.3 – Введення кольору вручну

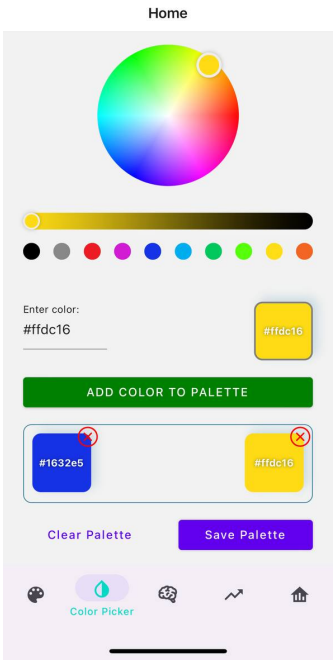


Рис. 3.4 - Видалення кольору

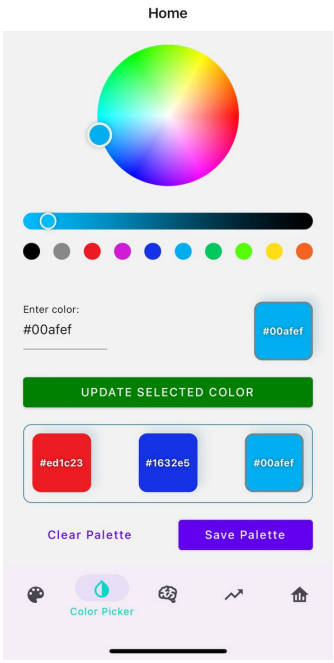


Рис. 3.5 - Редагування кольору

Після створення, користувач може зберегти палітру для подальшого використання (рис. 3.6). Ці збережені палітри відображаються на екрані збережених палітр, де користувач може їх шукати (рис. 3.7), редагувати, ділитися з іншими користувачами (рис. 3.8), а також застосувати як тему для інтерфейсу програми (рис. 3.9):

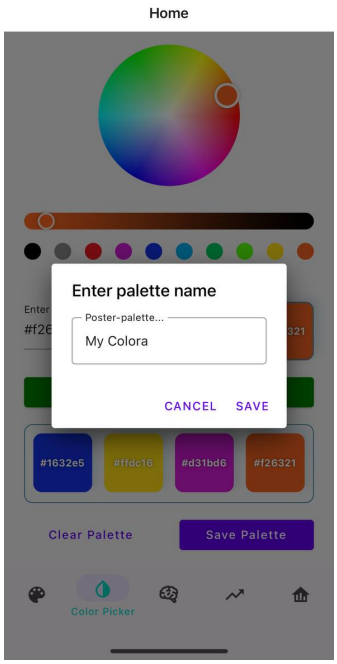


Рис. 3.6 - Збереження палітри

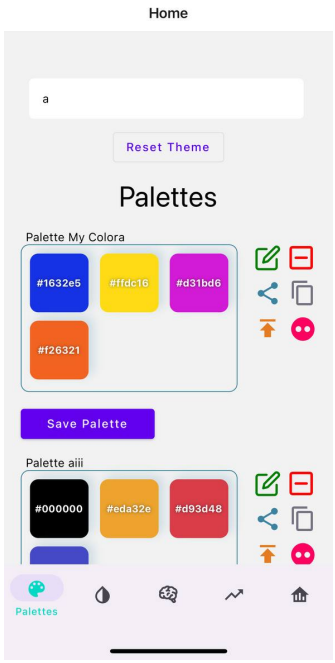


Рис. 3.7 – Пошук палітр за ім’ям

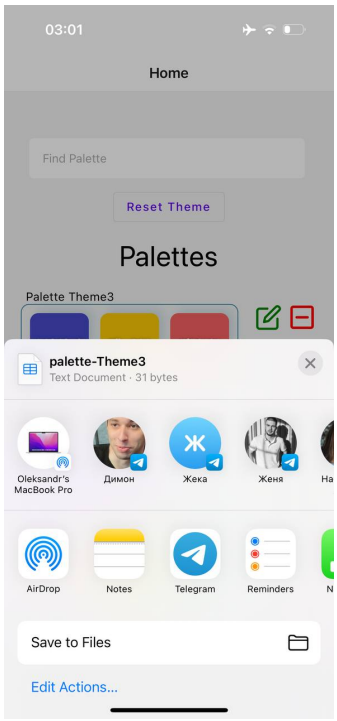


Рис. 3.8 - Поділитися палітрою або її завантажити

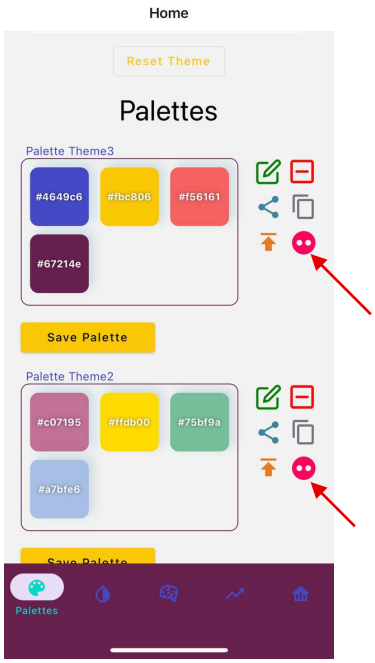


Рис. 3.9 - Змінити кольорову палітру для інтерфейсу

Програма також надає можливість копіювати окремі кольори (натиснути на будь-який колір) або цілу палітру (рис. 3.10), яка може бути корисною при створенні нових схем. Користувач також може публікувати свою палітру у розділ трендів, де інші користувачі можуть її переглянути та оцінити (3.11):

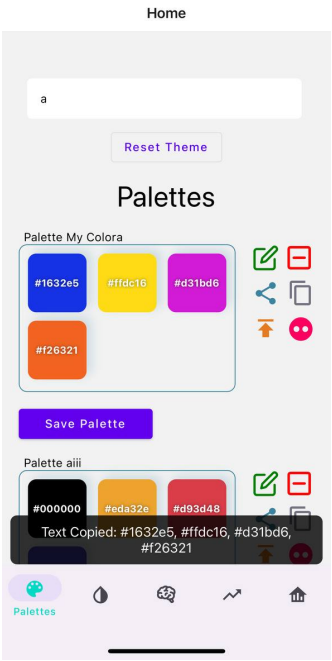


Рис. 3.10 – Скопіювати всю палітру

Рис. 3.11 – Опублікувати палітру до трендів

На екрані редагування, користувач може обирати колір з колеса кольорів або вводити його вручну. Також доступна функція очищення палітри (рис. 3.12), видалення окремих кольорів з палітри або заміна окремих кольорів у палітрі. Після редагування, змінену палітру можна зберегти (див. рис. 3.3, 3.4, 3.5):

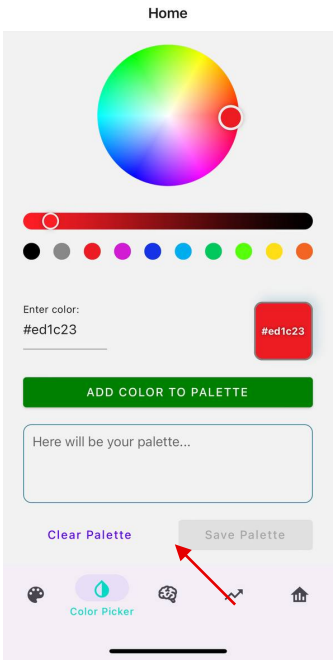


Рис. 3.12 – Очистити палітру

Окрім того, програма надає можливість генерації нових палітр з заданим основним кольором та заданою кількістю кольорів в палітрі (рис. 3.13). Згенеровані палітри можуть бути збережені для подальшого використання (рис. 3.14):

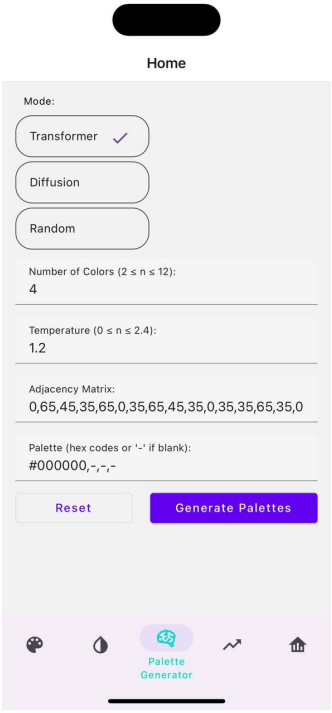


Рис. 3.13 - Введення відповідних параметрів для генерації палітр

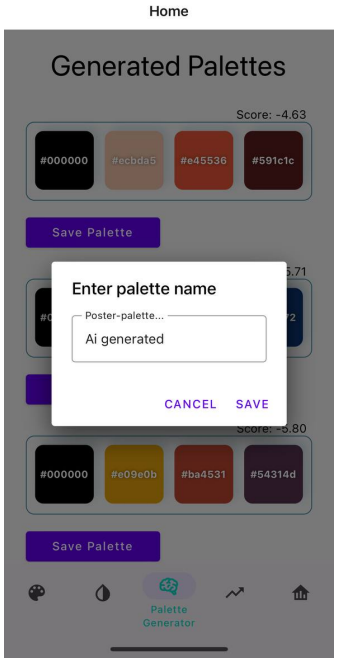


Рис. 3.14 - Збереження згенерованої палітри

На екрані трендів користувач може переглядати палітри інших користувачів, вподобати їх (рис. 3.15) або зберегти для подальшого використання (рис. 3.16):

Рис. 3.15 – Перегляд та вподобання трендів серед палітр

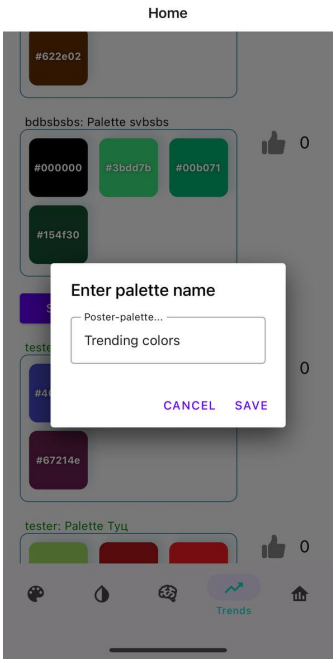


Рис. 3.16 – Збереження палітри з екрану трендів

Щодо аналізу кольорів, програма дозволяє вводити основний колір та колір, контрастність до якого, користувач бажає порівняти (рис. 3.17). Результати аналізу можуть бути збережені як нова палітра (рис. 3.18):

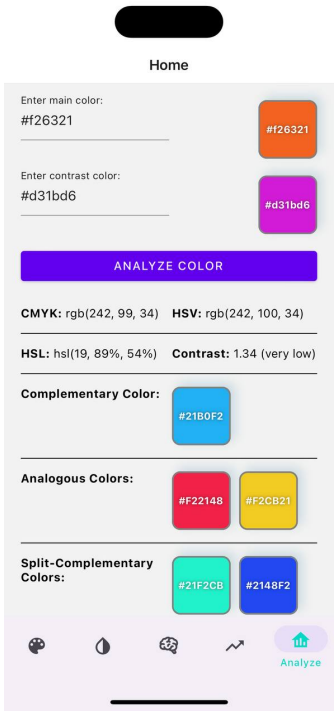


Рис. 3.17 – Результат аналізу кольору

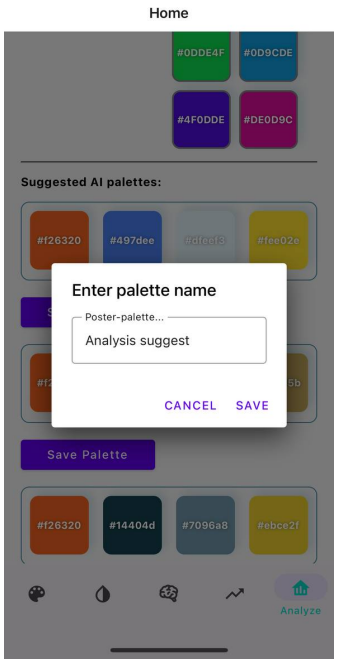


Рис. 3.18 - Згенеровані за допомогою ШІ палітри для заданого кольору

І, нарешті, якщо користувач бажає видалити палітру, він може вибрати її з історії створених палітр і натиснути кнопку “Видалити" (рис. 3.19). Після цього палітра буде видалена.

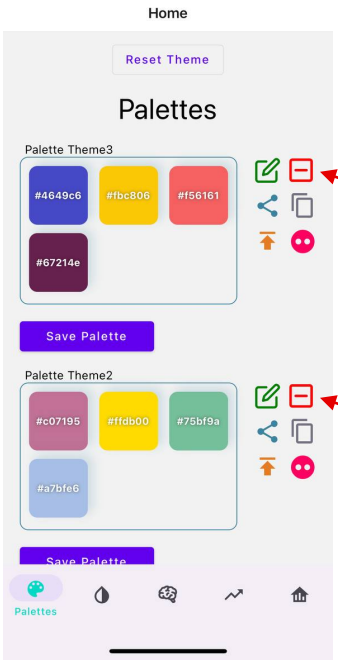


Рис. 3.19 – Кнопка видалення палітри

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ЗАСТОСУНОК З ПІДБОРУ КОЛЬОРОВИХ СХЕМ ДЛЯ ІНТЕРФЕЙСІВ

Графічний матеріал

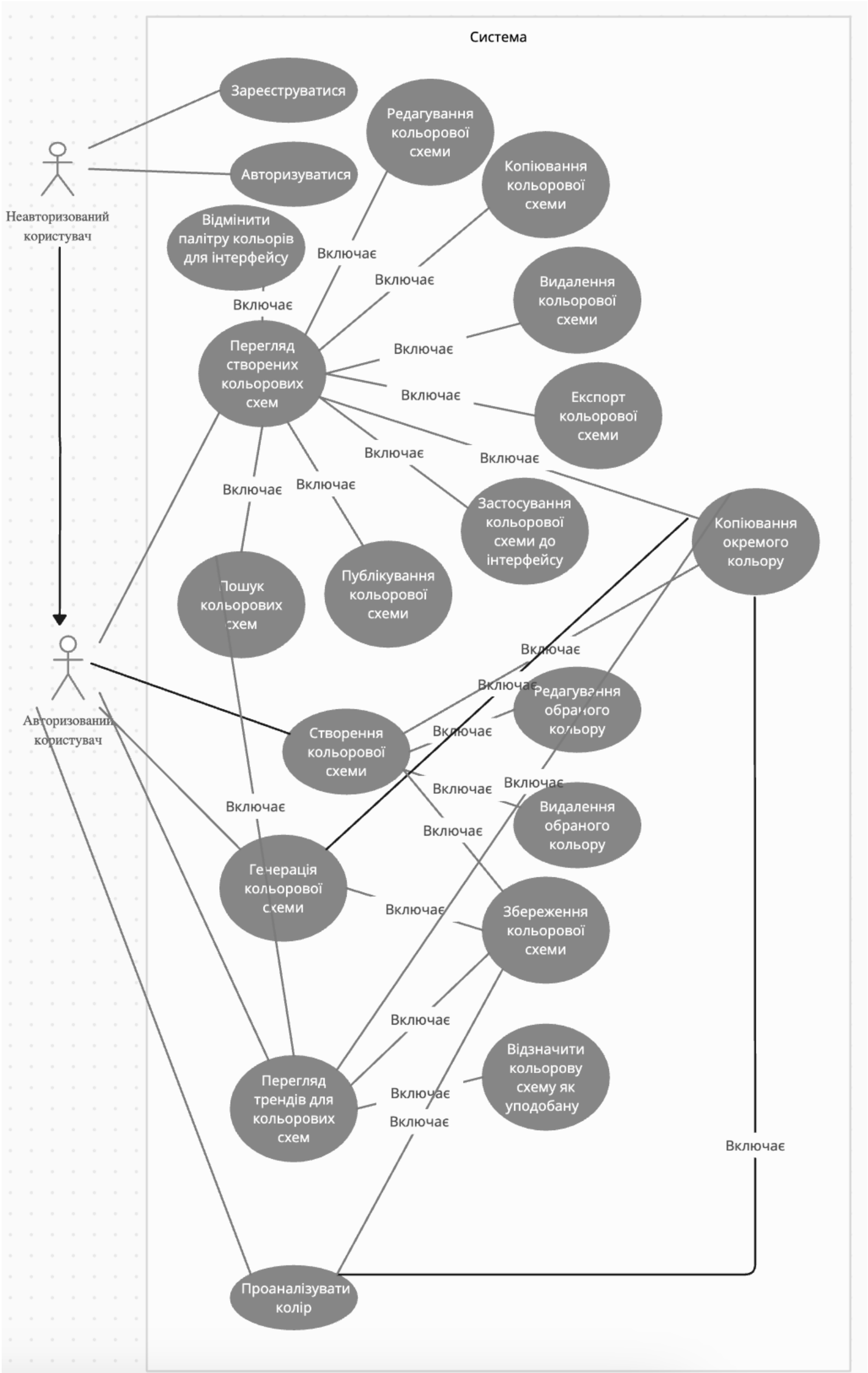
ПІ.ІТ-9222.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Ірина ВІТКОВСЬКА

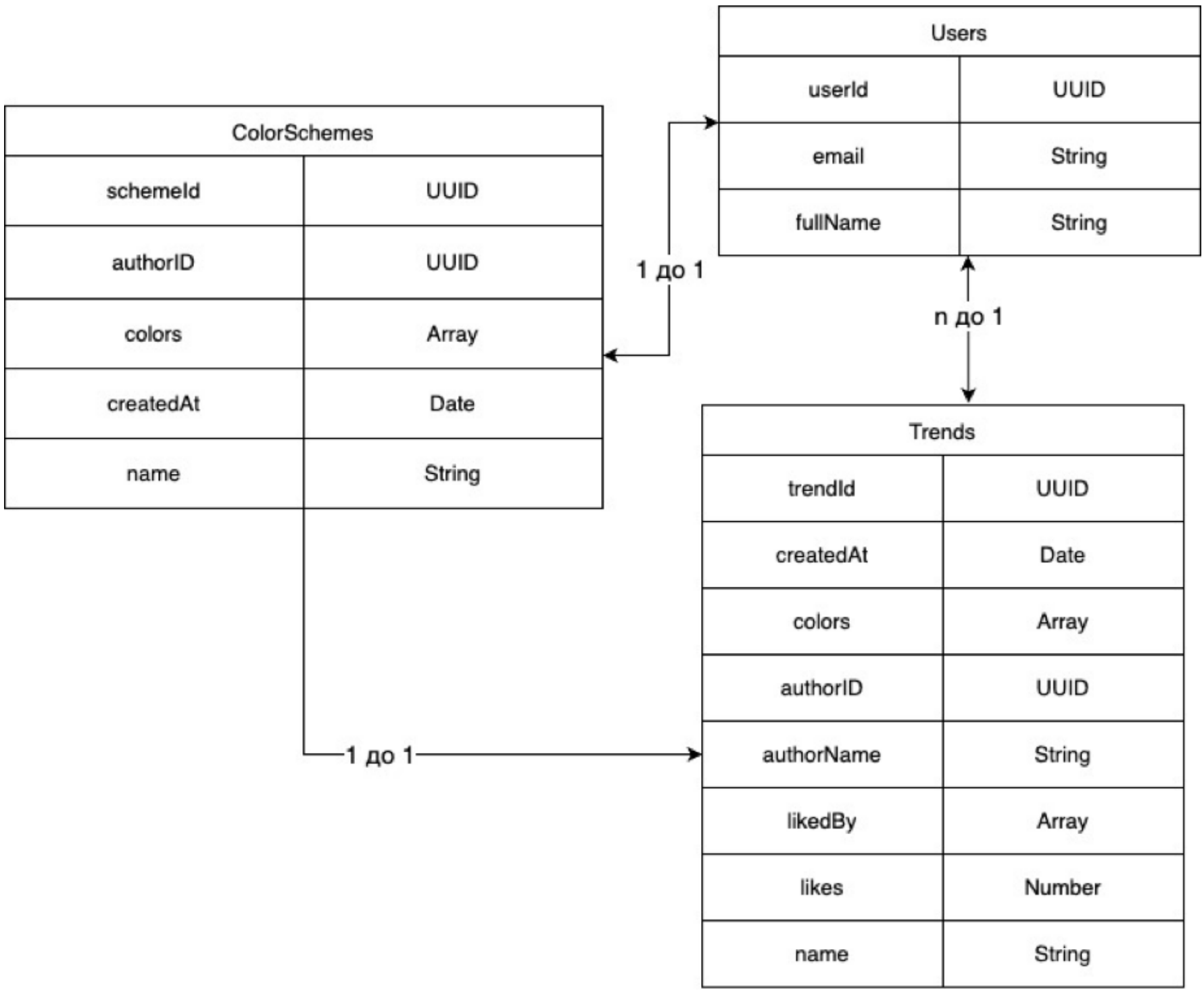
Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Олександр РЯБКО

Київ – 2023



					КПІ.ІТ-9222.045440.06.99.ССВ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Рябко О.В.			Схема структурна варіантів використань	Лит.	Арк. / Аркушів
Перев.		Вітковська І.І.					1
Т. Кон.						Аркуш	Аркушів
					Застосунок з підбору кольорових схем для інтерфейсів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92	
Н. Кон.		Вітковська І.І.					
Затв.		Вітковська І.І.					



					КПІ.ІТ-9222.045440.06.99.СБД	
					Схема бази даних	Лит. / Арк. / Аркушів: 1
Зм.	Арк.	№ докум.	Підп.	Дата		Аркуш / Аркушів
Розроб.		Рябко О.В.			Застосунок з підбору кольорових схем для інтерфейсів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Перев.		Вітковська І.І.				
Т. Кон.						
Н. Кон.		Вітковська І.І.				
Затв.		Вітковська І.І.				

Home

Find Palette

Reset Theme

Palettes

Palette sugg

#eda32d #403043 #c07195 #e0e5e9

Save Palette

Palette aiii

#000000 #eda32e #d93d48 #4649c6

Palettes

Home

Enter color:
#ffdc16

#ffdc16

ADD COLOR TO PALETTE

#ed1c23 #1632e5 #ffdc16

Clear Palette Save Palette

Color Picker

Login Registration

Full Name

E-mail

Password

Confirm Password

Create account

Already got an account? Log in

Home

Enter main color:
#0038ad

#0038ad

Enter contrast color:
#FFFFFF

#FFFFFF

ANALYZE COLOR

CMYK: rgb(0, 56, 173) **HSV:** rgb(0, 55, 173)

HSL: hsl(221, 100%, 34%) **Contrast:** 9.68 (high)

Complementary Color: #AD7500

Analogous Colors: #008FAD #1E00AD

Split-Complementary Colors: #AD1E00 #8FAD00

Analyze

Login

E-mail

Password

Log in

Don't have an account? Sign up

Home

Enter color:
#ffdc16

#ffdc16

ADD COLOR TO PALETTE

#1632e5 #ffdc16

Clear Palette Save Palette

Color Picker

Home

Palette (hex codes or '-' if blank):
#000000,-,-,-

Reset Generate Palettes

Generated Palettes

Score: -4.71

#000000 #f29d1b #d64f37 #5e021a

Save Palette

Score: -4.72

#000000 #c5c4c3 #00a3e0 #0038ad

Save Palette

Score: -4.94

Palette Generator

Home

Find Palette

Palettes

tester: Palette Iii

#ef4f0f #1632e5 #00afef #00c65c

2

tester: Palette Zz

#000000 #f7b339 #a17530 #622e02

2

Trends

					КПІ.ІТ-9222.045440.06.99.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Рябко О.В.			Креслення вигляду екранних форм	Літера / Маса / Масштаб
Перевірив		Вітковська І.І.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.І.			Застосунок з підбору кольорових схем для інтерфейсів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-92
Затвердив		Вітковська І.І.				