Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва факультету)

Кафедра комп'ютерних та інформаційних технологій і систем

(повна назва кафедри)

Пояснювальна записка до дипломного проекту (роботи)

бакалавра
(освітньо-кваліфікаційний рівень)

на тему

Розробка мобільного додатку для створення Instagram-історій та фотопрезентацій

Виконав: студент 4 курсу, групи 401-ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Гусак Віктор Олександрович
(прізвище та ініціали)

Керівник Ляхов О.Л.
(прізвище та ініціали)

Рецензент Смірнов А.Л.
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розробка мобільного додатку для створення Instagram-історій та фотопрезентацій»

Студента групи 401-ТН Гусака Віктора Олександровича

Керівник роботи
доктор технічних наук,
професор Ляхов О.Л.

Завідувач кафедри
кандидат технічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Пояснювальна записка містить: 57 с., 26 рисунків, 1 додаток, 10 джерел.

Об’єкт дослідження – процес розроблення мобільного додатку для створення Instagram-історій та фотопрезентацій.

Мета кваліфікаційної роботи – розроблення, тестування та введення в експлуатацію мобільного додатку для створення Instagram-історій та фотопрезентацій.

Ключові слова: мобільний додаток, фреймворк, React, React-Native, JavaScript, база даних.

ABSTRACT

Bachelor's qualification work: 57 pages, 26 drawings, 1 appendix, 10 sources.

Object of research – the process of developing a mobile application for creating Instagram stories and photo presentations.

The purpose of the qualification work – development, testing and commissioning of a mobile application for creating Instagram-stories and photo presentations.

Keywords: mobile application, framework, React, React-Native, JavaScript, database.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTP – це протокол для передачі гіпертекстових документів, таких як HTML. Він створений для зв'язку між веб-браузерами і веб-серверами

БД – база даних

ОС – операційна система

JavaScript (JS) – мова програмування, що є прототипна-орієнтованим. Він відображає мову ECMAScript, чиїм прототипом спочатку і був. Перша варіація з'явилася ще в 1995 році і з тих пір постійно вдосконалювалася, поки не прийшла до нинішнього вигляду

Vue – це веб-фреймворк для розробки призначених для користувача інтерфейсів на мові програмування JavaScript. Vue створений для поступового впровадження в існуючу програму. Він вирішує різні завдання рівня уявлення (view), спрощує роботу з іншими бібліотеками та дозволяє створювати складні односторінкові додатки (SPA, Single-Page Applications)

Laravel – це фреймворк для web-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних наболілих завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel - це спроба об'єднати все найкраще, що є в інших PHP фреймворк

React – це бібліотека Javascript з відкритим вихідним кодом для побудови компонентів інтерфейсу, які можна перевикористати.

React Native - це багатоплатформовий фреймворк з відкритим вихідним кодом для розробки нативних мобільних і настільних додатків на JavaScript і TypeScript, створений Facebook, Inc. React Native підтримує такі платформи як Android, Android TV, iOS, macOS, Apple tvOS, Web, Windows і UWP, дозволяючи розробникам використовувати можливості бібліотеки React поза браузера для створення нативних додатків, що мають повний доступ до системних API платформ

Хостинг – це послуга з надання обчислювальних потужностей для

розміщення інформації на сервері, що постійно знаходиться в мережі (зазвичай Інтернет). Хостингом також називається послуга з розміщення обладнання клієнта на території провайдера із забезпеченням підключення його до каналів зв'язку з високою пропускною здатністю

HTML (англ. Hyper Text Markup Language – мова розмітки гіпертексту) – стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді

Веб-ресурс – це сукупність електронних документів, що знаходяться у веб-просторі мережі.

Інтерфейс користувача (ІК) – сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача

ВСТУП

Сьогодні мобільний і швидкий доступ до інформації використовують все більше людей. З кожним роком відсоток користувачів мобільних версій сайтів зростає, а, комп'ютерні версії стають все менш популярними. Безліч програмних продуктів в еру цифрових технологій, розробляється для мобільних телефонів.

Наразі у світі практично не залишилося людей, які б не користувалися мобільним пристроєм. Причинами їх популярності можна назвати: функціонал, доступ до всіх способів зв'язку і, звичайно ж, невеликий розмір пристрою. Потрібно зауважити, що мобільні пристрої практично замінили настільний комп'ютер в будинку.

Однак існують і недоліки, оскільки у зв'язку зі збільшенням мобільної техніки у населення з'являються деякі труднощі, які полягають в відстеження всіх потоків інформації на всіх пристроях. На сьогодні, основну класифікацію мобільних телефонів ділять на: BlackBerry, Symbian OS, Windows Mobile, Android, MacOS і інші. ОС Android є «однією з найпростіших і одночасно комплексних платформ, вся система завантажується за один раз ».

Мобільні телефони вже перестали бути чимось незвичайним і добре справляються зі своєю функцією – є засобом комунікації між людьми. При цьому, недавно з'явилися, але вже міцно ввійшли в наше життя смартфони настільки функціональні, що важко сказати, чого вони не вміють: це і плеєр, і фотоапарат, і можливість використання Інтернет-ресурсів та інше. По суті, всі смартфони стали невеликий копією комп'ютера, який зручно мати при собі.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис предметної області

Мобільний додаток для створення Instagram-історій та фото презентацій допомагає користувачам створювати власні Stories та наповнити Instagram сторінку постами різних форматів.

Для того, щоб розібратися, що таке Stories в Instagram, необхідно уявити, розповідь друзям історії в реальному житті. Для цього знадобилося б використання фото, або коротке відео. Все це можна зробити в Інтернеті, не виходячи з дому і охоплюючи велику аудиторію, розташовану в різних регіонах. Додаток Instagram – це можливість бути ближче до друзів, перебуваючи в будь-якій точці світу [11].

Безкоштовна функція дозволяє ділитися не тільки з друзями, але і з широкою аудиторією користувачів Instagram.

«Instagram-історії – це телебачення XXI століття», – вважає медіаексперт Франак Вячорка. Вже зараз використання історій серед користувачів зростає в 15 разів інтенсивніше, ніж інші типи контенту.

Головною перевагою Instagram Stories є те, що це потужний інструмент, який служить для залучення потенційних клієнтів до бізнесу. За перші шість місяців свого існування функція розваг стала інструментом маркетингу.

Понад 300 мільйонів людей щодня спостерігають за Stories, і кожні 5-6 історій отримують відгуки від аудиторії. Відомі бренди Adidas, Toyota, Gucci, а також зірки шоу-бізнесу вже давно звертали увагу на функцію [11].

Окрім засобів просування бізнесу в соціальних мережах, завдяки функції створення, розміщення та управління вмістом, є можливість створювати історії щодня і не переживати, що вони заповнять головну сторінку інформацією.

Завдяки Instagram є можливість ділитися з друзями та іншими

користувачами цікавими моментами з життя, історіями, особистими думками, опублікувати статтю, фото чи відео.

Ця функція дозволяє додавати геотеги, хештеги, посилання на особистий блог або профіль у соціальних мережах, а також посилання з рекламою чи продажем інформації. Все це можна доповнити наклейками та смайликами, щоб привернути увагу. Хештеги можуть повідомляти додаткову інформацію про користувача чи місцезнаходження:

- розташування;
- погода;
- час зйомки.

Ця подача інформації виглядає більш привабливою. Хештеги можуть бути накладені на фото та відео, адаптовані до рухомих об'єктів. Фотографії та відео відображаються переважно у вертикальному положенні, нещодавно додана можливість відображати горизонтальні фотографії та відео, але ця функція не набула широкого поширення, оскільки більшість користувачів переглядають вміст на своєму смартфоні вертикально.

Стандартне відео може бути замінено «циклом», який називається бумерангом, і в цьому випадку кілька кадрів будуть повторюватися безперервно, спочатку в стандартному, а потім у зворотному порядку.

Історії в Instagram – це повноцінний інструмент для роботи з матеріалами не тільки в особистих цілях, але і в комерційних. Правила роботи з історіями загалом такі самі, як і при публікації історій – їх потрібно планувати, а також підтримувати в одному стилі та відповідно до тенденцій [11].

Публікуючи яскраві, красиві та цікаві історії є можливість залучити більше користувачів на свою сторінку. Увага аудиторії забезпечує найкраще місце при ранжуванні облікового запису не тільки в стрічці користувачів, але і в загальному рейтингу соціальної мережі. Більш вигідна позиція збільшує популярність особистості або власного бренду. Stories чудово підходить для

реклами.

Додаток – сучасний онлайн редактор для створення унікальних історій для Instagram та інших соціальних мереж. Наявна можливість використовувати готові шаблони або створювати свої, отримувати колажі фотографій в красивій обгортці з текстом і в потрібному розмірі.

Основні можливості:

- створення історій і постів;
- створення карусельних панорам;
- додавання фото в встановлені місця;
- додавання і редагування тексту;
- обкладинки, рамки, фони і багато іншого.

Головною метою розробки додатку є надання користувачу можливості для оформлення та майбутнє наповнення сторінки в Instagram, це може бути власний аккаунт, або комерційний. Додаток має містити всі необхідні інструменти для створення Stories і постів, які наповнюють сторінку Instagram та допоможуть залученню нової аудиторії, що є важливою метою для користувачів і бізнес-акаунтів.

1.2 Огляд існуючих програмних рішень

Розробка якісного та конкурентоспроможного продукту без аналізу ринку існуючих додатків неможлива. Тому аналіз програм для створення історій в Instagram та фотопрезентацій є важливим кроком. Основною метою аналізу ринку додатків є збір статистики користувачів та виявлення недоліків існуючих мобільних додатків.

На ринку доступна програма Unfold. Це безкоштовна програма для створення Історій. Доступна для смартфонів Android та iOS.

Ця програма для редагування фотографій дає 25 безкоштовних шаблонів для сторінок у мінімалістичному стилі (рис. 1.1). Шаблони – це макети з

.контейнерами для зображень або тексту. Кількість преміум-шаблонів (платних) – 90.

Рисунок 1.1 – Сторінка редактора в додатку «Unfold»

Фотографії для Stories можна завантажити з галереї (рис. 1.2).

Рисунок 1.2 – Галерея додатку «Unfold»

За допомогою макетів можна робити стильні колажі (рис. 1.3).

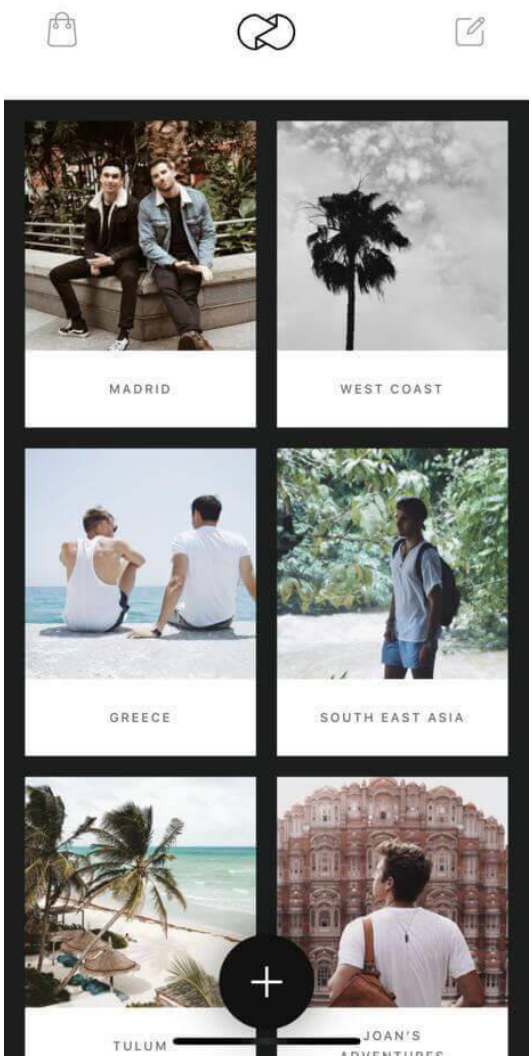

Рисунок 1.3 Коложі додатку «Unfold»

Можливості додатку:

- змінити колір тла (7 кольорів);
- додавання та редагування тексту;
- налаштувати колір тексту;
- міжрядковий інтервал та інтервал між літерами;
- зробити текст жирним або курсивом та додати кольоровий фон;
- 6 шрифтів для тексту доступні безкоштовно.

Тільки три працюють з кирилицею. Також є платний набір із 5 шрифтів,

налаштування представлені на рис. 1.4.

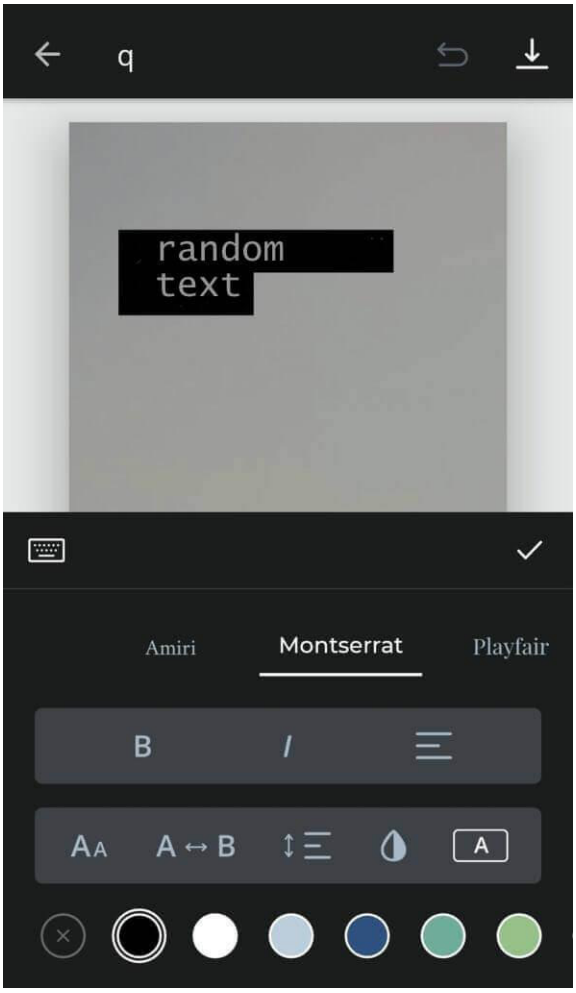

Рисунок 1.4 – Налаштування в додатку «Unfold»

Є можливість придбати додаткові набори макетів та шрифтів у додатку. Вартість платних наборів – від 26 до 60 гривень.

В додатку для створення окремих історій, можна створити послідовність Stories, додавши нові екрани. Зручно підготувати до публікації цілу серію Stories відразу. Наявна можливість переглянути екрани, переконатися, що серіали зроблені в одному стилі, і оцінити, як будуть виглядати сторінки в певному порядку. Можливо змінити порядок екранів і видалити непотрібні.

Готову сторінку можна зберегти в галереї або завантажити в Instagram безпосередньо з програми.

Over - це графічний редактор соціальних мереж, який також можливо

використовувати для створення Stories.

Створюючи полотно в додатку, наявна можливість вибрати пропорції з заданих параметрів або вручну налаштувати розмір зображення в пікселях. Оптимальний розмір для Stories – 1080x1920 пікселів (рис. 1.5).

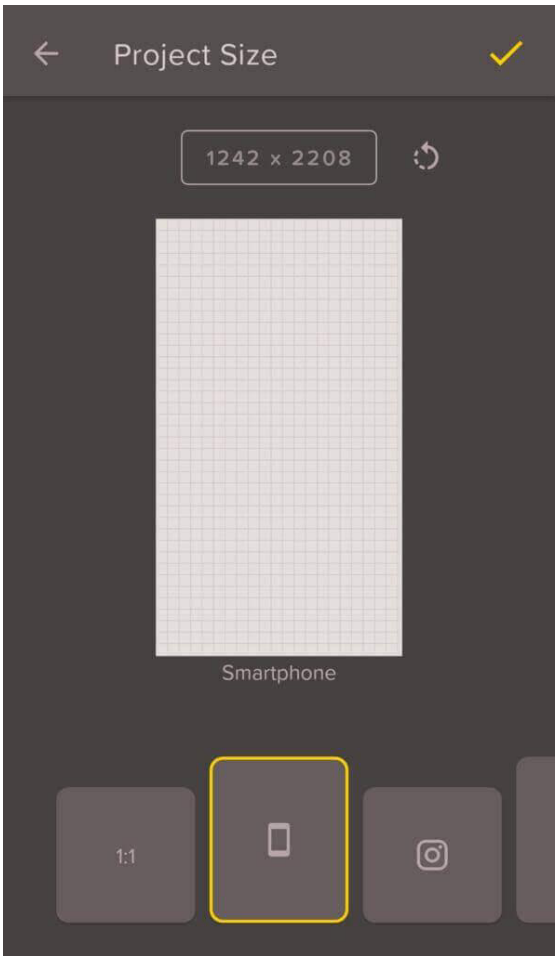

Рисунок 1.5 – Stories в додатку «Unfold»

Існують засоби корекції кольорів (рис. 1.6):

- регулювання експозиції контраст,
- насиченість,
- прозорість зображення,
- додавати тіні.

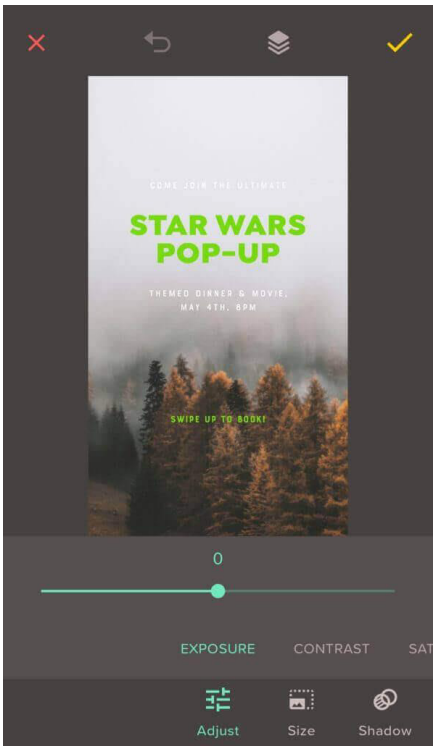

Рисунок 1.6 – Налаштування в додатку «Unfold»

Фотографії можна завантажити з галереї або скористатися вбудованим пошуком на Unsplash (рис. 1.7).

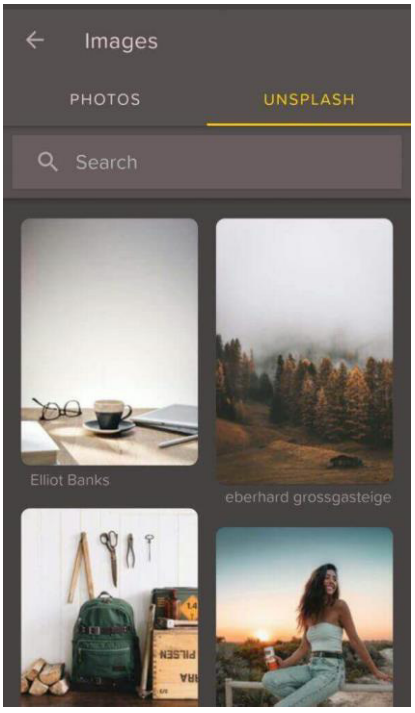

Рисунок 1.7 – Сторінка Unsplash в додатку «Unfold»

Додаток має безліч шрифтів (більшість з них платні). Ще один недолік – майже всі шрифти працюють лише з латиницею.

Можливості додатку (рис. 1.8):

- налаштувати міжрядковий інтервал та міжрядковий інтервал для тексту;
- налаштувати кольору тла;
- загрузку графікі - геометричні фігури, написи, піктограми та іншу графіку;
- зміна їх кольору і розміру.

Рисунок 1.8 – Сторінка графіки в додатку «Unfold»

Insta Story Art – додаток для створення Stories. Традиційно для таких послуг недостатньо безкоштовного контенту. Однак, на відміну від інших програм, усі фони та шрифти доступні у безкоштовній версії, їх багато і вони якісні. Але шрифти мають проблеми з кирилицею, і додаток не працює зі стікерами, вони зроблені через спеціальний редактор, в якому можна зробити прозорий фон. Останній недолік – це неможливість скасувати дії.

Додаток має розширену функціональність та шаблони, які легко

налаштувати.

Всі фотографії на шаблоні можуть бути:

- обрізані за бажанням;
- обернуті;
- зі зміненою пропорцію.

Якщо не подобаються встановлені фони, доступна можливість завантажити своїх власних.

В інших додатках на вибір лише кілька кольорів, тут можливо вибрати будь-який і навіть зареєструвати його код за допомогою #.

Розширене редагування тексту:

- курсив;
- жирний текст;
- зміна ширини вікна.

Шаблони розділені за завданнями:

- мода;
- їжа;
- знижки;
- інтер’єри;
- фітнес;
- нова колекція та класика;
- без певної теми.

Ця програма добре підходить для брендів, наприклад, для Інтернет-магазинів розробники створили анімовані шаблони для продажів та колекцій.

Insta Story Art має безліч цікавих фонів, і всі вони безкоштовні.

Додаток має редактор підсвічування – це збережена колекція історій, яку можна побачити над основними публікаціями акаунта. Існує можливість зробити обкладинку смайлів, піктограм та спеціальних ілюстрацій або об’єднати їх в одне зображення (рис. 1.9).

Рисунок 1.9 – Сторінка редактора в додатку «Insta Story»

1.3 Постановка задачі

Завданням кваліфікаційної роботи є розробка структури мобільного додатку, реалізація цієї структури на практиці за допомогою JavaScript, React, React Native, PHP, MySQL, а також тестування цього додатку і його введення у експлуатацію.

Мобільний додаток для створення Instagram-історій та фото презентацій має складатися з таких функціональних блоків:

1. Головна сторінка
    1.1. Сітка готових шаблонів
    1.2. Фільтрація по тегам
    1.3. Фільтрація по назві
2. Сторінка оформлення підписки для розблокування платних шаблонів

2.1. Відновлення раніше придбаних підписок

2.2. Ознайомлення з політикою конфендіальності та умовами використання

3. Редактор готових шаблонів

3.1. Область розміщення макету для редагування

3.1.1. Можливість заповнювати медіа

3.1.2. Можливість редагувати розміщення, поворот, зум медіа

3.1.3. Можливість редагувати текст, колір текст, шрифт, жирність, горизонтальне вирівнювання

4. Редактор готових шаблонів

4.1. Збереження в історію

4.2. Збереження в галерею фотографій

5. Збережені шаблони

Мобільній додаток повинен забезпечувати такі функції:

- надавати користувачам всю необхідну інформацію про шаблони;
- пошук необхідних шаблонів;
- можливість редагувати шаблони;
- можливість зберегти відредагований шаблон.

Для того, щоб визначити функціонал модулів розглянемо функції, які можуть виконувати користувачі та контент-менеджер.

Таблиця 1.1 – Функції адміністратора в адміністративному модулі

Функція	**Виконавець**
РЕДАГУВАННЯ. Створення та редагування макетів	Дизайнер
РЕДАГУВАННЯ. Опис шаблону, добавлення інформації для відображення в додатку	Контент-менеджер

Таблиця 1.2 – Функції користувачів мобільного додатку

Функція	**Виконавець**
Тестування. Якісний аналіз створених шаблонів	Контент-менеджер
ПЕРЕГЛЯД. Ознайомлення з існуючими шаблонами	Користувач
РЕДАГУВАННЯ. Завантаження зображення	Користувач
РЕДАГУВАННЯ. Зміна тексту в шаблоні	Користувач
РЕДАГУВАННЯ. Добавлення нового тексту в макет	Користувач
ЗБЕРЕЖЕННЯ. Збереження готового макету для майбутнього доопрацювання	Користувач
ЗБЕРЕЖЕННЯ. Збереження готового макету в форматі фотографії в галерею	Користувач

РОЗДІЛ 2
ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ СТВОРЕННЯ INSTAGRAM-ІСТОРІЙ ТА ФОТОПРЕЗЕНТАЦІЙ

2.1 Функціонал та структура мобільного додатку

Враховуючи потребу, в мобільному додатку для покращення продукту, а саме можливість заповнення та редагування створених шаблонів без оновлення програми з App Store для користувачів iOS та Play Market для користувачів Android, може бути важко. Через проблеми, що виникають, проект, що розробляється, буде розділений на ряд підпроектів для кожного типу користувачів: веб-додаток для дизайнерів, веб-додаток для менеджера вмісту, мобільний додаток для користувачів. Цей підхід скорочує час на створення нових шаблонів та додавання їх до мобільного додатку для використання користувачем без оновлення нової версії програми.

REST – це архітектурний стиль взаємодії компонентів розподіленої системи в комп'ютерній мережі. REST визначає стиль взаємодії (обміну даними) між різними компонентами системи, кожна з яких може бути фізично розташована в різних місцях.

Цей архітектурний стиль є узгодженим набором обмежень, які враховуються при проектуванні розподіленої системи. Ці обмеження іноді називають принципами REST.

REST визначає, як компоненти розподіленої системи повинні взаємодіяти між собою. У загальному випадку це робиться за допомогою запитів та відповідей. Компонент, що надсилає запит, називається клієнтом. Компонент, який обробляє запит і надсилає відповідь клієнту, називається сервером. Запити та відповіді найчастіше надсилаються через HTTP (HyperText Transfer Protocol).

Як правило, сервер – це веб-додаток. Клієнтів може бути не один, а досить багато. Наприклад, мобільний додаток, який запитує дані із сервера. Або

браузер, який надсилає запити з веб-сторінки на сервер для завантаження даних.

Додаток А може запитувати дані в додатку В. Тоді А – клієнт відносно В, а В - сервер відносно А. Одночасно А може обробляти запити від В, D, D тощо. У цьому випадку Додаток А - це і сервер, і клієнт. Все залежить від контексту. Однак не кожна система, компоненти якої взаємодіють через запити відповідей, є системою REST (або RESTful). Щоб система вважалася RESTful, вона повинна містити обмеження REST.

2.2 Приведення архітектури до моделі клієнт-сервер

В основі даного обмеження лежить розмежування потреб. Необхідно відокремлювати потреби клієнтського інтерфейсу від потреб сервера, що зберігає дані. Дане обмеження підвищує переносимість клієнтського коду на інші платформи, а спрощення серверної частини покращує масштабованість системи. Саме розмежування на «клієнт» і «сервер» дозволяє їм розвиватися незалежно один від одного.

Архітектура REST вимагає дотримання наступної умови. У період між запитами сервера не потрібно зберігати інформацію про стан клієнта і навпаки. Всі запити від клієнта повинні бути складені так, щоб сервер отримав всю необхідну інформацію для виконання запиту. Таким чином і сервер, і клієнт можуть "розуміти" будь-яке повідомлення, не спираючись при цьому на попередні повідомлення.

Клієнти можуть виконувати кешування відповідей сервера. У тих, у свою чергу, повинна бути явне або неявне позначення як кешована або некешована, щоб клієнти у відповідь на наступні запити не отримували застарілі або невірні дані.

Правильне використання кешування допомагає повністю або частково усунути деякі клієнт-серверні взаємодії, ще більше підвищуючи продуктивність і розширюваність системи.

До фундаментальних вимог REST архітектури відноситься і уніфікований, однаковий інтерфейс. Клієнт повинен завжди розуміти, в якому форматі і на які адреси йому потрібно надсилати запит, а сервер, в свою чергу, також повинен розуміти, в якому форматі йому слід відповідати на запити клієнта. Цей єдиний формат клієнт-серверної взаємодії, який описує, що, куди, в якому вигляді і як відсилати і є уніфікованим інтерфейсом.

Під шарами мається на увазі ієрархічна структура мереж. Іноді клієнт може спілкуватися безпосередньо з сервером, а іноді - просто з проміжним вузлом. Застосування проміжних серверів здатне підвищити масштабованість за рахунок балансування навантаження і розподіленого кешування.

Переваги, які надає REST:

- надійність (не потрібно зберігати інформацію про стан клієнта, яка може бути втрачена);
- продуктивність (за рахунок використання кеша);
- масштабованість;
- прозорість системи взаємодії;
- простота інтерфейсів;
- портативність компонентів;
- легкість внесення змін;
- здатність еволюціонувати, пристосовуючись до нових вимог.

2.3 Проектування бази даних

Необхідно здійснити проектування бази даних. Визначимо структуру бази та логічні зв'язки і представимо все у веб-додатку phpMyAdmin (рис. 2.4). Визначимо основні сутності:

- категорії;
- пресети;
- кольори;

- елементи;
- теги;
- текстури.

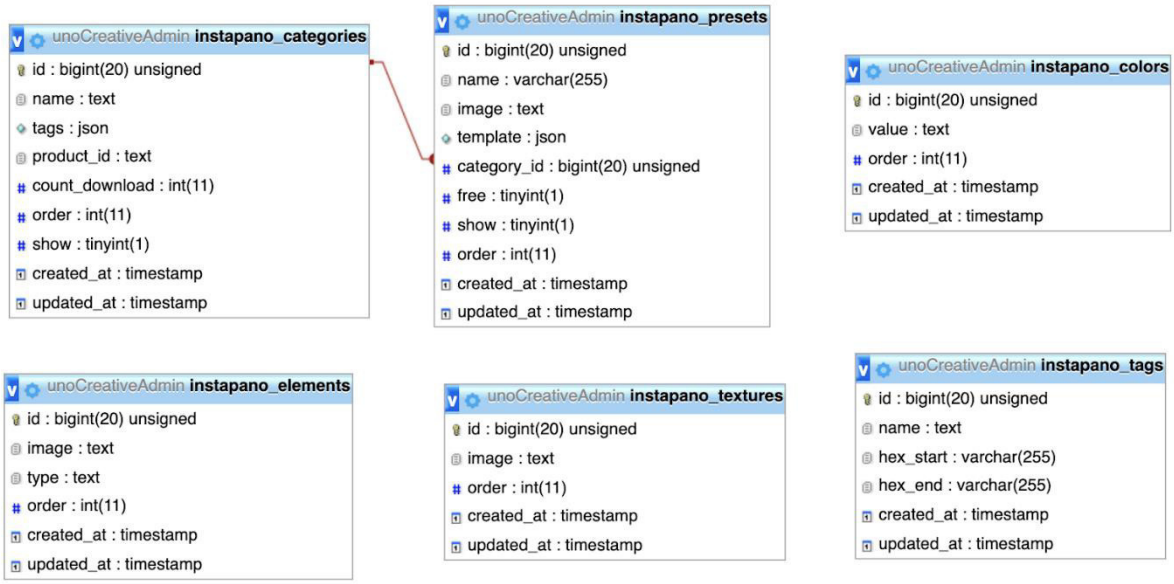

Рисунок 2.4 – Структура БД

Визначимо основні сутності:

Сутність «Категорії» містить інформацію про категорії та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;
- ім’я – має тип text, не може бути NULL;
- теги – має тип json, не може бути NULL;
- id продукту – має тип text, може бути NULL;
- кількість скачувань – має тип int(11), не може бути NULL;
- порядок – має тип int(11), не може бути NULL;
- показ – має тип tinyint(1), не може бути NULL;

Сутність «Пресети» містить інформацію про пресети та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;

- ім’я – має тип varchar(255), не може бути NULL;
- зображення – має тип text, не може бути NULL;
- шаблон – має тип json, може бути NULL;
- id користувача– ідентифікатор користувача, має тип bigint(20), є вторинним ключем, не може бути NULL;
- безкоштовно – має тип tinyint(1), не може бути NULL;
- показ – має тип tinyint(1), не може бути NULL;
- порядок – має тип int(11), не може бути NULL;

Сутність «Кольори» містить інформацію про кольори та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;
- значення – має тип text, не може бути NULL;
- порядок – має тип int(11), не може бути NULL;

Сутність «Елементи» містить інформацію про елементи та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;
- зображення – має тип text, не може бути NULL;
- тип – має тип text, не може бути NULL;
- порядок – має тип int(11), не може бути NULL;

Сутність «Теги» містить інформацію про теги та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;
- ім’я – має тип text, не може бути NULL;
- колір в початку – має тип varchar(255), не може бути NULL;
- колір в кінці – має тип varchar(255), не може бути NULL;

Сутність «Текстури» містить інформацію про текстури та має наступні атрибути:

- id – ідентифікатор користувача, має тип bigint(20), є первинним ключем, не може бути NULL;
- зображення – має тип text, не може бути NULL;

порядок – має тип int(11), не може бути NULL;

2.4 Розробка користувацького інтерфейсу

Всі частини проекту, адміністративний та веб-редактор, мобільний додаток повинні мати продуманий дизайн для зручності використання. Розробимо макети сторінок програмних модулів за допомогою сервісу Figma і наведемо їх нижче (рис. 2.5 – 2.9).

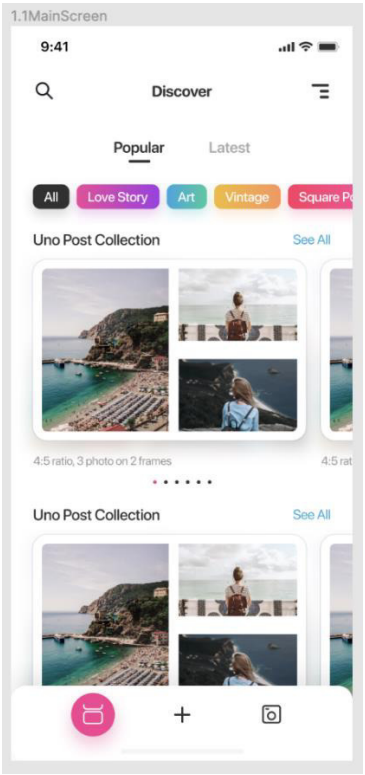

Рисунок 2.5 – Макет головної сторінки

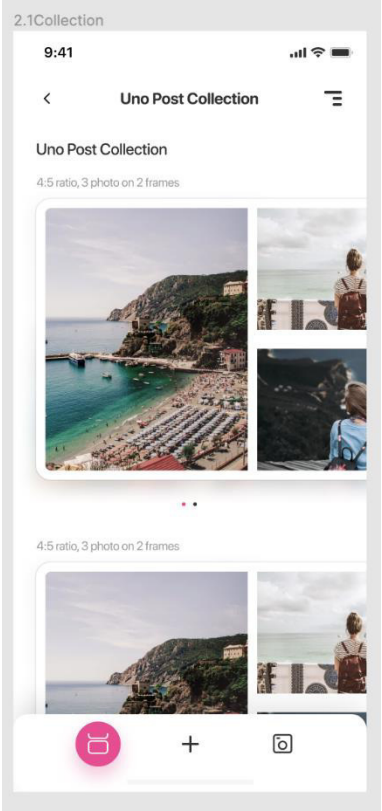

Рисунок 2.6 – Макет сторінки вибраної колекції

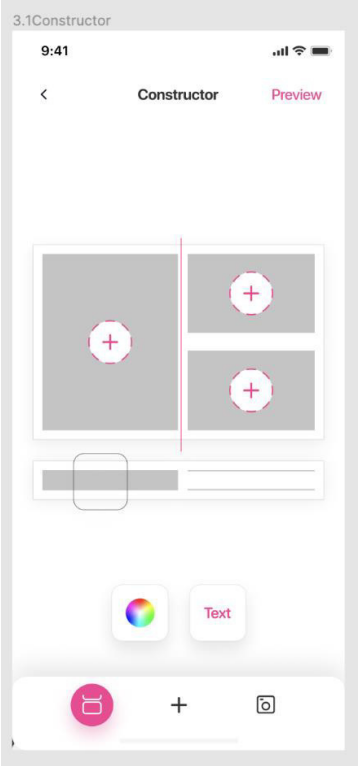

Рисунок 2.7 – Макет сторінки конструктору

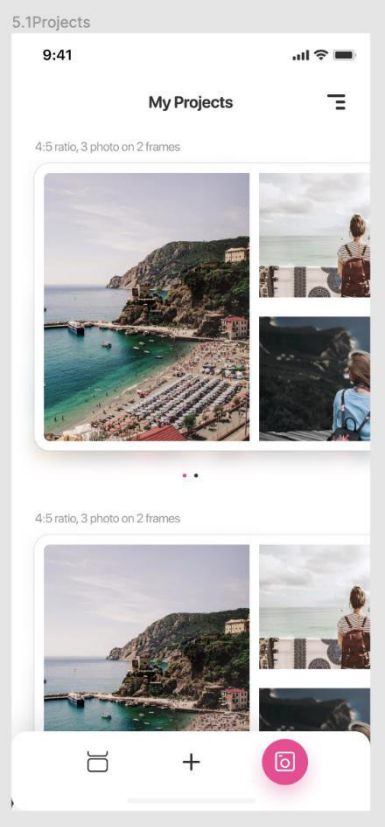

Рисунок 2.8 – Макет сторінки моїх проектів

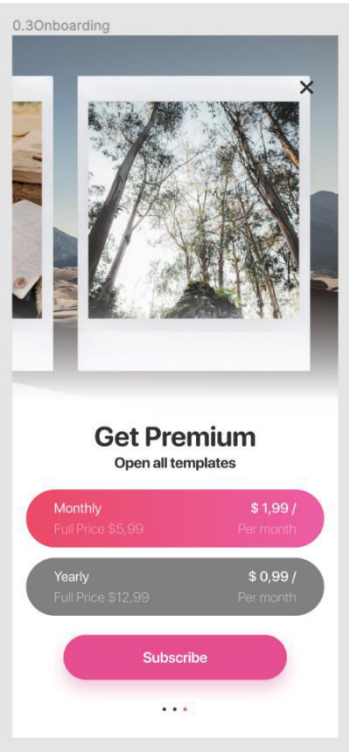

Рисунок 2.9 – Макет сторінки оформлення підписки

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ СТВОРЕННЯ INSTAGRAM-ІСТОРІЙ ТА ФОТОПРЕЗЕНТАЦІЙ»

3.1 Вибір та обґрунтування технологій для розроблення мобільного додатку

Додаток, що розробляється, складається з веб-додатків та мобільного додатка, тому для реалізації обрані можливості мови програмування JavaScript та бібліотеки React для веб-додатків та React-native для мобільних додатків.

React – це найпопулярніша бібліотека JavaScript для розробки користувальницького інтерфейсу (UI) [8].

Компоненти цього інструменту розроблені Facebook. Він був випущений як інструмент JavaScript з відкритим кодом у 2013 році. На даний момент React випереджає своїх основних конкурентів, таких як Angular та Bootstrap, дві найбільш популярні бібліотеки JavaScript свого часу.

React використовують сотні великих компаній по всьому світу, включаючи Netflix, Airbnb, American Express, Facebook, WhatsApp, eBay та Instagram. Це є доказом того, що інструмент має ряд переваг, з якими неможливо конкурувати.

Розглянемо переваги використання React.

1. React – це бібліотека графічного інтерфейсу користувача з відкритим кодом, орієнтована на одну конкретну мету – ефективне виконання завдань при розробці інтерфейсу користувача. Можна класифікувати як категорію «V» в архітектурному шаблоні MVC (модель-представлення-контролер).

2. React дозволяє повторно використовувати компоненти, розроблені в інших програмах, що використовують ту саму функцію. Можливість повторного використання компонента є очевидною перевагою для розробників.

3. Компонент React простіше створити, оскільки він використовує JSX, необов’язкове розширення синтаксису JavaScript, яке дозволяє поєднувати HTML з JavaScript. JSX – це гарне поєднання JavaScript та HTML. Це робить весь процес написання структури сайту зрозумілішим. Крім того, розширення також значно спрощує надання багатьох функцій. Незважаючи на те, що JSX може бути не найпопулярнішим розширенням синтаксису, воно виявилося ефективним у розробці користувацьких компонентів або великих програм.

4. React ефективно оновлює процес DOM (об'єктна модель документа). Інструмент дозволяє створювати віртуальні DOM і розміщувати їх у пам'яті. Як результат, щоразу, коли відбувається зміна реального DOM, віртуальний змінюється негайно. Ця система не дозволить фактичному DOM примушувати постійні оновлення. Як результат, швидкість програми не постраждає.

5. React дозволяє створювати користувальницькі інтерфейси, до яких можна отримати доступ з різних пошукових систем. Ця функція є величезною перевагою, оскільки не всі фреймворки JavaScript оптимізовані для SEO. Оскільки React може пришвидшити роботу додатка, він також може покращити результати SEO. Швидкість завантаження відіграє важливу роль в SEO-оптимізації. Однак варто зазначити, що React – це просто бібліотека JavaScript. І тому він не може зробити все сам. Необхідно використовувати додаткові бібліотеки для управління статусом, маршрутизації та взаємодії.

React Native – це фреймворк JavaScript для розробки мобільних додатків для операційних систем iOS та Android. React Native розроблений Facebook у 2015 році.

Розробка для iOS здійснюється на мові програмування Swift, а для Android – на Java або Kotlin. Таким чином, для того, щоб створити власний додаток для цих двох платформ, потрібно розробити два абсолютно різні програмні продукти.

Фреймворк React Native дозволяє розробляти мобільні додатки для iOS та Android, використовуючи лише одну мову програмування – JavaScript, а також використовувати в цих двох додатках більшу частину загального коду.

React Native має нижчу продуктивність, ніж інші рішення, і його використання для складних та нестандартних додатків є складним. React Native не повинен створювати ігри, програми для роботи з відео, фото, аудіо чи доповненою реальністю. Але в галузі бізнес-додатків React Native, безсумнівно, є гарним вибором – продуктивність у цьому випадку не надто відрізняється від власних рішень, а можливостей платформи достатньо для реалізації необхідних функціональних можливостей.

React Native дозволяє створювати мобільні додатки, використовуючи лише JavaScript із тією ж структурою, що і React. Це дає можливість складати багатофункціональний мобільний інтерфейс, використовуючи декларативні компоненти.

Ще однією перевагою технології є швидкозростаюче співтовариство компаній, які використовують цю технологію, інвестують у неї та підтримують її розвиток: GeekBrains, Yandex, Airbnb, Wix, Tesla, Soundcloud, Walmart.

3.2 Інтерфейс додатку

В результаті розробки мобільного додатку кінцевий продукт відповідає розробленим макетам (рис. 3.1 – 3.5).

Головна сторінка містить список колекцій, фільтрів та пошук для більш зручного знаходження потрібної колекції.

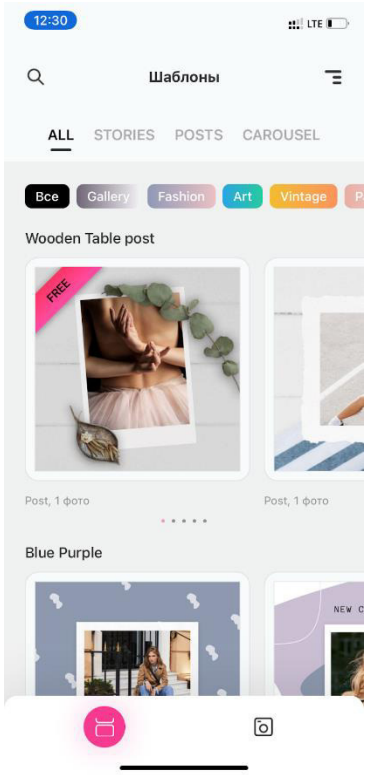

Рисунок 3.1 – Головна сторінка

Сторінка вибраної колекції містить список шаблонів.

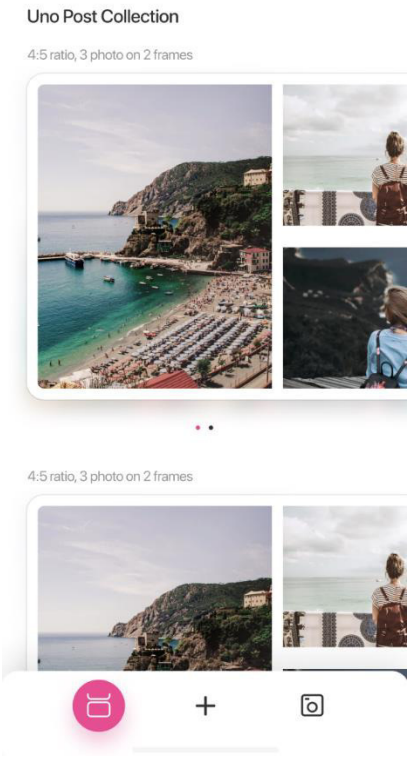

Рисунок 3.2 – Сторінка вибраної колекції

Сторінка дизайнера містить набір інструментів для роботи з фотографіями та текстом, а саме:

- зміна розміру;
- поворот на потрібний кут;
- зміна кольору тексту.

А також перегляд роботи та збереження її в галерею або у своїх проектах.

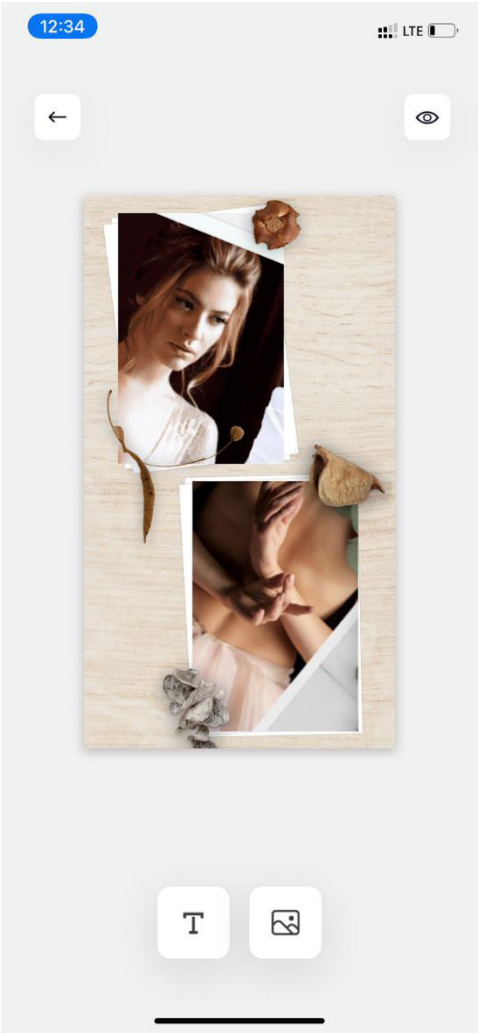

Рисунок 3.3 – Сторінка конструктору

Сторінка «Мої проекти» містить список робіт, які можна видалити або відредагувати.

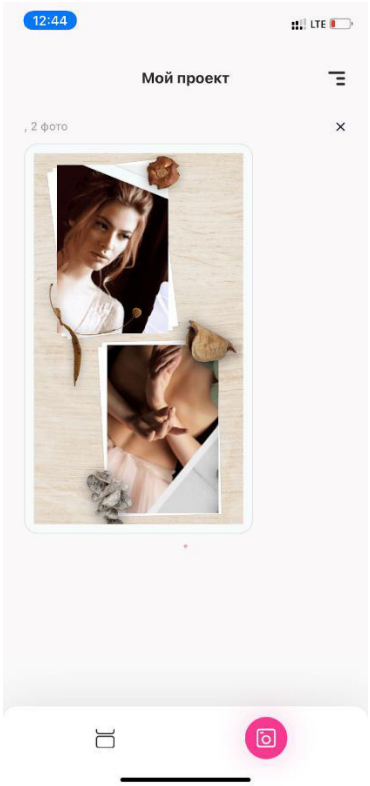

Рисунок 3.4 – Сторінка мої проекти

Сторінка підписки містить можливість оформлення підписки та поновлення підписки.

Рисунок 3.5 – Сторінка оформлення підписки

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ВВЕДЕННЯ В ЕКСПЛУАТАЦІЮ

4.1 Вибір виду тестування

Розглянемо головні чотири підходи, що застосовуються в тестуванні програм для телефону: на базі емуляції, в хмарі на базі пристроїв і з впровадженням краудсорсингу.

Емулятори. Емулятори є непоганим методом для початку проведення тестів. Вони допомагають виявити більшу частину помилок на самій ранній стадії життєвого циклу розробки ПЗ. Якщо в наявності немає потрібних девайсів або інший пристрій зайнято іншим тестувальником, на допомогу приходять мобільні емулятори, які здатні повністю імітувати поведінку телефону / планшетного комп'ютера. Спочатку вони розроблялися для проведення тестів, тому є частина SDK розробника. Зручно, що емуляторі запускаються на ПК або серверах, більше потужних пристроях, ніж телефон. Однак тому деякі помилки можуть бути не знайдені або викликати неправильне уявлення, наприклад, час відгуку або продуктивність програми. На рис. 4.1 зображено схему тестування на основі емуляції.

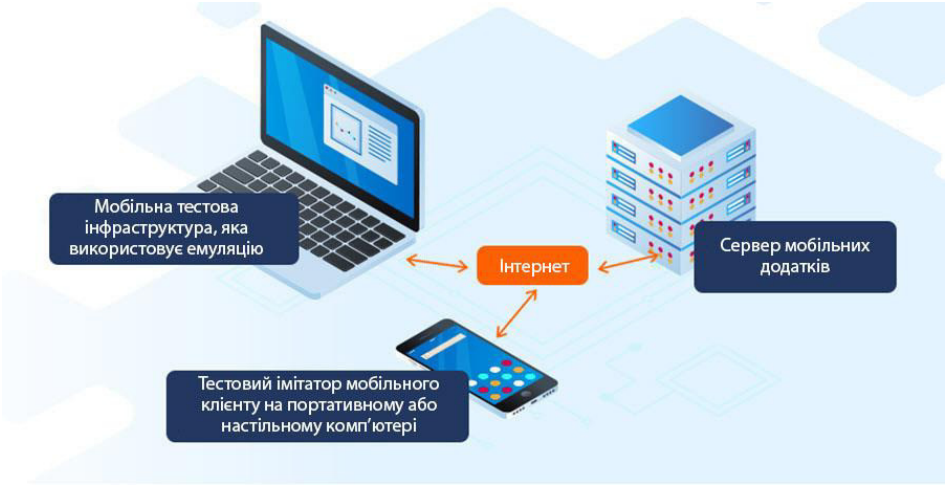

Рисунок 4.1 – Схема тестування на основі емуляції

Важливим фактором є те, що тестування на емуляторі дешевше порівняно з покупкою нового пристрою.

До недоліків належать помилкові спрацьовування, обмежений набір жестів та багато іншого, що не можна перевірити без мобільного пристрою. Фактичне тестування є дуже важливим, оскільки емулятори просто не можуть охопити всіх проблем, які можуть виникнути під час безпосередньої взаємодії з користувачем. Для найкращого результату тестування краще використовувати змішаний підхід - за допомогою емулятора та пристрою. При такому підході вам потрібно лише розподілити тести.

У хмарі. Device Cloud - це мобільне середовище, яке включає справжні пристрої Android та iOS з різними комбінаціями версій ОС, розмірів екрану, оперативної пам’яті тощо. Хмарні пристрої - прекрасне рішення для широкомасштабного мобільного тестування.

Хмарне тестування дозволяє підключатися до різних мобільних пристроїв незалежно від їх розташування. Пристрої підтримують паралельне тестування, придатні для швидкого розвитку, записують результати та доступні цілодобово. З точки зору безпеки ви можете вибрати приватну хмару замість загальнодоступної. На рис. 4.2 показана схема хмарного тестування.

Рисунок 4.2 – Схема тестування у хмарі

У реальній хмарі пристроїв тестер може протестувати користувальницький інтерфейс на кожному пристрої, виміряти продуктивність, відстежувати, як додаток працює на розрядженому акумуляторі, а також у режимі офлайн тощо.

Що стосується ціни, то для хмарного тестування вам доведеться заплатити лише за тестове середовище, яке використовується. Хмарне тестування мобільних додатків використовується для мінімізації витрат на створення тестових лабораторій. Цей тип тестування мобільних додатків є фінансово вигіднішим, ніж тестування мобільних додатків на реальних мобільних пристроях.

На базі пристроїв. Тестування мобільних додатків на основі пристрою показує найкращі результати, оскільки програмне забезпечення є точно таким же, як для кінцевого користувача. При тестуванні на реальних мобільних пристроях враховуються всі особливості ОС, а також якість мережевих послуг (QoS).

Це дорогий підхід до тестування, оскільки всі пристрої потрібно придбати. А враховуючи те, що ринок смартфонів досить часто зростає та оновлюється - це не вигідна інвестиція. Тільки фінансова сторона при такому підході є недоліком. На рис. 4.3 зображено схему тестування на базі пристроїв.

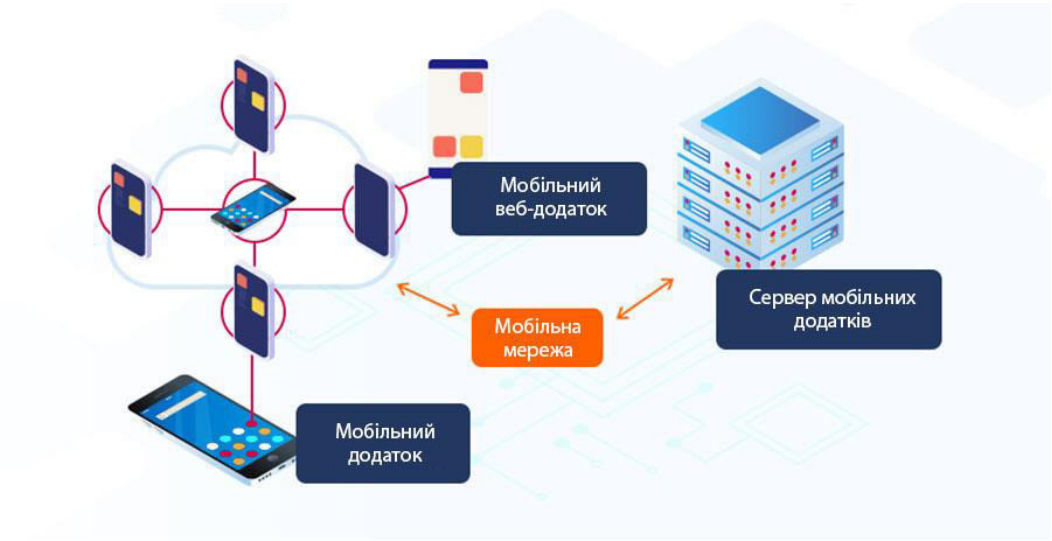

Рисунок 4.3 – Схема тестування на базі пристроїв

З використанням краудсорсингу. Краудсорсинг – гарний спосіб перевірити функціональність програми на пізніх стадіях розробки, щоб охопити всі можливі варіанти використання.

Фрілансери тимчасово наймаються для тестування. Існують спеціальні сайти, де є користувачі, які готові протестувати програмне забезпечення, а також ті, хто потребує тестування. Таким чином, клієнт може протестувати сайт спеціально для своєї цільової аудиторії.

На рис. 4.4 зображено схему тестування з використанням краудсорсингу.

Рисунок 4.4 – Схема тестування з використанням краудсорсингу

Тестери можуть використовувати власні пристрої для тестування програми або доступу до емуляторів пристроїв через платформу тестування краудсорсингу, залежно від вимог замовника.

Завдяки такому підходу є можливість отримати важливі відгуки для покращення UX та забезпечення зручності використання.

До недоліків краудсорсингу належать можливі проблеми з конфіденційністю програми, що тестується, а також ненадійність тестувальників та їх роботи.

Кожен підхід має свої переваги та недоліки при тестуванні мобільних

додатків. Емулятори добре підходять для тестування користувальницького інтерфейсу та початкового контролю якості, для перевірки продуктивності потрібні реальні пристрої, а хмарне тестування - хороший спосіб протестувати додаток на великій кількості пристроїв та операційних систем. Краудсорсинг дозволяє протестувати програмне забезпечення в різних умовах, наближаючи їх до реальних [10].

4.2 Тест-план

Для проведення тестування необхідно скласти плани тестування та визначити функції, які будуть тестуватися.

Особливості, які потрібно протестувати відповідно до плану тестування мобільного додатка:

- перевірка можливості переглядати шаблони;
- редагування шаблону;
- збереження шаблону.

Таблиця 4.1 – Тест-кейси для мобільного додатку

Опис	Перевірка можливості переглядати шаблони	
Передумови	Завантажена сторінка «Головна»	
№	Дія	Очікуваний результат
1	Прокрутити сторінку донизу	Відображаються шаблони
2	Натиснути на тег	Відбувається фільтрація шаблонів по обраному тегу, та буде відображаються актуальні шаблони
3	Натиснути на пошук та надрукувати текст	Відбувається фільтрація шаблонів по назві, та буде відображаються актуальні шаблони

Продовження таблиці 4.1

4	Натиснути на шаблон	Відбудеться перехід на сторінку редактора
Опис	Редагування шаблонів	
Передумови	Завантажена сторінка «Редактор»	
№	Дія	Очікуваний результат
1	Завантажити фотографію	Фотографія відобразиться
2	Відредагувати текст	Відбудеться зміна тексту в макеті
3	Натиснути на збереження шаблону	Користувач отримує можливість вибору місця куди відбувається збереження шаблону
Опис	Збереження шаблону	
Передумови	Завантажена сторінка «Зберігання»	
№	Дія	Очікуваний результат
1	Натиснути на збереження шаблону в галерею	Відбудеться конвертація шаблону в png та збереження в галереї
2	Натиснути на збереження шаблону в «Мої проекти»	Відбудеться збереження шаблону

Тестування проводилося на пристроях на базі операційної системи IOS та Android. Під час тестування у вищезазначених умовах мобільний додаток продемонстрував свою ефективність. Тестування мобільного додатку не виявило жодних недоліків, усі контрольні списки успішно пройшли.

4.3 Введення в експлуатацію

Після того, як мобільний додаток буде готовий до публікації в App Store

та Play Market, його можна завантажити за допомогою Xcode та Android Studio відповідно.

Введення в експлуатацію здійснюється за допомогою XCode та App Store Connect. На рис. 4.5 зображено App Store Connect.

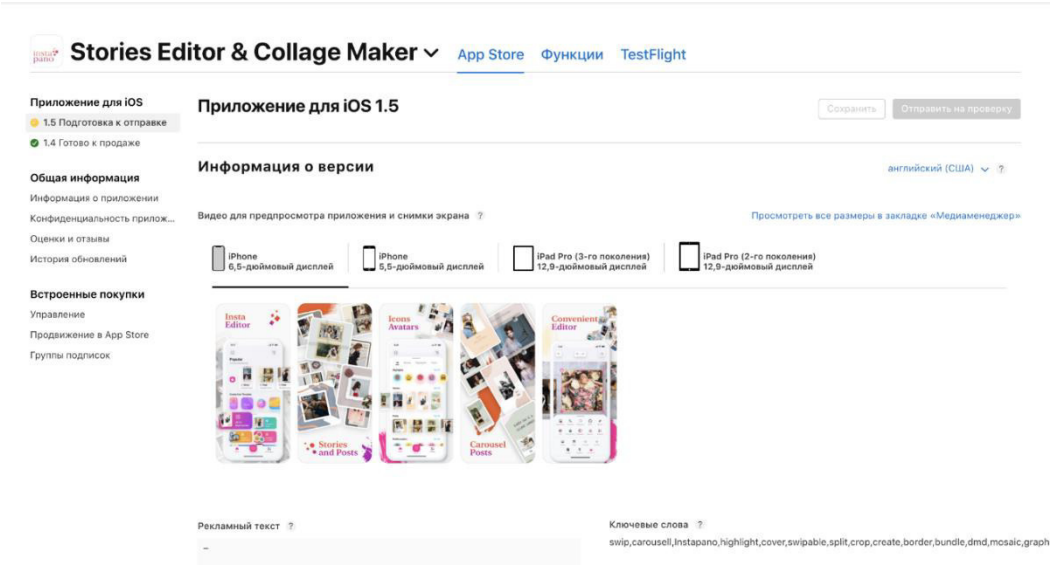

Рисунок 4.5 – App Store Connect за адресою – https://appstoreconnect.apple.com/

Введення в експлуатацію здійснюється за допомогою XCode та App Store Connect. На рисунку 4.4 зображено App Store Connect.

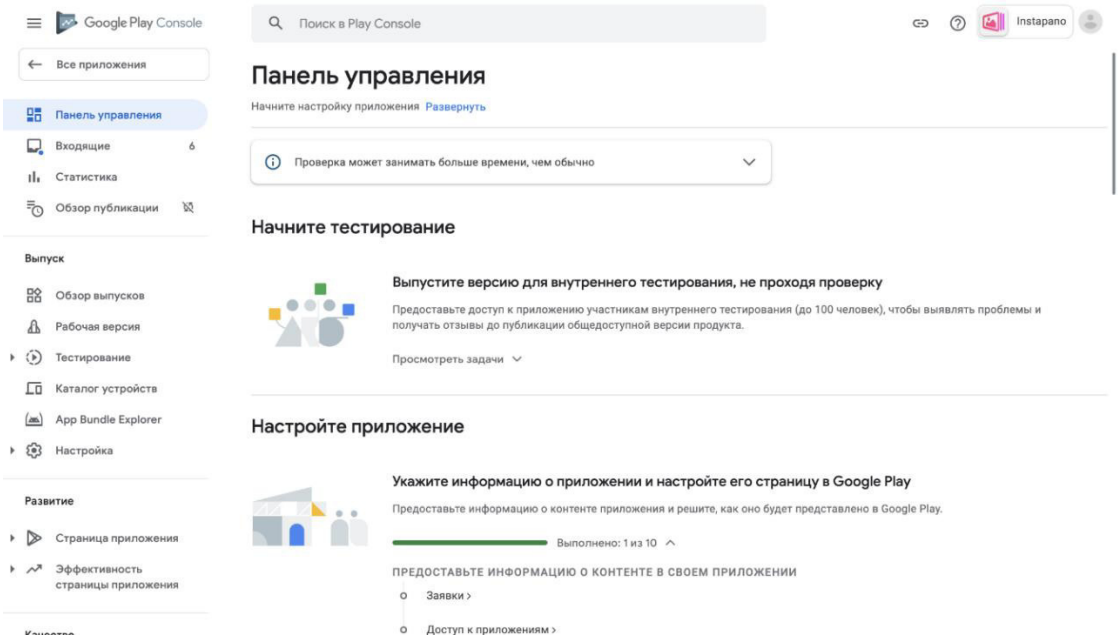

Рисунок 4.4 – Google Play Console за адресою – https://play.google.com/console

ВИСНОВКИ

Мета дипломної роботи полягала в розробці мобільного додатку для створення Instagram-історій та фотопрезентацій. Також розроблено два веб-додатки: систему управління контентом та редактор для створення шаблонів.

Система управління контентом надає можливість зручно керувати шаблонами, переглядати вже створені шаблони, створювати нові та додавати інформацію про них, редагувати, видаляти шаблони. Для кожного шаблону створено направлення до веб-редактора, для зручної розробки та редагування шаблонів.

Веб-редактор створений для дизайнерів має зручний інтерфейс, надає можливість додавати текст, фотографії, елемент для завантаження фотографій користувачем, різни фігури, фонове зображення шаблону, для кожного з цих графічних елементи створено безліч допоміжних налаштувань для розміщення на полотні макету, зміни стилів і інших налаштування з якими може зіткнутись дизайнер під час створення макету.

Мобільний додаток розроблених для користувача, надає можливість переглядати доступні макети, додавати елементи, редагувати їх під свої потреби. Після редагування макету користувач має можливість зберегти в розділ «Мої проекти» або відразу в форматі зображення в галерею. Користувач може оглядати відредаговані шаблоні і вносити в них зміни.

Мобільний додаток для створення Instagram-історій та фотопрезентацій пройшов тестові випробування та опублікований в App Store та Play Market.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології та моделювання бізнес-процесів : навч. посіб. / О. М. Томашевський, Г. Г. Цегелик, М. Б.Вітер, В. І. Дубук. – К. : ЦУЛ, 2012. - 296 с.

2. Основные преимущества СУБД MySQL [Електронний ресурс]. – Режим доступу: https://studopedia.su/4_4741_osnovnie-preimushchestva-subd-MySQL.html

3. Введение в JavaScript [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/intro

4. Навчальний посібник з дисципліни «Методи тестування і оцінки якості програмного забезпечення» для студентів денної та заочної форми навчання: 6.050101 – «Комп'ютерні науки та інформаційні технології». Укл.: Колектив провідної української компанії з тестування програмного забезпечення QATestLab, О.Л. Ляхов, О.О. Бородіна. / Полтава: ПолтНТУ, 2015 – 372 с.

5. Г.В. Головко, А.М. Гафіяк Управління базами даних. Частина 1. Проектування баз даних. Бази даних та інформаційні системи. Конспект лекцій – Полтава, Полтавський національний технічний університет імені Юрія Кондратюка, 2008

6. Ідентифікація та аутентифікація [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/identifikaciataautentifikacia/

7. REACT для початківців від Харламова Іллі [Електронний ресурс]. – Режим доступу: https://ikharlamov21.gitbook.io/react-book/

8. Redux. A Predictable State Container for JS Apps [Електронний ресурс]. – Режим доступу: https://redux.js.org

9. Підходи до тестування мобільних додатків від компанії QATestL [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/approaches-to-testing-mobile-applications/

10. Сторіс в Інстаграм -усе,що потрібно знати про тренди просування [Електронний ресурс]. – Режим доступу: https://creativesmm.com.ua/storis-v-instahram-use-shcho-vam-potribno-znaty-pro-trendy-u-2019/

ДОДАТОК А
ПРОГРАМНИЙ КОД МОБІЛЬНОГО ДОДАТКУ

package.json

```
{
 "name": "InstaPano",
 "version": "0.0.1",
 "private": true,
 "scripts": {
   "android": "react-native run-android",
   "ios": "react-native run-ios",
   "start": "react-native start",
   "test": "jest",
   "lint": "eslint ."
 },
 "dependencies": {
   "@dudigital/react-native-zoomable-view": "^1.0.15",
   "@react-native-community/async-storage": "^1.9.0",
   "@react-native-community/blur": "^3.6.0",
   "@react-native-community/cameraroll": "^4.0.0",
   "@react-native-community/image-editor": "^2.3.0",
   "@react-native-community/masked-view": "^0.1.7",
   "@react-native-community/picker": "^1.6.6",
   "@react-native-community/slider": "^3.0.3",
   "@react-navigation/bottom-tabs": "^5.6.1",
   "@react-navigation/native": "^5.6.1",
   "@react-navigation/stack": "^5.2.8",
   "@unimodules/core": "^6.0.0",
   "add": "^2.0.6",
   "axios": "^0.19.2",
   "babel-plugin-inline-import": "^3.0.0",
   "buffer": "^5.6.0",
   "colorsys": "^1.0.22",
   "expo": "^40.0.0",
   "expo-asset": "^8.2.1",
   "expo-gl": "^9.2.0",
   "expo-graphics": "^1.1.0",
   "expo-three": "^5.5.1",
   "find": "^0.3.0",
   "fs": "^0.0.1-security",
   "i18next": "^19.3.4",
   "immutable": "^4.0.0-rc.12",
   "lodash": "^4.17.20",
```

```
    "lottie-react-native": "^3.5.0",
    "markdown-it": "^10.0.0",
    "moment": "^2.24.0",
    "pngjs": "^5.0.0",
    "prop-types": "^15.7.2",
    "qs": "^6.9.3",
    "query-string": "^6.11.1",
    "react": "16.11.0",
    "react-dom": "^16.13.1",
    "react-i18next": "^11.3.4",
    "react-native": "0.62.0",
    "react-native-animatable": "^1.3.3",
    "react-native-canvas": "^0.1.37",
    "react-native-color-wheel": "https://github.com/netbeast/react-native-color-
wheel.git",
    "react-native-customized-image-picker": "^0.2.0",
    "react-native-device-info": "^5.5.4",
    "react-native-directed-scrollview": "^2.0.0",
    "react-native-drawer": "^2.5.1",
    "react-native-fast-image": "^8.1.5",
    "react-native-flash-message": "^0.1.15",
    "react-native-fs": "^2.16.6",
    "react-native-gesture-handler": "^1.6.1",
    "react-native-iap": "^5.1.1",
    "react-native-image-pan-zoom": "^2.1.12",
    "react-native-image-picker": "^2.3.3",
    "react-native-image-viewer": "^0.0.3",
    "react-native-image-zoom-viewer": "^3.0.1",
    "react-native-interactable": "^2.0.1",
    "react-native-iphone-x-helper": "^1.2.1",
    "react-native-keyboard-accessory": "^0.1.11",
    "react-native-linear-gradient": "^2.5.6",
    "react-native-loading-spinner-overlay": "^1.1.0",
    "react-native-modal": "^11.5.6",
    "react-native-orientation": "^3.1.3",
    "react-native-photo-view-ex": "^1.1.0",
    "react-native-picker-select": "^8.0.0",
    "react-native-pinch-zoom-view": "^0.2.0",
    "react-native-rate": "^1.2.1",
    "react-native-reanimated": "^1.10.1",
    "react-native-redash": "^14.2.3",
    "react-native-render-html": "^4.2.2",
    "react-native-safe-area-context": "^0.7.3",
    "react-native-save-view": "^0.2.3",
    "react-native-scalable-image": "^1.0.0",
    "react-native-screens": "^2.4.0",
    "react-native-share": "^3.1.2",
```

```
    "react-native-snap-carousel": "^4.0.0-beta.6",
    "react-native-splash-screen": "^3.2.0",
    "react-native-svg": "^12.1.0",
    "react-native-svg-animations": "^0.2.1",
    "react-native-svg-uri": "^1.2.3",
    "react-native-swipe-gestures": "^1.0.5",
    "react-native-video": "^4.4.5",
    "react-native-view-shot": "^3.1.2",
    "react-native-web": "^0.13.5",
    "react-native-webview": "^10.3.3",
    "react-navigation": "^4.3.7",
    "react-redux": "^7.2.0",
    "react-three-fiber": "^5.3.10",
    "recompose": "^0.30.0",
    "redux": "^4.0.5",
    "redux-devtools-extension": "^2.13.8",
    "redux-persist": "^6.0.0",
    "redux-persist-transform-immutable": "^5.0.0",
    "redux-saga": "^1.1.3",
    "reselect": "^4.0.0",
    "rn-fetch-blob": "^0.12.0",
    "stream": "^0.0.2",
    "styled-components": "^5.1.0",
    "svg-to-jsx": "^1.0.2",
    "three": "^0.124.0",
    "throttle-debounce": "^2.1.0",
    "timers": "^0.1.1",
    "yarn": "^1.22.4",
    "zlib": "^1.0.5",
    "zoomable-svg": "^5.0.1"
  },
  "devDependencies": {
    "@babel/core": "^7.9.0",
    "@babel/runtime": "^7.9.2",
    "@react-native-community/eslint-config": "^1.0.0",
    "babel-jest": "^25.2.3",
    "babel-plugin-module-resolver": "^4.0.0",
    "eslint": "^6.8.0",
    "jest": "^25.2.3",
    "metro-react-native-babel-preset": "^0.59.0",
    "module-resolver": "^1.0.0",
    "react-native-svg-transformer": "^0.14.3",
    "react-test-renderer": "16.11.0"
  },
  "jest": {
    "preset": "react-native"
  }
```

```
}
```

App.js

```
import "react-native-gesture-handler";
import "./config-i18n";

import React, { useEffect, useState } from "react";
import { StatusBar, Text } from "react-native";

import AppNavigation from "./navigation/AppNavigation";
import Orientation from "react-native-orientation";
import OrientationContainer from "./containers/orientation";
import { PersistGate } from "redux-persist/integration/react";
import { Provider } from "react-redux";
import RootContainer from "./containers/root";
import MenuContainer from "./containers/menu";
import SplashScreen from "react-native-splash-screen";
import SubscriptionContainer from "./containers/Subscription";
import { TabBarContext } from "@/context";
import configureStore from "./config-store";

const { store, persistor } = configureStore();

const App = () => {
 const [showTabBar, setShowTabBar] = useState(null);
 useEffect(() => {
   Orientation.lockToPortrait();

   SplashScreen.hide();
 }, []);

 return (
   <>
     <StatusBar barStyle="dark-content" backgroundColor="#fff" />

     <PersistGate loading={null} persistor={persistor}>
       <TabBarContext.Provider
         value={{
           showTabBar,
           changeShowTabBar: setShowTabBar
         }}
       >
         <Provider store={store}>
```

```
            <RootContainer>
              <OrientationContainer>
                <SubscriptionContainer>
                  <AppNavigation />
                </SubscriptionContainer>
              </OrientationContainer>
            </RootContainer>
          </Provider>
        </TabBarContext.Provider>
      </PersistGate>
    </>
  );
};

export default App;
```

HomeView.js

```
import { View } from "react-native";
import React from "react";
import {
 StyledContainer,
 StyledContent,

} from "./home-styled";

import { TabBar } from "@/components";

import HomeCreateFromTemplate from "./components/create-from-template";
import HomeHeader from "./components/header";
import HomeMenu from "./components/menu";
import HomeModal from "./components/modal";
import HomePopular from "./components/popular";
import { useSafeArea } from "react-native-safe-area-context";
import { useTranslation } from "react-i18next";

const HomeView = ({
 presetsData,
 defaultData,
 search,
 showSearch,
 showDrawer,
 showModal,
 activeSort,
```

```
 showHelp,
 purchases,
 isSubscribe,
 heightContent,
 onChangeSearch,
 onChangeShowSearch,
 onShowDrawer,
 onChangeActiveSort,
 onChangeShowHelp,
 onChangeShowModal,
 onChangeHeightContent,
}) => {
 const { t } = useTranslation("home");

 const insets = useSafeArea();

 return (
   <View style={{ flex: 1 }}>

     <HomeMenu
       showDrawer={showDrawer}
       changeShowDrawer={onShowDrawer}
       onChangeShowHelp={onChangeShowHelp}
     >
       <StyledContainer insets={insets}>
         <HomeHeader
           search={search}
           showSearch={showSearch}
           showHelp={showHelp}
           onChangeShowHelp={onChangeShowHelp}
           onShowDrawer={onShowDrawer}
           onChangeSearch={onChangeSearch}
           onChangeShowSearch={onChangeShowSearch}
         />

         <StyledContent onLayout={(event) =>
onChangeHeightContent(event.nativeEvent.layout.height)}>
           <HomePopular presetsData={defaultData} />

           <HomeCreateFromTemplate />
         </StyledContent>


       </StyledContainer>

       <HomeModal
         defaultData={defaultData}
```

```
          presetsData={presetsData}
          showModal={showModal}
          activeSort={activeSort}
          purchases={purchases}
          isSubscribe={isSubscribe}
          heightContent={heightContent}
          onChangeShowModal={onChangeShowModal}
          onChangeActiveSort={onChangeActiveSort}
        />

        <TabBar />

      </HomeMenu>
    </View>
  );
};

export default HomeView;
```

HomeRedux.js

```
import { changeShowDrawer, changeShowModal } from "src/modules/home/actions";
import { compose, withProps } from "recompose";
import { favoritesSelector, showModalSelector } from "src/modules/home/selectors";
import {
 isSubscribeSelector,
 purchasesSelector,
} from "src/modules/iap/selectors";
import {
 loadingPresetSelector,
 presetDataSelector,
} from "src/modules/preset/selectors";

import HomeContainer from "./home-container";
import { connect } from "react-redux";
import { fetchPresets } from "src/modules/preset/actions";
import { saveFavorites } from "src/modules/home/actions";

const mapStateToProps = (state) => {
 return {
   presetsData: presetDataSelector(state),
   loading: loadingPresetSelector(state),
   isSubscribe: isSubscribeSelector(state),
   favorites: favoritesSelector(state),
   showModal: showModalSelector(state),
```

```
   purchases: purchasesSelector(state),
 };
};

const mapDispatchToProps = {
 fetchPresets,
 changeShowDrawer,
 changeShowModal,
 saveFavorites,
};

export default compose(
 withProps(),
 connect(mapStateToProps, mapDispatchToProps)
)(HomeContainer);
```

HomeContainer.js

```
import React, { useContext, useEffect, useMemo, useState } from "react";

import HomeView from "./home-view";
import { OrientationContext } from "@/context";
import { useNavigation } from "@react-navigation/native";

const HomeContainer = ({
 showModal,
 presetsData,
 loading,
 isSubscribe,
 purchases,
 fetchPresets,
 changeShowModal
}) => {
 const { isVerticalOrientation } = useContext(OrientationContext);
 const navigation = useNavigation();

 const [successShowSubscribe, setSuccessShowSubscribe] = useState(false);
 const [showSearch, setShowSearch] = useState(false);
 const [showDrawer, setShowDrawer] = useState(false);
 const [search, setSearch] = useState(null);
 const [activeFilter, setActiveFilter] = useState(["all"]);
 const [activeSort, setActiveSort] = useState("all");
 const [scrollPosition, setScrollPosition] = useState(0);
```

```
const [shortHeader, setShortHeader] = useState(false);
const [showContent, setShowContent] = useState(false);
const [heightContent, setHeightContent] = useState(0);

const renderData = useMemo(() => {
  let res = {};

  presetsData.forEach(group => {

    const item = group.presets[0];

    const type =
      item.template.canvas.type === "square"
        ? "post"
        :
        item.template.canvas.type === "horizontal"
        ? "carousel"
        : item.template.canvas.type;

    // items.push(item)

    if (!res[type]) {
      res[type] = {
        type
      }

      res[type].items = [item]
    } else res[type].items.push(item)
  });

  if (activeSort === "all") return res;
  else {
    return {
      [activeSort]: res[activeSort]
    }
  }

}, [search, activeFilter, activeSort, presetsData]);

 useEffect(() => {
  let timerNavigate = null;
  let timerContent = null;

  fetchPresets();

  if (!isSubscribe) {
    navigation.navigate("Subscription", { firstStart: true });
```

```
  }

  return () => {
    clearTimeout(timerNavigate);
    clearTimeout(timerContent);
  };
}, []);

useEffect(() => {
  if (
    !isSubscribe &&
    !loading &&
    !successShowSubscribe
  ) {
    setSuccessShowSubscribe(true);
  }

}, [loading, renderData]);


const handleClearFilter = () => {
  setSearch(null);
  setShowSearch(false);
};

const handlerShowDrawer = () => {
  setShowDrawer(!showDrawer);
};

const handleChangeActiveFilter = value => {
  if (activeFilter == value) setActiveFilter("all");
  else setActiveFilter(value);
};

const handleChangeShowSearch = value => {
  setShowSearch(value);

  if (!value) handleClearFilter();
};

const handlerChangeActiveSort = value => {
  if (value === activeSort) setActiveSort("all");
  else setActiveSort(value);
};

const handlerScroll = position => {
  if (position > scrollPosition + 100 || position < scrollPosition - 100) {
```

```
      if (position > scrollPosition && scrollPosition > 100 && !shortHeader) {
        setShortHeader(true);
        console.log("ddd");
      } else if (position < scrollPosition && shortHeader) {
        setShortHeader(false);
        console.log("ddd");
      }

      setScrollPosition(position);
    }
  };

  return (
    <HomeView
      showModal={showModal}
      presetsData={renderData}
      defaultData={presetsData}
      search={search}
      loading={loading}
      showSearch={showSearch}
      showDrawer={showDrawer}
      isVerticalOrientation={isVerticalOrientation}
      activeSort={activeSort}
      activeFilter={activeFilter}
      isSubscribe={isSubscribe}
      purchases={purchases}
      shortHeader={shortHeader}
      showContent={showContent}
      heightContent={heightContent}
      onChangeHeightContent={setHeightContent}
      onChangeShowModal={changeShowModal}
      onClearFilter={handleClearFilter}
      onChangeSearch={setSearch}
      onShowDrawer={handlerShowDrawer}
      onChangeShowSearch={handleChangeShowSearch}
      onChangeActiveSort={handlerChangeActiveSort}
      onChangeActiveFilter={handleChangeActiveFilter}
      onRefresh={fetchPresets}
      onScroll={handlerScroll}
    />
  );
};

export default HomeContainer;
```

HomeStyled.json

```
import { heightTopBar, isTablet } from "@/helpers";
import { margin, padding } from "@theme/spacing";

import styled from "styled-components/native";

export const StyledContainer = styled.View`
 position: relative;
 background-color: #F8F8FA;
`;

export const StyledContent = styled.View`
 padding-bottom: 24px;
`;

export const StyledHeader = styled.View`
 padding-bottom: ${isTablet ? padding.large : padding.small}px;

 background-color: #f6fafb;

 z-index: 1;

 ${props =>
   props.fixed
     ? `
   position: absolute;
   top: 0px;
   left: 0px;
   right: 0px;
 `
     : ""}
`;

export const StyledSortContainer = styled.View`
`;

export const StyledSearch = styled.View`
 padding: 0px ${padding.large}px ${padding.large}px;
`;

export const StyledBody = styled.View``;

export const StyledHelp = styled.View`
```

```
 margin-top: ${isTablet ? padding.large : 0}px;
`;

export const StyledKeyboardWatch = styled.View`
 flex: 1;

 justify-content: center;
`;
```