Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Розробка редактора піксельної графіки для платформи Android

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
Калітовський Б. В.
_____
(підпис)
“____” _____ 2021 р.

Виконала студентка
Куренкова О. В.
“____” _____ 2021 р.

Київ - 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
Доцент., к. ф.-м. н.
_____О. П. Жежерун
(підпис)
“____” _____ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

Студентці Куренковій Олені Володимирівні факультету інформатики 3 курсу
ТЕМА Розробка редактора піксельної графіки для платформи Android
Вихідні дані: мобільний застосунок з можливістю створення та збереження піксельної графіки.
Зміст ТЧ до курсової роботи:

Календарний план
Вступ
1. Актуальність та огляд існуючих рішень. Постановка завдання.
2. Теоретичні відомості щодо технологій мобільної розробки
3. Опис реалізації програмного продукту
Висновки
Список використаної літератури
Додатки

Дата видачі “____” _____ 2021 р. Керівник _____
(підпис)
Завдання отримала _____
(підпис)

Тема: Розробка редактора піксельної графіки для платформи Android

Календарний план виконання роботи:

п/п	Назва етапу	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи	листопад 2020	
2.	Вивчення предметної області	листопад-грудень 2020	
3.	Вивчення технологій для розробки	січень 2021 р	
4.	Побудова технічного завдання	січень-лютий 2021р	
5.	Написання практичної частини роботи	лютий-березень 2021 р.	
6.	Написання текстової частини роботи	квітень 2021 р	
7.	Перегляд змісту роботи з керівником	квітень-травень 2021	
8.	Внесення змін до курсової роботи відповідно до зауважень наукового керівника	травень 2021	
9.	Створення презентації	травень 2021	
10.	Захист роботи	травень 2021	

Студент Куренкова О. В.

Керівник Калітовський Б. В.

“_____” _____

Зміст

Анотація

У цій курсовій роботі описана розробка мобільного редактора для піксельної графіки на платформі Android засобами React Native. Проведено аналіз предметної області та розглянуто існуючі рішення та технічні засоби, інструменти мобільної розробки.

Вступ

Піксельна графіка з’явилась разом з першими комп’ютерними іграми. Обмежене місце, обмежені кольори - усе змушувало розробляти візуальний дизайн до останнього пікселя. З часом технології поліпшились, можливостей стало більше, кольорів тепер не 256, а понад 16 мільйонів. Але піксельне мистецтво залишилось популярним серед інді розробників, художників. Спробувати відтворити піксельні зображення може кожен, хто має доступ до редактору растрових зображень.

Метою курсової роботи є розробка редактора піксельної графіки на платформі Android. Мобільний додаток дозволить швидко і зручно створити зображення невеликого розміру за допомогою різних засобів. Після створення картинки, користувач матиме можливість зберегти зображення або поділитись власною творчістю зі спільнотою.

Предметом дослідження є особливості піксельної графіки, необхідні засоби для її створення та існуючі рішення. Також розглядають технології потрібні для розробки мобільного додатку, а саме - фреймворк для розробки кросплатформних застосунків React Native та платформи Firebase.

Текстова частина складається з трьох розділів.

У першому розділі розглянута актуальність вибраної теми: аналіз предметної області та існуючих рішень. Також сформульовано постановку задачі та визначені вимоги до майбутнього застосунку.

У другому розділі розглянуто теоретичні відомості про використані технології та проаналізовано особливості розробки.

Третій розділ присвячений опису реалізації практичної частини, аналізу проблем та користувацькій інструкції.

Розділ 1. Актуальність та огляд існуючих рішень

1.1 Актуальність

Піксельна графіка (pixel art) - це відгалуження комп'ютерної графіки, особливість якої полягає у створенні зображення на рівні пікселів. Піксель - це найменша одиниця растрового зображення, зазвичай квадратної форми.[1] Вперше термін "піксель арт" був опублікований у статті Адель Голдберг та Робертом Флегал у 1982 році, але концепція існувала значно раніше. Поняття піксельної графіки тісно пов'язується з відео-іграми. Перші ігри мали дуже обмежені ресурси, малу розподільну здатність екрану, тож зображення виходили дуже обмеженими у розмірах та кольорах. З розвитком технологій з'являлось більше можливостей для відображення, з'явилась 8-бітна графіка. Один байт відповідав за один піксель, тож максимальна кількість кольорів становила 256 кольорів. З часом палітра збільшилась до 16 бітів, а наразі маємо TrueColor, що вміщує 24 біти та представляє понад шістнадцять мільйонів кольорів. Хоч необхідна потреба у піксельній графіці занепала, оскільки зникли обмеження, вона продовжує існувати і є чудовим способом створення візуального дизайну ігор, до якого звертаються інді розробники. Піксельне мистецтво не важко опанувати, тому воно також є популярним серед ілюстраторів - початківців.

1.2 Аналіз існуючих рішень

Для створення піксельної графіки потрібно зовсім небагато, насправді підійде будь-який растровий редактор, навіть стандартний Paint. Щоправда заради зручності та більшої кількості можливостей варто використовувати спеціальні редактори. Наразі є різні засоби для роботи: онлайн редактори, десктопні застосування та мобільні додатки.

Базовий функціонал, який мають всі редактори:

- можливість створення та редагування зображення шляхом попіксельної обробки (замальовування):
  - пензлик,
  - гумка,
  - лінії та фігури,
  - вибір кольору,
  - заливка тощо;
- збереження зображення: популярними форматами є PNG та GIF. А от JPG не користується популярністю, оскільки стискаючи зображення, пошкоджує його.

Додатковий функціонал, який можуть мати редактори:

- створення шарів зображення - можливість створення окремих зображень, які при накладанні одне на одне складають цілу картинку;
- створення кадрів - можливість створити кілька зображень, які послідовно змінюються, утворюючи анімацію;
- створення та/або збереження палітри - при створенні кількох пов’язаних зображень варто використовувати одні й ті самі кольори. Зручною є можливість створити палітру з зображення - кольори, які були знайдені на зображенні.

1.2.1 Онлайн редактори

Онлайн піксельні редактори - веб-сайти та платформи, які дозволяють створювати зображення в браузері. Серед них Piskel, Lospec, Pixel Art Maker та інші. Ці редактори надають базові можливості. Деякі з них дають можливість зберегти зображення на сервері, опублікувати. Проблемою онлайн-редакторів є відсутність мобільних версій.

Редактор Piskel - проект з відкритим кодом. Він надає інструменти для роботи з шарами та кадрами, може створювати палітри з зображення. Надає декілька варіантів експорту зображення: PNG, GIF та ZIP (архів з кількох кадрів). Редактор також має зручну десктопну версію для роботи без підключення до мережі Інтернет.

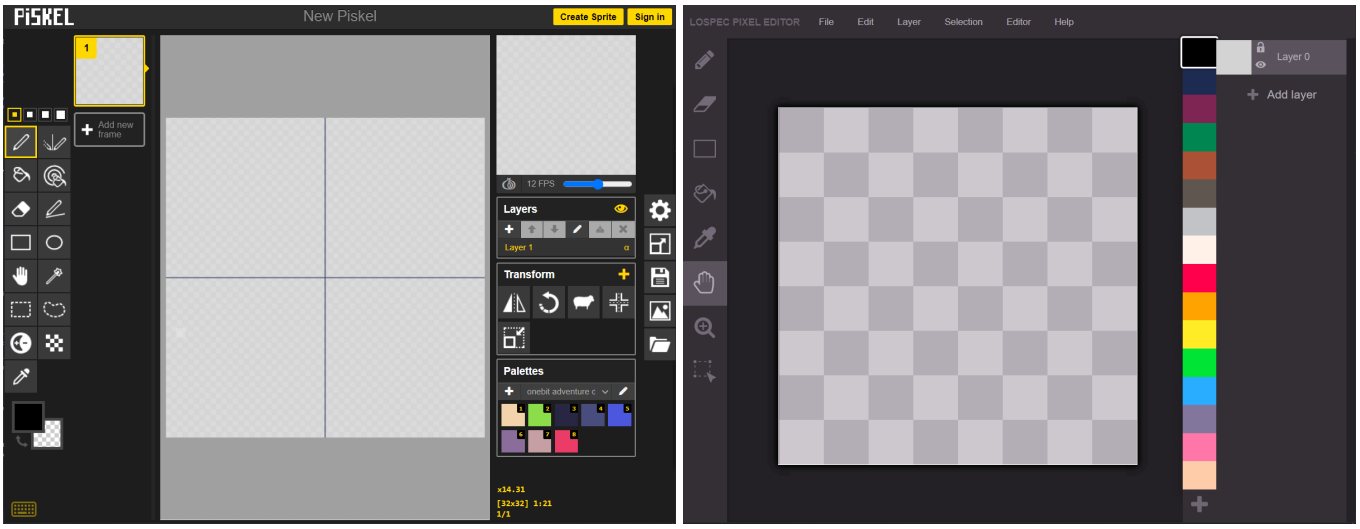


Рисунок 1.1. Користувацькі інтерфейси редакторів Piskel (а) та Lospec (б)

Редактор Lospec - частина веб-ресурсу Lospec, що пропонує багато матеріалів пов’язаних з піксельною графікою. Тут можна знайти посібники, корисні матеріали, палітри, можна надихнутись творчістю інших та спробувати власні сили безпосередньо в браузері.

1.2.2 Комп’ютерні (десктопні) редактори

Найзручнішими та найбільш прогресивними є комп’ютерні застосунки. Майже будь-який редактор можна налаштувати для зручної дрібної піксельної роботи. І все ж можна виокремити спеціалізовані піксельні редактори.

Aseprite - популярний платний застосунок з відкритим кодом. Дозволяє вільно керувати налаштуваннями зображень та анімацій. Є можливість керувати налаштуванням палітри - це може бути звичайне зображення, де кожен піксель

містить значення RGBA (червоного, зеленого, синього каналів та прозорість) або індексоване зображення, де кожен піксель містить індекс кольору в палітрі, а саму палітру можна представити списком. Цікавим також є поєднання шарів та кадрів - Aseprite містить кадри у шарі. Тобто для кількох кадрів існує один шар зображення, який навіть може залишатися незмінним протягом всієї анімації, у той час як в інших редакторах один кадр містить кілька шарів.

Pixen - це професійний платний редактор піксельних зображень для операційних систем macOS та iOS, призначений для роботи з растровими зображеннями з низькою роздільною здатністю, наприклад, з 8-бітними спрайтами. Pixen містить всі необхідні інструменти, та інтуїтивно зрозумілий інтерфейс, що включає масштабування, редагування анімації, кольорові палітри та багато іншого.[6]

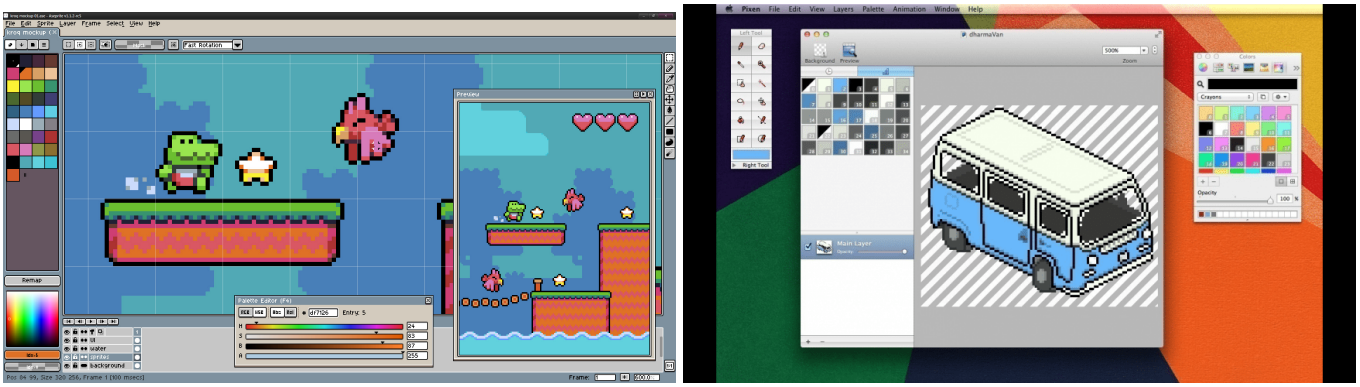


Рисунок 1.2. Користувацькі інтерфейси редакторів Aseprite(а) та Pixen (б)

1.2.3 Мобільні редактори

Незважаючи на те що мобільні застосунки обмежені площею екрану мобільних пристроїв, вони не поступаються функціоналом. Вони націлені на зручність редагування малих зображень, а також на створення соціального простору, до можна переглянути творчість інших та за бажання опублікувати власні надбання.

Pixilart - соціальна платформа для піксельних митців будь-якого віку. Основна місія - забезпечити безкоштовне програмне забезпечення для піксельного мистецтва

та навчання. Основна мета - зробити спільноту Pixilart позитивною та веселою можливістю для тих, хто хоче малювати та спілкуватися.[7] Мобільний додаток має необхідний функціонал, хоч і значно менший, ніж онлайн-версія. Значно більше можливостей для соціалізації: галереї робіт, комікси та групи створені спільнотою, конкурси, посібники тощо.

Pixel Studio - безкоштовний десктопний та мобільний застосунок для піксельної графіки з великим функціоналом, синхронізацією з Google Drive та власною соціальною мережею. Надає можливість працювати з шарами, кадрами, проте інтерфейс може бути незручним для новачків.

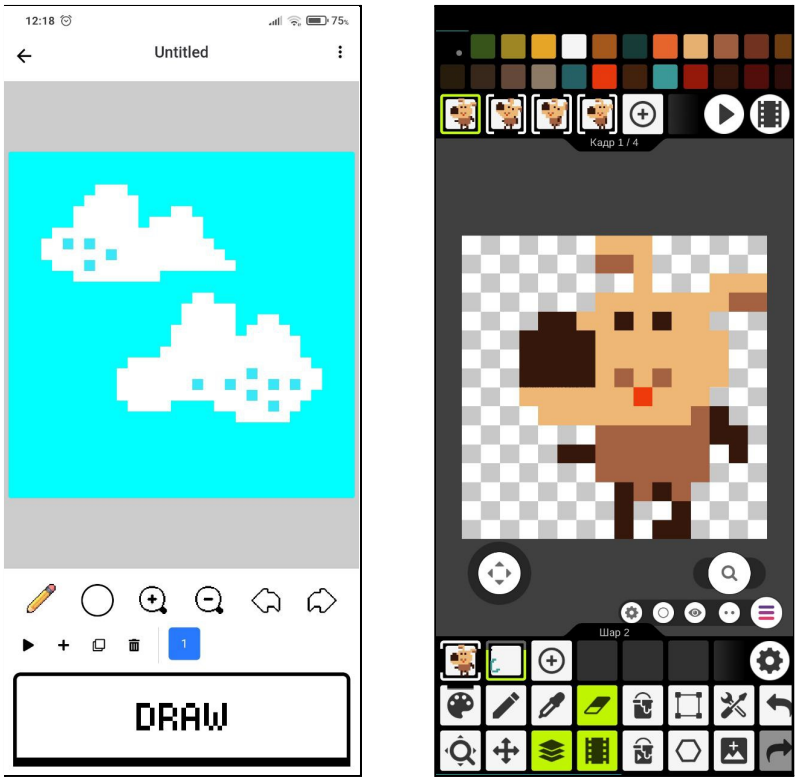


Рисунок 1.2. Користувацькі інтерфейси мобільних редакторів Pixilart (а) та Pixel Studio(б)

1.3 Постановка задачі

Мета розробки - створення мобільного застосунку на платформі Android зі зручним функціоналом для швидкого і легкого створення малих піксельних зображень.

1. Редактор має мати базовий необхідний функціонал для роботи з зображенням:
    - робоча область;
    - пензель/гумка;
    - зміна розміру, прозорості пензля;
    - вибір кольору;
    - додавання кольору в палітру.
2. Можливість створювати нові зображення, задаючи розміри зображення.
3. Можливість редагувати існуючі зображення, які можна обрати з галереї мобільного пристрою.
4. Можливість зберігати зображення у форматі PNG у локальне сховище.
5. Авторизація користувача.
6. Публікація зображення та перегляд опублікованих зображень.
7. Інтерфейс редактора повинен бути зручним та зрозумілим.

1.4 Висновки до розділу 1

У розділа була описана актуальність редакторів піксельної графіки. Також були проаналізовані існуючі аналоги на різних платформах: онлайн редактори, комп'ютерні застосунки та мобільні додатки. Створена постановка задачі та описані вимоги до мобільного редактора.

Розділ 2. Теоретичні відомості щодо технологій мобільної розробки

2.1 Аналіз особливостей розробки

Розробка мобільного застосунку - багатоетапний процес:

1. Перш за все необхідно зрозуміти ідею, основну мету створюваного додатку, тип додатку. Потрібно виокремити задачі ставлять користувачі які стоять перед застосунком та як вони їх вирішують.
2. Проектування та прототипування - це створення каркасу над ідеєю, базова візуалізація концепцій. Це створення версії додатку на папері або за допомогою спеціальних програм, наприклад Figma чи InShot. На цьому етапі створюється схематичне зображення, конкретизація вимог до функціоналу та інтерфейсу.
3. Дизайн - етап, на якому увага приділяється увага зручності користувача, кольоровим рішенням, зрозумілості інтерфейсу та загалом зовнішньому вигляду майбутнього продукту.
4. Розробка - етап створення безпосередньої функціональної частини, реалізація обговорених можливостей.
5. Тестування - цей етап перевіряє справність розробленого додатку, зручність використання на різних пристроях, швидкодію тощо. Може поєднуватись з етапом розробки.

2.2. Огляд використаних технологій та інструментів

2.2.1 React Native

Для реалізації мобільного застосунку був обраний кросплатформний фреймворк - React Native. Ця платформа призначена для мобільної розробки і

спитається на JavaScript бібліотеку React. React — це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів”.[8] React Native діє схожим чином, тільки замість веб-компонентів використовує нативні (вбудовані) компоненти системи. Це спрощує розробку і дозволяє писати одночасно для різних мобільних операційних систем, не заглиблюючись у вивчення їх особливостей.

Нативні компоненти - компоненти та їх відображення, вбудовані в систему. React Native використовує універсальні React компоненти і використовує відповідні нативні для відображення. Наприклад компонент <View>, аналог у веб-розробці, перетвориться на <ViewGroup> для системи Android та на <UIView> для iOS. React Native має великий набір таких компонентів та дозволяє створювати свої компоненти, дає контроль над зовнішнім виглядом усього додатку, надаючи синтаксис схожий на CSS.

В роботі також використовуються компоненти створені спільнотою, представленні з бібліотеках @react-native-community, react-native-elements, react-navigation тощо.

2.2.2 Expo: інструмент для роботи з React Native застосунками

Expo - це фреймворк та платформа, яка допомагає розробляти універсальні React застосунки.[10] Оснащена багатьма інструментами ця платформа спрощує розробку, дозволяючи писати застосунок одразу для кілької платформ: iOS, Android, та веб-середовища. Expo в поєднанні з мобільним додатком Expo Go - це зручний інструмент для відслідковування внесених змін за лічені секунди. Миттєві сповіщення про помилки та засоби відлагодження коду допомагають швидко виявити несправності.

Expo допомагає не тільки з розробкою, а й публікацією застосунків. Кількома командами командного рядка можна збудувати/спакувати застосунок для потрібної платформи та поділись ним. Для невеликих прикладів чи експериментів, є засіб Snack, який дозволяє писати код прямо у браузері і там же або на власному пристрої (телефоні) емулювати його роботу.

2.2.3 Бібліотека шейдерів - gl-react

Основа редактора - це робоча область, в якій власне можна малювати. У веб-розробці перевага надається елементу <canvas>, який дозволяє малювати відтворювати зображення тощо. У бібліотеці React Native такого елементу немає. І хоч існує бібліотека react-native-canvas, створена спільнотою, я обрала цікавішу для вивчення бібліотеку - gl-react.

gl-react - кросплатформна бібліотека шейдерів, яка дозволяє писати і використовувати мову GLSL. Шейдер - це набір інструкцій, які виконуються одночасно для кожного пікселя.[12] Такі програми використовуються для 3D моделювання, для відображення середовища у відео іграх і виконуються графічним процесором. GLSL - C-подібна мова для написання шейдерів. Вона має статичну строгу типізацію та засоби що полегшують роботу з векторами, матрицями - тим чим можна описати простір. Існує 2 типи шейдерів:

- Вершинні шейдери - оброблюють вершинні дані, координати, описують простір, положення камери тощо.
- Фрагментні шейдери - оброблюють фрагменти чи пікселі, визначають колір, текстуру, освітлення в залежності від місцезнаходження пікселя. Часто використовують для пост обробки.

Бібліотека gl-react дозволяє створювати React компонент <Surface>, який відображатиме виконання всіх вкладених нод, відображення яких рендериться

(обчислюється) за допомогою шейдерів. При цьому gl-react має поєднуватись з підбібліотеками gl-react-dom для React DOM, веб-застосунків, (підтримує WebGL) або gl-react-native чи gl-react-expo для React Native, (підтримує OpenGL та реалізацію Expo). gl-react містить уніфіковані засоби роботи з шейдерами, а підбібліотеки реалізують компонент <Surface>, будова якого залежить від платформи.[11]

У роботі використано поєднання gl-react та gl-react-native.

2.2.4 Firebase: бекенд як послуга

Firebase - платформа для розробки веб та мобільних застосунків, яка надає засоби для створення бекенду, як послугу. Вона полегшує розробку застосунків, оскільки може виступати у ролі бекенду чи серверу та надає можливості для майбутнього розширення. Можливості Firebase можна спробувати безкоштовно з деякими обмеженнями, а при розширенні системи користувач платить тільки за використані ресурси. Має дуже багатий функціонал:

1. Авторизація. За допомогою Firebase можна швидко і зручно впровадити різну авторизацію у застосунку: ввійти за допомогою Google, Facebook, авторизуватись через електронну пошту чи телефон, анонімна авторизація тощо.
2. Бази даних в реальному часі. Firebase дає змогу зберігати дані у хмарній NoSQL базі даних, які синхронізуються для кожного користувача застосунку незалежно від платформи. На вибір є 2 бази даних Cloud Firestore та Realtime Database, які відрізняються технологією та деякими особливостями. З використанням Firebase немає потреби у створенні серверу застосунків, оскільки клієнт може звертатись напряму до бази даних. Передбачені також системи захисту інформації, доступ обмежений правилами на читання, додавання та видалення даних залежно від ролей.

3. Хмарне сховище даних. Firebase пропонує захищене хмарне сховище, для зберігання таких користувацьких даних як зображення чи відео. Операції завантаження чи вивантаження є контрольованими - їх можна зупинити та продовжити з місця зупинки.
4. Хостинг. За допомогою Firebase можна хостити веб-застосунки чи сервіси через мережу доставки контенту (CDN).

У роботі наразі використовуються такі можливості платформи Firebase: авторизація та хмарне сховище.

2.3 Висновки до розділу 2

У другому розділі розглянуті вибрані технології, що були використані для розробки мобільного застосунку, висвітлені їх можливості та специфікації.

Розділ 3. Опис реалізації програмного продукту

3.1 Аналіз технічного завдання

Розглянуті мобільні застосунки можна поділити на 2 частини: локальна - сам редактор піксельної графіки та соціальна мережа, яка дозволяє публікувати чи передивлюватись зображення.

Для клієнтської частини потрібно розробити функціонал редактора, а саме:

- створення канви чи полотна - компоненту, який реагуватиме на рухи користувача та малюватиме зображення;
- має бути доступ до внутрішньої галереї зображень, оскільки зображення потрібно зберегти або відкрити для редагування;
- має бути зручний та зрозумілий інтерфейс: панель інструментів для обробки зображення, кольорова палітра, налаштування початкових розмірів зображення тощо.

У якості серверної частини, яка відповідатиме за збереження опублікованих зображень можна скористатись бекендом, як послугою, наприклад Firebase. Таким чином створювати виокремлену серверну частину не потрібно, оскільки клієнту достатньо додати функціонал авторизації та публікації зображення.

3.2 Архітектура додатку

Оскільки мобільний застосунок обмежений невеликою площею, контент має бути розділений і відображений на різних “сторінках” - екранах. Для зручності між цими екранами має бути навігація. Отож архітектура додатку складається з кількох екранів пов’язаних навігацією. Навігація здійснюється компонентами, наданими бібліотекою react-navigation. Всі можливі переходи користувача між екранами застосунку описані у додатку А.

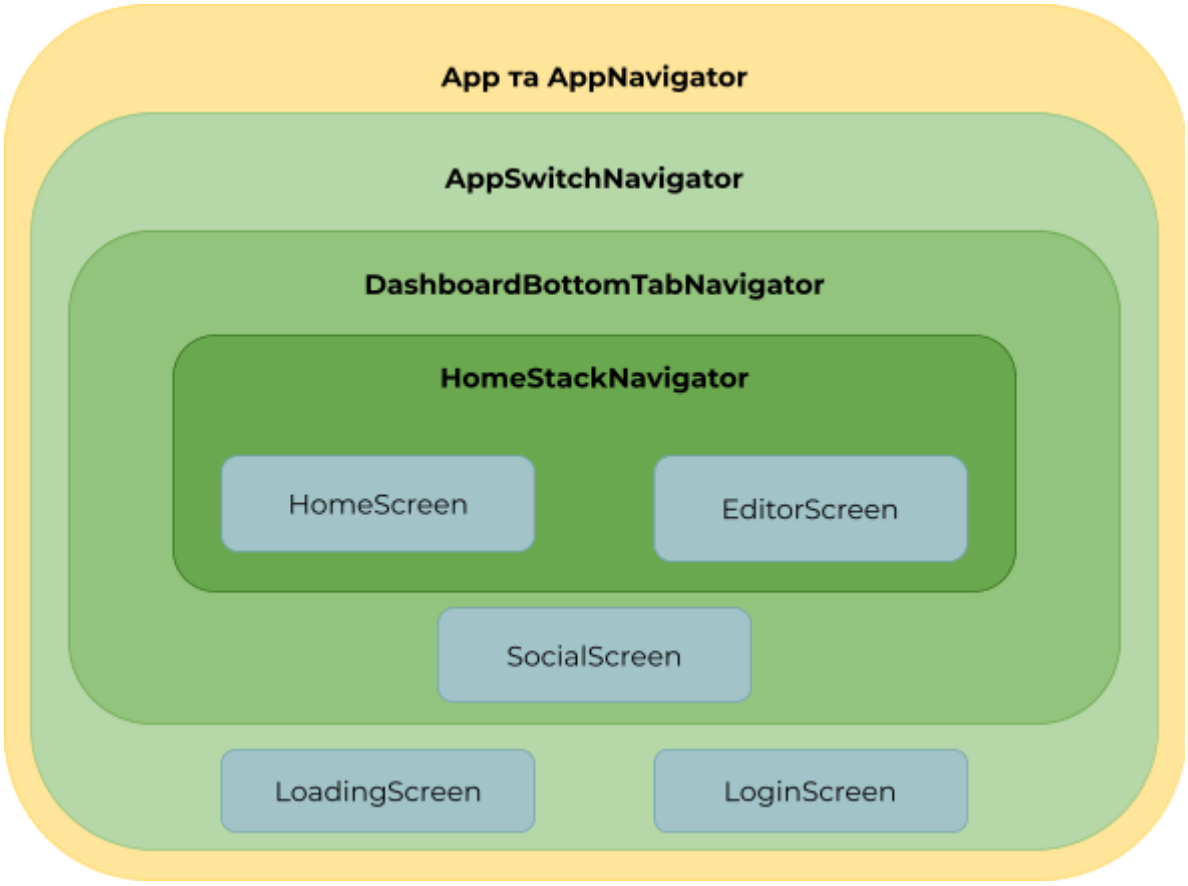


Рисунок 3.1 Структура додатку

- **App** - це компонент найвищого рівня - він є точною запуску додатку. Він містить в собі контейнер для навігації **AppNavigator**, який в свою чергу має всередині AppSwitchNavigator.
- **AppSwitchNavigator** - компонент навігації, що дозволяє переміщуватись між екранами LoadingScreen та LoginScreen та компонентом навігації DashboardBottomTabNavigator.
    - *LoadingScreen* - екран, який видно користувачу під час завантаження додатку. Якщо користувач не авторизований, його одразу перенаправляє на екран LoginScreen. В іншому випадку завантажується наступний елемент навігації - DashboardBottomTabNavigator.
    - *LoginScreen* - екран, де користувач має можливість зареєструватись за допомогою Google або анонімно.

- **DashboardBottomTabNavigator** - компонент навігації, що дозволяє переміщуватись між екраном SocialScreen та компонентом навігації HomeStackNavigator за допомогою вкладок знизу екрану. За замовчанням спрацьовує HomeStackNavigator і користувач потрапляє
    - *SocialScreen* - екран, на якому користувач має можливість переглянути опубліковані зображення.
- **HomeStackNavigator** - компонент навігації, що дозволяє переміщуватись між екранами HomeScreen та EditorScreen:
    - *HomeScreen* - початковий екран, який бачить зареєстрований користувач. Це основний екран для навігації.
    - *EditorScreen* - екран - реалізація редактора.

Особливу увагу варто приділити екрану - редактору. Він складається з різних релазованих компонентів:

- *ToolPanel* - панель з інструментами для роботи з зображенням, тут також можна викликати віконце для регулювання розміру та прозорості пензля.
- *Canvas* - поле для малювання, розбита на квадрати для зручності
- *ColorPanel* - панель, на якій можна вибрати колір з палітри за замовчуванням або додати новий колір.

Інші екрани реалізовані в більшості стандартними засобами React Native.

3.3 Опис розробки застосунку

Розробка програми умовно можна поділити на кілька частин:

1. Реалізація компонентів редактора та його функціоналу
2. Виклики до бекенду Firebase

Протягом розробки виникали непередбачувані складнощі, які потребували цікавих рішень.

3.3.1 Редактор та функціонал пов’язаний з ним

Візуально редактор складається з кількох компонентів: панелі з можливістю обрати інструмент для малювання та налаштувати його (пензлик чи гумка), панель з кольорами та місце для малювання - полотно. Усі ці компоненти реалізовані незалежно одне від одного, а для їхньої синхронізації та взаємної роботи використана методика підняття стану до батьківського компоненту - власне редактора. Підняття стану - це передача станів дочірніх компонентів батьківському і навпаки - батьківський компонент контролює стани дочірніх.[14]

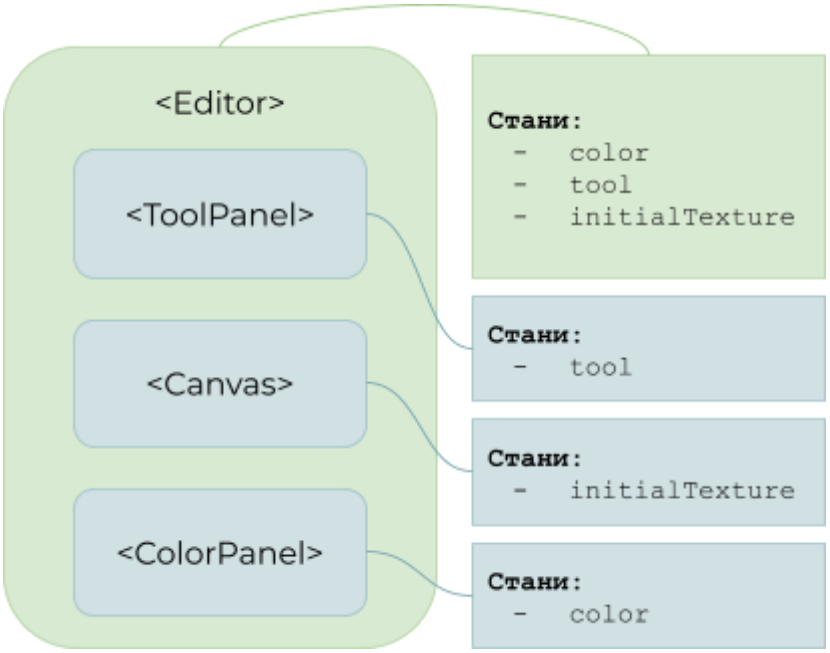


Рисунок 3.2 Структура редактору

<ToolPanel> - компонент, який відображає кілька інструментів у вигляді кнопок, кожна з який при натисканні визначає стан tool - тобто обирається інструмент. Додатковою можливістю - є кнопка налаштувань, яка показує вікно з повзунками розміру та прозорості обраного інструменту.

<ColorPanel> - діє аналогічним чином, тільки змінює стан, що відповідає за колір. Додатковою можливістю є виклик вікна, що дозволяє вибрати новий колір за

допомогою компоненту ColorPicker. Вибраний колір додається до списку наявних кольорів.

<Canvas> - складний компонент, який використовує PanResponder для відстеження рухів та компонент Surface для відображення канви. Surface - ключовий елемент бібліотеки gl-react, основна мета якого відтворити ноди які містяться всередині, які відображаються з урахуванням шейдерів. Коренева нода - PixelCanvas відповідає за першочергове створення полотна та відображення сітки, що показує границі між пікселями. Програма шейдеру:

```
        precision highp float; // встановлення точності
        // передача зміних
        varying vec2 uv;
        uniform vec4 color;
        uniform vec2 size, mouse, gridBorder, window;
        uniform float brushRadius;
        uniform sampler2D t;
        uniform bool drawing;

        // Функція, що вираховує круг з центром pos та радіусом radius
        float circle(in vec2 pos, in float radius){return step(length(pos*window), radius);}

        void main() { // соновна Функція, що буде викликана
            vec2 p = floor(uv * size) / size; // вирахування координат (ліва нижня вершина) пікселя
            vec2 remain = mod(uv * size, vec2(1.0));
            vec4 canvas = vec4(.0,.0,.0,.0); // прозоре полотно
            float m = step(remain.x, 1.0 - gridBorder.x) * step(remain.y, 1.-gridBorder.y); // вираховування сітки
            if (drawing) { // якщо є дотик, тобто відбувається малювання
                 float radius = min(window.x, window.y)*brushRadius; // вирахування радіусу пензля з урахуванням
розмірів вікна
                float circle = circle(uv-mouse, radius); // місце (круг) дотику
                vec4 c = mix(texture2D(t, uv), color, 0.6 * circle); // Піксель у зоні пензлю,
                float opacity = mix(1., c.a, m); // відображатимеьтся світлішим
                canvas+=vec4( c.rgb*m, opacity);
                vec4 circle_touch = vec4(vec3(0.3*circle), 0);
                canvas+=circle_touch; // додавання візуально коло дотику до полотна
                gl_FragColor = canvas; } // фрагмет зафарбовується відповідно до обчислень
            else {
                vec4 c = texture2D(t, uv); // якщо малювання не відбувається, то колір залишається незмінним
                gl_FragColor = vec4( m*c.rgb, mix(1.0, c.a, m));
            }}
```

Всередині PixelCanvas існують ноди, що відповідають за розмальовку під час руху користувача. Для реалізації малювання вони огоднути компонентом Bus. Bus -

це контейнер, який дозволяє кешувати змінні/текстури для дочірніх нод та перевикористовувати їх, працює як групування.[15]

Оскільки додаток надає можливість вибирати зображення для редагування, стан initialTexture, який описує зображення яке має бути на полотні, передається в редактор від батьківського компоненту - EditorScreen. Користувач має можливість створити картинку з пустого полотна, тоді initialTexture=null. Для того щоб обрати зображення з галереї, користувачу потрібно надати дозвіл доступу і тоді початкове значення initialTexture посилатиметься на обраний файл.

Для реалізації збереження зображення були використані вбудований метод .capture() елементу Surface та компонент MediaLibrary, що дозволяє асихронно завантажувати зображення у пам’ять пристрою користувача.

3.3.2 Запити до Firebase

Щоб мати можливість звертатись до Firebase необхідно зареєструвати свій застосунок у Firebase Console, задавши назву. Після цього в консолі можна вмикати різні сервіси за потреби. У додатку створено конфігураційний файл, який містить шляхи до сервісів Firebase. Бібліотека firebase вміщує реалізацію запитів до зареєстрованого додатку. Реалізованими запитами до бекенду є авторизація користувача, публікація зображення та перегляд всіх опублікованих зображень.

Авторизація налаштована за допомогою Firebase Console та реалізована у 2 варіантах: анонімно та через Google. Анонімна авторизація особливий дій не потребує, а от для Google авторизації потрібно налаштовувати API ключі, для доступу.

Публікація зображення - це вивантаження файлу до хмарного сховища Firebase. Для цього необхідно встановити шлях до бажаної папки та завантажити зображення,

перетворене на великий бінарний об’єкт (blob). Вивантаження файлу відбувається асинхронно.

Завантаження усіх опублікованих зображень - завантаження з хмарного сховище Firebase усіх файлів з бажаного каталогу. Методи firebase дозволяють отримати список усіх файлів у форматі url, але цього для відображення замало. Для відображення усіх опублікованих зображень використовується елемент FlatList, в якому створюються компоненти Image, які і завантажують та створюють картинку з наданої url.

3.3.3 Проблеми та їх вирішення

Процес розробки не простий, тому час від часу можуть викикати різні проблеми. Розробка на React Native має на увазі роботу з різними бібліотеками, та складнощами, що ховаються всередині. Як змусити бібліотеки працювати між собою, де знайти документацію, які бібліотеку ліпше обрати - ці та не тільки питання виникали під час розробки.

Однією з найбільших проблем стало завантаження зображення. По-перше полотно, на якому можна малювати значно збільшене задля зручності користувача. І тому використання методу capture() повертало реальні розміри полотна, яке бачить користувач - що було явно не тим, на що він очікував. Рішенням було створити аналогічне сховане полотно, яке відображає все те саме, що й основне, тільки без піксельної сітки та розміри цього полотна відповідають фактичним розмірам зображення, наприклад 16x16 пікселів. Це майже вирішило проблему - зображення завантажувалось маленьке, але розміри було все одно неправильні. Проблема полягала в тому, що одиниці вимірювання у стилях компонентів React Native - dp (density-independent pixels), які далеко не завжди відповідають пікселям справжнім (px). Тому полотно створювалось з розмірами 16x16 dp, що при завантаженні

створювало картинку з розмірами 44x44 px. Проблему було вирішено, нормалізацією розмірів полотна - ділення на співвідношення сторін пікселя.

3.4 Опис інтерфейсних рішень та користувацька інструкція

Неавторизований користувач при відкритті додатку побачить вікно, де може вибрати вид авторизації: увійти за допомогою Google акаунту або продовжити користуватись додатком анонімно. При виборі авторизації за допомогою Google користувача буде переспрямовано до вікна, що допоможе вибрати акаунт та завершити авторизацію.

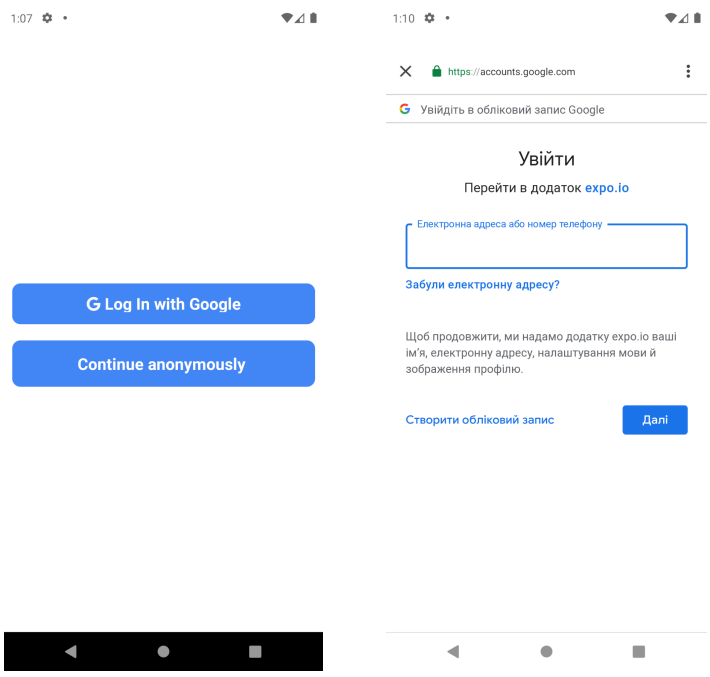


Рисунок 3.3 Вікно вибору авторизації (а), допоміжне вікно авторизації від Google (б)

Наступне вікно - вікно домашньої сторінки, де користувач може обрати, що він хоче робити: малювати, публікувати, чи вийти з додатку. Знизу є вкладки, де показується поточна (домашня сторінка) иа соціальна, де можна переглянути творчість спільноти.

Рисунок 3.4 Домашній екран з поясненнями (а), екран з опублікованими зображеннями (б)

При створенні нового зображення виникає вікно, де можна за допомогою повзунків обрати розмір полотна (від 2 до 64 пікселів у ширину та висоту) та галочка, яка свідчить про те, що полотно має бути квадратним. Якщо галочка активна, то пересування одного з повзунків впливатиме і на інший, їхнє значення буде однакове.

При відкритті зображення для редагування чи публікації користувач побачить знайомий інтерфейс вибору зображення з галереї пристрою. Якщо користувач опубліковував зображення, то він повернеться до домашньої сторінки та отримає сповіщення, що його зображення було опубліковано.

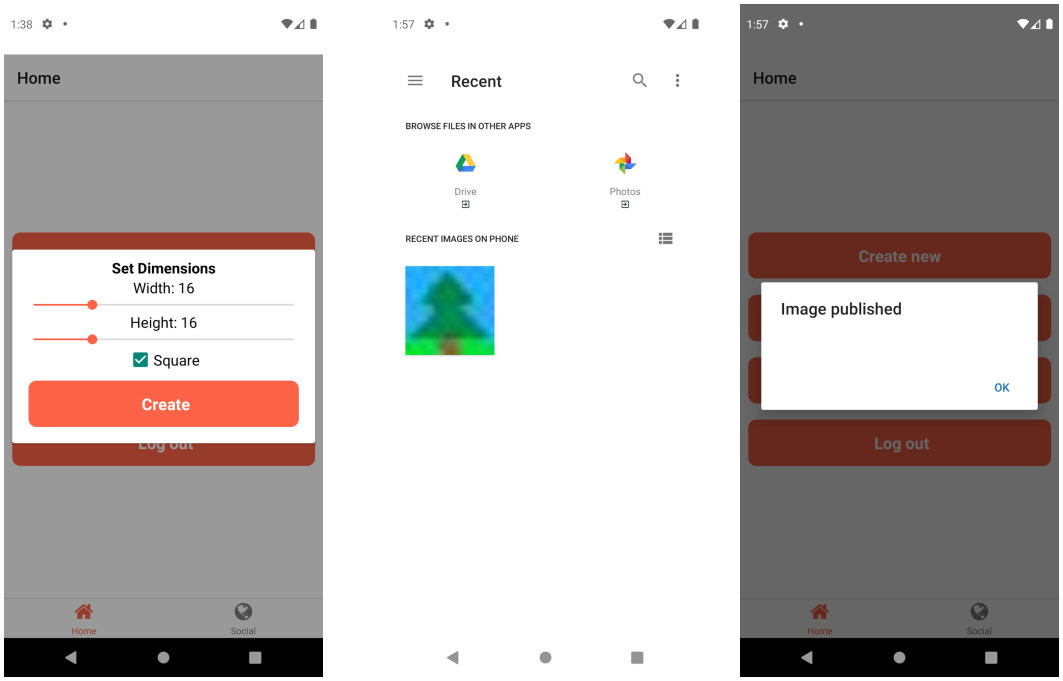


Рисунок 3.5 Налаштування при створенні нового зображення (а), вибір зображення з галереї пристрою (б), сповіщення про успішну публікацію (в)

Після обрання початково зображення шляхом налаштувань або вибору картинки, відкривається вікно редактору, де користувач може малювати на полотні, вибирати інструменти (пензлик чи гумка), налашатовувати їх. Знизу користувач може обрати колір з існуючих або додати новий за смаком. На цьому ж ерані користувач має можливість зберегти зображення, яке буде збережено у новий файл, незалежно від способу створення зображення. Також зображення можна одразу опублікувати та отримати таке саме сповіщення про успішність операції.

Рисунок 3.6 Екран редактору з поясненнями (а), зображення реалізоване засобами редактору (б)

Серед додаткових налаштувань, інструментів, які доступні користувачу - зміна розміру пензля та зміна прозорості. При натисканні на кнопку з шестернею з’являється віконечко, у якому за допомогою повзунків можна налаштувати потрібні значення, після чого натиснути кнопку ok.

При натисканні на кнопку з плюсом, біля кольорової палітри, відкривається віконечко, в якому можна налаштувати бажаний колір, який відображається у широкому прямокутнику внизу та натиснути на нього для вибору та додавання до палітри.

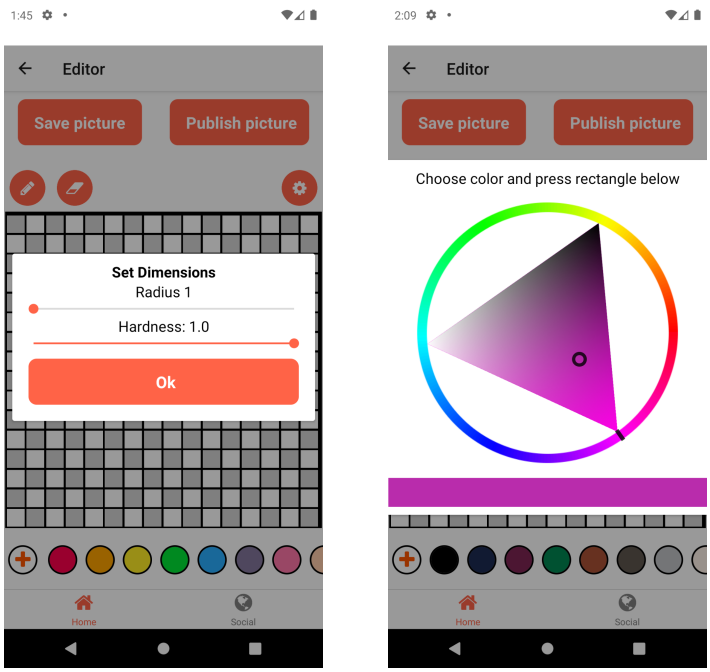


Рисунок 3.5 Додаткові налаштування інструментів (а), вибір кольору (б)

Інтерфейс був розроблений з урахуванням інтерфейсу існуючих аналогів з метою зробити додаток зручним для користувача. При створенні зображення обрані обмеження (від 2 до 64 пікселів) були образі з такою з метою.

3.5 Висновки до розділу 3

У цьому розділі було описані архітектура додатку, його реалізація. Були зазначені деякі проблеми, які виникали при розробці та шляхи їх вирішення. Також надана інструкція користувача з детальним описом доступного функціоналу.

Висновки по роботі та аналіз можливостей для подальшого розвитку застосунку

Метою даної роботи був аналіз такої предметної області як піксельна графіка. Були розглянуті існуючі аналоги та поставлена задача створення власного мобільного застосунку. Були визначені технічні та функціональні вимоги та розглянуті технології, інструменти та особливості розробки для мобільних пристроїв. Було розглянуто фреймворк для кросплатформної мобільної розробки - React Native.

Окреслене завдання зроблено - застосунок розроблено. Проте для нього є багато можливостей для подальшого розвитку. Додавання нових інструментів, створення шарів чи кадрів, розширення соціальної складової додатку тощо - це все різні можливості зробити цей додаток ліпшим.

Джерела

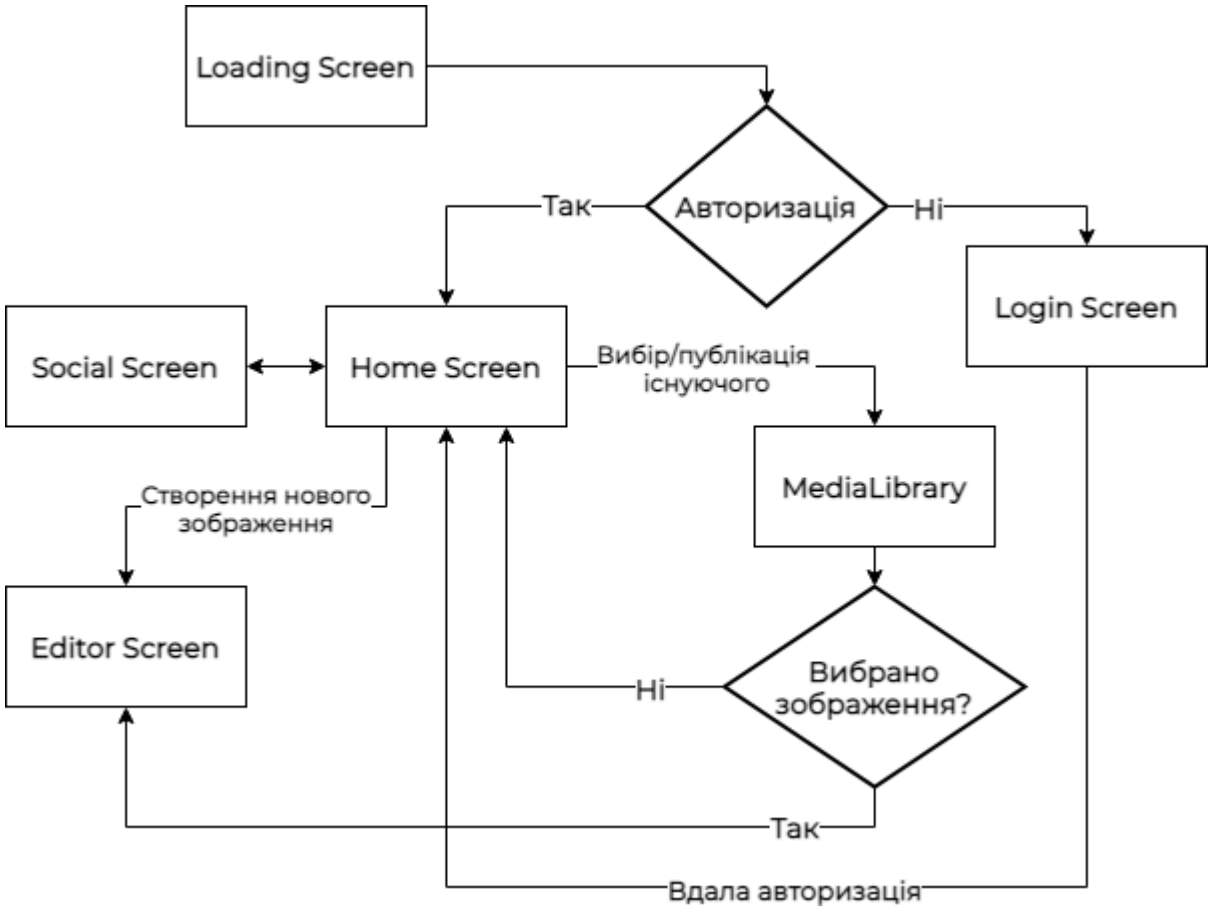
1. Pixel — Wikipedia [Електронний ресурс] – https://en.wikipedia.org/wiki/Pixel (дата публікації 08.05.2021).
2. The History Of Pixel Art — The Factory Times [Електронний ресурс] – http://www.thefactorytimes.com/factory-times/2018/9/27/the-history-of-pixel-art (дата публікації 27.09.2018).
3. Piskel - Free online sprite editor [Електронний ресурс] – https://www.piskelapp.com/.
4. Lospec - Free tools and resources for people making pixel art, voxel art and more [Електронний ресурс] – https://lospec.com/.
5. Aseprite - Animated sprite editor & pixel art tool [Електронний ресурс] – https://www.aseprite.org/.
6. Pixen - pixel art editor for macOS and iOS [Електронний ресурс] – https://pixenapp.com/.
7. Pixilart - Free Online Art Community and Pixel Art Tool [Електронний ресурс] – https://www.pixilart.com/.
8. Посібник: знайомство з React – React [Електронний ресурс] – https://uk.reactjs.org/tutorial/tutorial.html#what-is-react.
9. Learn the Basics – React Native [Електронний ресурс] – https://reactnative.dev/docs/tutorial.
10. Expo Documentation: Introduction to Expo [Електронний ресурс] – https://docs.expo.io/.
11. React library to write and compose WebGL shaders – gre/gl-react: gl-react [Електронний ресурс] – https://github.com/gre/gl-react.
12. The Book of Shaders [Електронний ресурс] / Patricio Gonzalez Vivo, Jen Lowe – 2015. – https://thebookofshaders.com/.
13. Google Firebase [Електронний ресурс] – https://firebase.google.com/.

14. Посібник: знайомство з React – React [Електронний ресурс] – https://uk.reactjs.org/tutorial/tutorial.html#lifting-state-up.
15. API – gl-react-coockbook [Електронний ресурс] – https://gl-react-cookbook.surge.sh/api#bus.

Додатки

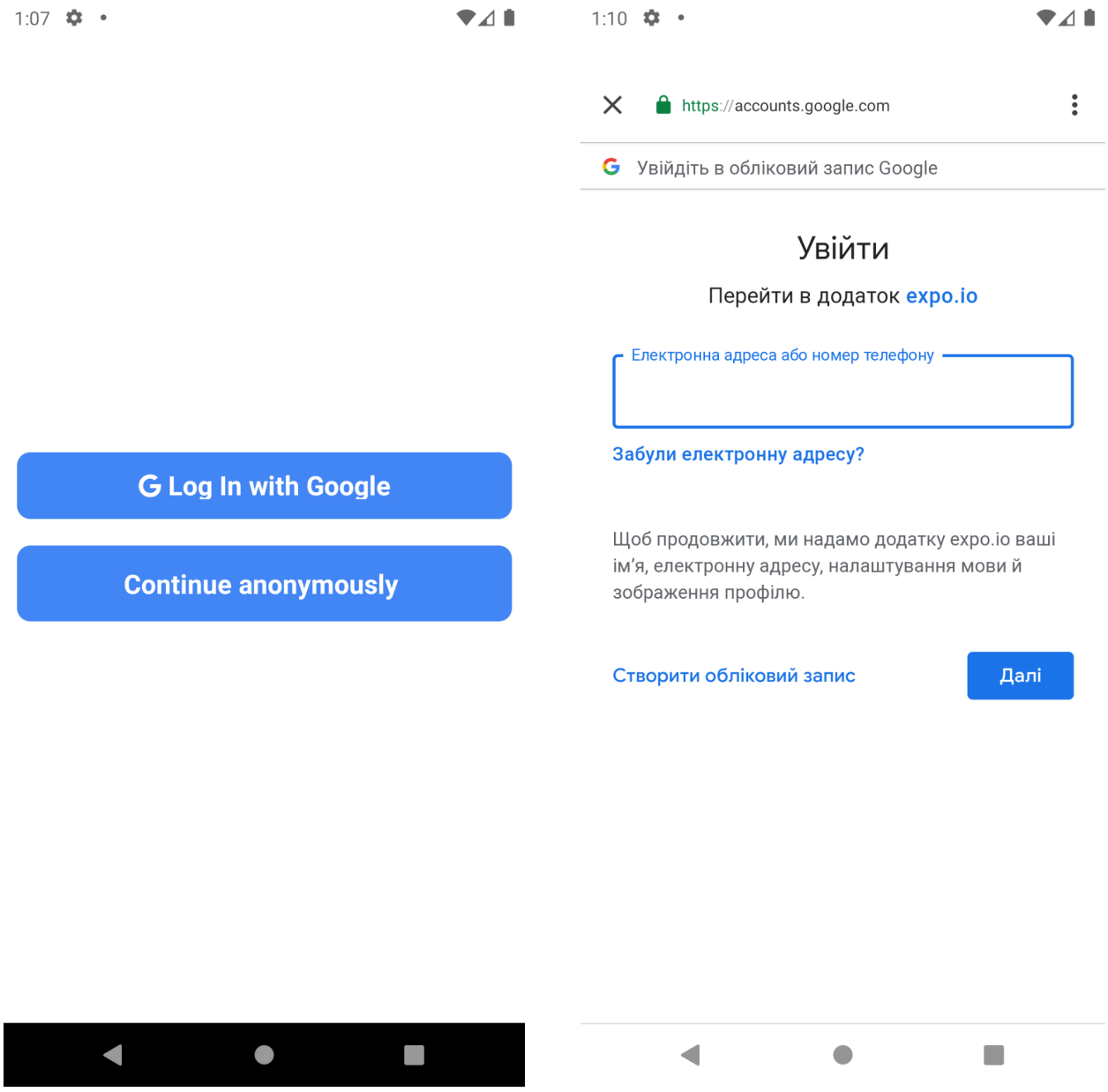
Додаток А (обов’язковий)

Карта можливих пересувань користувача



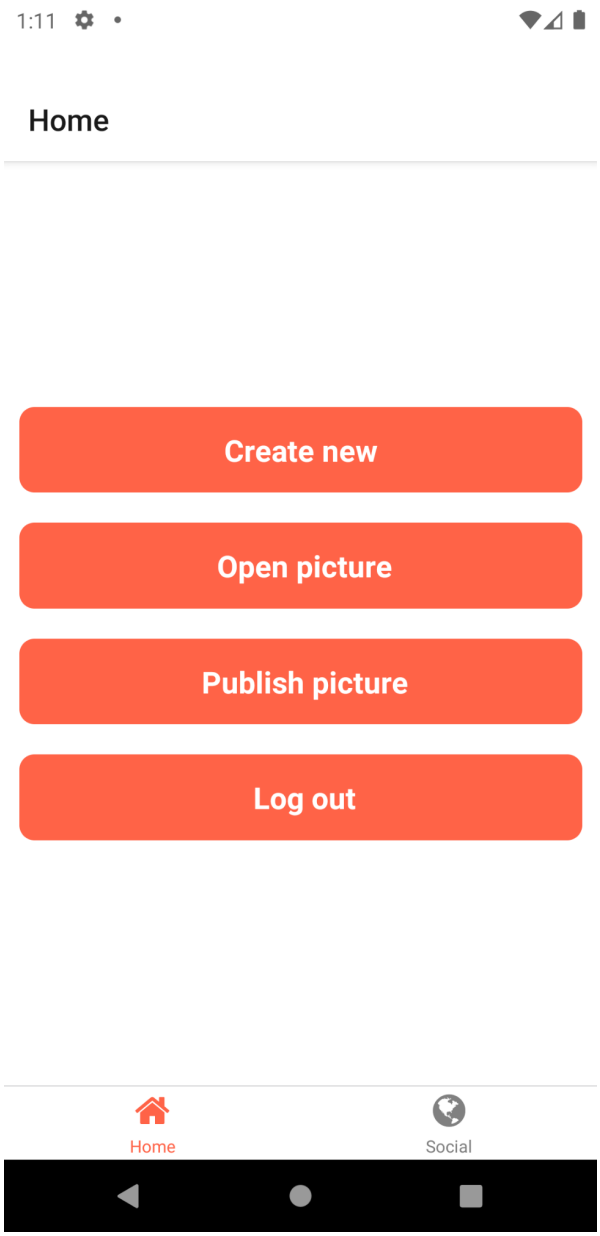
Додаток Б (довідниковий)

Екрани авторизації



Додаток В (довідниковий)

Екрани табуляції



Додаток Г

Екрани редактора