БАКАЛАВРСЬКА

РОБОТА

БР.ІП – 70.00.00.000 ПЗ

Група ІП-22-3

Корбутяк Назар

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Корбутяк Назар Тарасович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розроблення мобільного застосунку з офлайн-доступом
(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Корбутяк Н. Т.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Зікратий Серій Вікторович, к.т.н. доцент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

***ЗАТВЕРДЖУЮ*:**
Зав. кафедрою ***ІПЗ***
доц. В.В. Бандура
“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Корбутяку Назару Тарасовичу

(прізвище, ім’я, по-батькові)

1. ***Тема проекту (роботи)*** **“Розроблення мобільного застосунку з офлайн-доступом”**

керівник проекту (роботи) Зікратий Серій Вікторович, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026 р. № 179/7

2. ***Строк подання студентом проекту (роботи)*** 2026 р.

3. ***Вихідні дані до проекту (роботи)*** Результати і матеріали отримані під час проходження переддипломної практики

4. ***Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)***

1. Аналіз актуального стану досліджуваного питання
2. Збір та опрацювання даних і вимог до майбутньої програми, формування кінцевого завдання
3. Здійснення проєктування та графічного представлення системи
4. Реалізація програмного продукту
5. Тестування, виправлення помилок та публікація готового рішення

5. ***Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)***

1. Графік зміни швидкості росту інтернету 2006-2016р (рис. 1.2, ст. 15)
2. Діарама варіантів використання системи Unique Names Creator (рис. 3.1, ст. 32)
3. Діаграма послідовності процесу створення імен Unique Names Creator (рис. 3.5, ст. 35)
4. Відомості про збої Unique Names Creator (рис. 5.1, ст. 68)
5. Статистика випуску Unique Names Creator (рис. 5.5, ст. 75)

1. ***Консультанти розділів проекту (роботи)***

Розділ	***Консультант***	***Підпис, дата***	
		Завдання видав	***Завдання прийняв***

2. ***Дата видачі завдання*** _____

Керівник _____ Зікратий С. В.
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Корбутяк Н. Т.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	***Назва етапів дипломного проекту (роботи)***	***Строк виконання етапів проекту***	***Примітка***
1	Визначення та обґрунтування теми роботи	15.02.2024	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	25.02.2024	виконано
3	Побудова моделі або алгоритму власного рішення	15.03.2024	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	25.04.2024	виконано
5	Оформлення пояснювальної записки кваліфікаційної роботи	10.06.2024	виконано

Студент – дипломник Корбутяк Н. Т.
(підпис) (розшифровка підпису)

Керівник роботи Зікратий С. В.
(підпис) (розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 85 сторінок, 32 рисунків, список використаних джерел із 39 найменуваннями.

Мета роботи: розробка мобільного застосунку, що може повноцінно працювати оффлайн, без підключення до мережі будь-яким способом.

Об'єкт дослідження: процеси розробки та функціонування мобільних застосунків, робота програми повноцінно на пристрої без необхідності підключення до мережі, оптимізація використання ресурсів пристрою додатком.

Предмет дослідження: методи, технології та засоби реалізації повноцінного застосунку для мобільних пристроїв, зокрема повноцінної роботи оффлайн.

Результати дослідження: реалізовано Android-застосунок мовою програмування Java, що може повноцінно виконувати свою роботу оффлайн.

У першому розділі проаналізовано сучасний стан проблеми й тенденції створення додатків, описано та розібрано основні поняття та терміни області.

У другому розділі сформульовано вимоги, проведено збір та аналіз даних, сформовано вимоги до майбутнього продукту й виведено кінцеве завдання.

У третьому розділі здійснено проєктування системи шляхом вибору та аналізу загальної архітектурної моделі, представлено систему та її можливості графічно за допомогою UML-діаграм, проаналізовано обрані технології та інструменти розробки.

У четвертому розділі описано реалізацію проєкту, структуру коду та основні алгоритми, проведене модульне тестування.

У п’ятому розділі описано проведене тестування, здійснено публікацію застосунку в Play Store, проаналізовано отримані результати та відгуки.

Висновок: створений застосунок підтвердив можливість програмування мобільних додатків, що можуть повноцінно виконувати свою роботу оффлайн.

КЛЮЧОВІ СЛОВА: МОБІЛЬНИЙ ДОДАТОК, ANDROID, JAVA, РОБОТА ОФФЛАЙН, ГЕНЕРАЦІЯ, SHARED PREFERENCES, ФАЙЛОВЕ ЗБЕРЕЖЕННЯ, ЛОКАЛІЗАЦІЯ, РЕКЛАМНА МОНЕТИЗАЦІЯ.

ANNOTATION

This bachelor's thesis consists of 85 pages, 32 figures, a bibliography of 39 references.

Purpose of the thesis: development of a mobile application that is capable of functioning fully offline, without requiring a network connection in any form.

Object of research: the processes of development and operation of mobile applications, the functioning of an application entirely on a device without requiring a network connection, and optimization of the application’s use of device resources.

Subject of research: methods, technologies, and tools for implementing a fully functional mobile application for mobile devices with complete offline functionality.

Research results: an Android application developed in the Java programming language has been implemented, capable of fully performing its functions offline.

The first chapter analyzes the current state of the problem and modern trends in application development, describes and examines the key terminology of the field.

The second chapter formulates the requirements, presents data collection and analysis, defines the requirements for the future product, and outlines the final development objective.

The third chapter presents the system design through the selection and analysis of the general architectural model, provides a graphical representation of the system by using UML diagrams, and analyzes the selected development technologies and tools.

The fourth chapter describes the project implementation, code structure, and core algorithms, and presents the performed unit testing.

The fifth chapter describes the conducted testing, the publication of the application on Google Play Store, and analyzes the obtained results and user feedback.

Conclusion: the developed application confirmed the feasibility of programming mobile applications that are capable of fully performing their functions offline.

KEYWORDS: MOBILE APPLICATION, ANDROID, JAVA, OFFLINE FUNCTIONALITY, GENERATION, SHARED PREFERENCES, FILE STORAGE, LOCALIZATION, AD MONETIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API — інтерфейс прикладного програмування

AI/ШІ — штучний інтелект

APK — Android Package — формат файлу з розширенням .apk

CCPA — фундаментальний закон штату Каліфорнія про конфіденційність

DB/БД — база даних

GDPR — Загальний регламент Європейського Союзу про захист персональних даних

GUI/UI — графічний інтерфейс користувача

ID — ідентифікатор, унікальний номер або рядок символів

IDE — інтегроване середовище розробки

LLM — велика мовна модель

MVP — мінімально життєздатна версія продукту

SDK — Software Development Kit — повний комплект для розробки, який містить усі необхідні інструменти для створення програмного забезпечення

SOLID — п’ять базових принципів об’єктно-орієнтованого програмування

UML — Unified Modeling Language — уніфікована мова моделювання

UMP — User Messaging Platform — інструмент від Google для отримання згоди на показ персоналізованої реклами

Wi-Fi — бездротова технологія для підключення до інтернету

ООП — об’єктно-орієнтоване програмування

ОС — операційна система

ВСТУП

Сучасний світ стрімко рухається до діджиталізації у всіх сферах життя людини від навчання і до різного типу роботи, залишаючи позаду все та всіх, хто за ним не встигає. Щомісячно випускаються тисячі додатків для різноманітних цілей та аудиторії і при цьому усі поголовно рухаються у бік використання готових сервісів, реально створюючи лише оболонку, що стукається до віддаленого API з будь-якого запиту. Це все, звісно, потребує підключення до мережі та завантаження великого обсягу даних. І здавалось би інтернет зараз є усюди і це не має стати проблемою, але це далеко не так.

В умовах війни, різних терористичних та природніх катаклізмів з'єднання може зникнути будь-якої миті. І це ми вже не згадуємо про частини нашої планети, де мережа досі не є прокладеною, або дуже нестабільною. Як можна користуватися такими сервісами без неї? Відповідь є дуже простою – ніяк, просто неможливо, в таких умовах вони стають не більше ніж просто зайнятим місцем на пристрої. То навіщо тоді взагалі потрібні такі застосунки? Вони звісно забезпечують рух прогресу та науки, але що ж робити у критичних ситуаціях залишається досі не зрозумілим. Чомусь раптом усі вирішили, що пріоритетом у цьому світі є погоня за технологіями, при якій стабільність, зручність та надійність відкладаються на другий план. І ми не агітуємо зараз відкинути увесь прогрес людства та повернутися до печерного або кочівного життя, скоріше просто нагадати, щоб в пориві за технологіями, тією химерною крутістю ми, як розробники не забували, що основною метою нашої роботи є продукт, комфортний для користувача, такий, що можна використовувати у різних ситуаціях і він ніколи не підведе, чи то без інтернету, чи то без світла, чи в інших критичних ситуаціях.

Актуальність теми цієї роботи зумовлена стрімким поширенням мобільних пристроїв та постійним зростанням попиту на мобільні застосунки, що забезпечують швидкий доступ до потрібного функціоналу без складного налаштування та без обов'язкового підключення до мережі Інтернет, у час коли користувачі все частіше надають перевагу автономним рішенням, які можуть

працювати незалежно від якості мережевого з’єднання та не потребують створення облікового запису, гарантуючи доступність функцій у будь-який момент.

Об’єктом дослідження є процеси розробки мобільних застосунків на платформі Android, організація їх автономної роботи без використання мережевого підключення.

Предметом дослідження виступають методи, технології та програмні засоби зі створення мобільних застосунків із повноцінним оффлайн-функціоналом.

Метою роботи є реалізація мобільного застосунку, що може повноцінно виконувати свої функції оффлайн, без будь-якого підключення до мережі.

Завданням роботи виступає здійснення повного циклу розробки програмного продукту, аж до його випуску, включаючи аналіз сучасного стану мережі та галузі з опрацюванням відповідних джерел, аналіз отриманих даних та формулювання функціональних і нефункціональних вимог, проєктування архітектури системи та розробку самого рішення, проведення тестів та процесу рефакторингу, аналіз отриманих результатів, включаючи попит, оцінку та відгуки користувачів.

Методи досліджень, що були застосовані при виконанні роботи включають в себе аналіз літературних та технічних джерел та існуючих рішень вирішення проблеми, проєктування архітектури із використанням графічної візуалізації, дослідження досвіду тестувальників та коиистувачів при взаємодії з продуктом.

Новизна отриманих результатів полягає у створенні та дослідженні роботи специфічного алгоритму, що працює виключно на пристрої користувача і використовує символи та комбінації для створення різноманітних унікальних результатів.

Практичне значення отриманих результатів полягає у дослідженні взаємодії користувача із графічним інтерфейсом застосунку й основним алгоритмом, що створює можливості для покращення користувацького досвіду у

майбутніх випусках застосунку. Також створюються можливості дослідження роботи основного розробленого коду, взаємодії контрольованого рандому з символами та їх комбінаціями алфавітів зовсім різних мов. Зміна параметрів такого алгоритму може значно впливати на отримані результати, а тому є критично важливим його дослідження до здійснення нових випусків для уникнення зниження користувацького досвіду та рейтингу додатку відповідно.

У результаті здійсненої роботи було вдало створено робочу версію застосунку Unique Names Creator з її публікацією у Google Play Store з монетизацією через рекламні банери від Google AdMob та отримано позитивні відгуки від тестувальників та користувачів.

Цей проєкт ще має багато можливостей для розвитку через створення окремого меню для вибору типів генерацій від імен та електронних пошт і до паролей, прізвищ, найменувань міст та компаній. Збільшення його аудиторії через розповсюдження рекламних відео та іншими можливими методами збільшення користувачів і з майбутніми перспективами принесення доходу.

Бакалаврська робота містить 100 сторінок, 29 рисунків, список використаних джерел із 39 найменуваннями.

РОЗДІЛ 1. АКТУАЛЬНИЙ СТАН ПРОБЛЕМИ

1.1 Сучасний стан проблеми

З розвитком технологій розвивається і потреба людей у постійному доступі до мережі. Створюються все складніші системи і пристрої, як наприклад нейрокомп'ютери, центри обробки даних, сервери та інше, програмне забезпечення до яких потребує неабияких потужностей і не може бути запущеним чи то навіть завантаженим через брак ресурсів пристрою.

Чи можливо вирішити таку проблему значно наростивши потенціал, скажімо телефонів чи ноутбуків, пристроїв, що призначені для мобільного використання? Якби мова йшла виключно про персональні комп'ютери відповідь може бути простою - величезний б'юджет і приміщення. Але не кожен може собі таке дозволити, та й до того чи є хоч у когось бажання бути прикутим до такої-собі лабораторії? Відповідь буде різнитися від людини до людини, але ми мусимо враховувати і бажання тих, хто хто хоче жити більш активним життям. Скажімо пересуваючись у трейлері по країні й виконуючи роботу віддалено, за допомогою того ж таки ноутбука, який можна взяти із собою всюди, або навіть більш мобільного пристрою.

На рисунку 1.1 - Desktop vs Mobile vs Tablet Market Share Worldwide - March 2026 зображена діаграма, що показує відсоток використання десктопних комп'ютерів, мобільних пристроїв та планшетів для доступу до інтернету у світі за період березня 2025 - березня 2026 року, взята із сервісу statcounter [1]. На ній чітко видно, що у світі для доступу до інтернету люди частіше використовують мобільні пристрої, аніж десктопні, а саме 55 та 43.31 відсотки відповідно, а також 1.69% планшетів, що тільки збільшує кількість відсотків мобільних пристроїв. Якщо ми маємо таку статистику саме для користувачів інтернетом, то при використанні оффлайн кількість використовуваних мобільних пристроїв порівняно із персональними комп'ютерами тільки зросте.

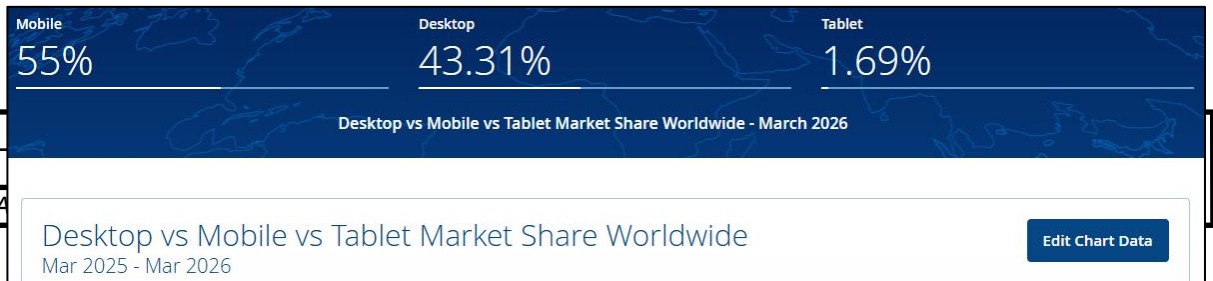

Рисунок 1.1 - Desktop vs Mobile vs Tablet Market Share Worldwide - March 2026

Скільки разів на день ви використовуєте свій смартфон бодай для перегляду години? Набагато частіше аніж персональний комп’ютер. А для використання калькулятора, нотаток чи календаря, є ще й будильники врешті решт. Тим не менш є дуже багато сервісів використання яких можливе тільки із доступом до мережі. Візьмемо навіть LLM, що не так давно стали мега популярними і починають навіть замінювати всім звичні пошуковики. І ніяк інакше до таких сервісів доступу не отримати. Смартфони ще не доросли до того рівня щоб могти мати вбудовану LLM, а тим більше таку, що зможе працювати в умовах відсутності інтернету.

І хотілося б сказати, що ми живемо у сучасному світі, проблем із мережею немає, вона є всюди! Але це не так. Багато маленьких сіл, а також місця де немає населених пунктів можуть мати дуже слабку мережу, або не мати її взагалі. Так само, як вона може бути відсутня і у більших містах менш розвинених країн. Окрім того ніхто не відміняв можливі втрати мережі. Причин для цього є дуже багато, як проблеми із боку провайдера так і користувачів, а можливо й погодні умови пошкодили якийсь кабель. Так наприклад інтернет може зникнути не

тільки у вашій квартирі чи будинку, а й у цілому місті чи державі. І не завжди є можливість відновити його в той же день чи хоча б тиждень.

Причинами проблем можуть бути [2]:

- Перевантаження мережі.
- Аварії чи ремонтні роботи.
- Використання застарілої інфраструктури.
- Погодні умови.
- Віруси або інші шкідливі програми.
- Пошкодження кабелів.
- Та інше.

Також варто згадати, що доступ до інтернету не є безкоштовним чи дешевим. Звісно можна користуватися публічно доступними точками роздачі Wi-Fi, але вони не є безпечними і можуть без вашого відома встановити вам шкідливе програмне забезпечення. Тому люди зазвичай використовують власну домашню або робочу бездротову мережу, а також мобільні дані, надані компаніями зв’язку через карточки. Зменшення його використання допоможе не аби-як зекономити. Є цілі статті присвячені заощадженню трафіку, де пропонуються різні варіанти виходу з ситуації від вимкнення оновлень додатків та зменшення трафіку стрімінгів аж до налаштувань соціальних мереж [3].

Людство звикло вирішувати проблеми за мірою надходження, питання тільки скільки часу і зусиль, спроб і помилок потрібно на це витратити. Проблема доступу до інтернету не є для нас новою. Зважаючи до того ж і на те, що він з’явився ще порівняно недавно, якщо дивитися на повну історію нашого виду аж до сьогодення. Та технологія усе більше й більше захоплює планету і в недалекому майбутньому, експерти кажуть, що не раніше 2050 року, його будуть мати якщо не усі, то більшість планети точно. Станом на зараз це всього половина планети. На рисунку 1.2 - графік зміни швидкості росту інтернету 2006-2016р. видно, що з 2006 року приріст людей, у яких з’являвся інтернет різко знизилися [4].

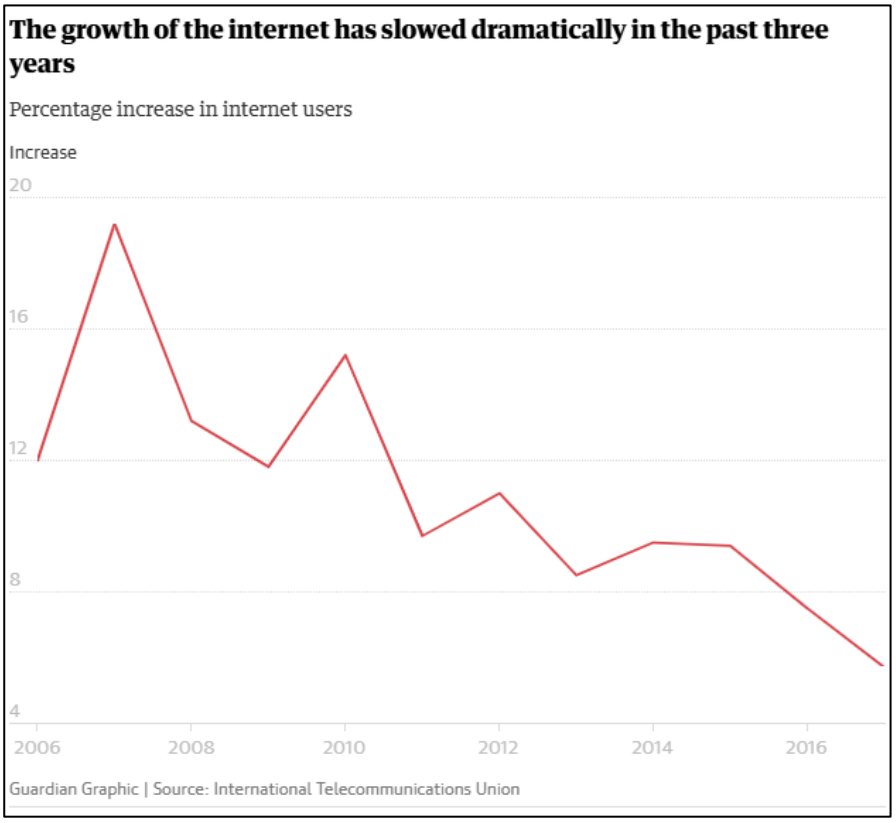

Рисунок 1.2 - Графік зміни швидкості росту інтернету 2006-2016р.

Це сталося тому, що більшість людей які могли отримати його вже тоді це зробили, інші ж не мають такої можливості з різних причин і через вартість проведення у цілі села і через неефективність використання і через незнання і через те, що місцеве населення просто розкрадало ново-встановлену інфраструктуру [4]. Головне місце тут таки посідає незнання або не розуміння технології. На нашу думку якби населення було більш обізнаним воно б не чинило спротиву, а навпаки різними способами допомагало у монтажі. Але це так і є, що складно навчити чомусь групу людей без наявності бази для цього, та й грошей багато необхідно витратити. Тому компанії-постачальники воліли б скоріше не проводити інтернет у такі місця, адже це приносить тільки збитки, а тому і фінансувати подібні ідеї ніхто не буде.

1.2 Основні поняття та терміни в області

Щодо основних понять та термінів у галузі, то так як розроблюваний нами

продукт буде Android-застосунком, сфокусованим на можливостях використання його оффлайн, терміни та поняття будуть відповідними. Перш за все - це визначення поняття мережі. Інтернет (від англ. Internet) — всесвітня система взаємосполучених комп'ютерних мереж, що дозволяє зберігати, передавати інформацію та базується на Інтернет – протоколах, визначених міжнародними стандартами [5]. Мережа набула неабиякої популярності в світі за останніх чверть століття. Та ще не усюди вона є доступною і швидкість впровадження технології сильно знижений через складнощі роботи із місцевим населенням та розміщення машин на не простих ландшафтах, що підтверджується графіком на рис. 1.2.

Мобільний застосунок — програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях [6]. По суті, у самій операційній системі, Android-додатки вважаються окремими користувачами, які оперують у своїй області дії. Це можливо тому, що ОС насправді є багатокористувацькою Linux, де кожна програма вважається окремим користувачем [7]. Для пояснення система просто призначає кожному додатку власне ID користувача і встановлює права доступу до всіх файлів таким чином, що доступ надається тільки за певним ідентифікатором користувача. Кожен додаток є процесом, а процеси мають власну віртуальну машину, тому коли у вас на пристрої запускається якийсь додаток - це не просто програма, а повноцінна віртуальна машина. Коли застосунок закривається - процес просто знищується, звільняючи пам’ять для наступного додатку. Це має свої плюси як для безпеки так і для заощадження ресурсів, оскільки якщо якийсь із додатків виявиться вірусом, то він не зможе заразити нічого більше окрім дозволених файлів власної віртуальної машини, тобто ваші особисті дані в інших програмах знаходяться у відносній безпеці. Така будова ОС також допомагає заощаджувати ресурси пристрою, оскільки на одну віртуальну машину виділяється певна кількість оперативної пам’яті та ресурсів процесору, а отже одна програма не може стати причиною вимкнення пристрою через перевантаження системи, якщо звісно цей додаток не є таким “із коробки”.

Activity(діяльність, активність) являє собою один екран з призначеним для користувача інтерфейсом [7]. По суті активності - це окремі екрани у Android-додатках, щось схоже на вкладки у браузерах, просто окрема сторінка одного застосунку зі своїм інтерфейсом та окремими функціями, має власний клас-контролер для надання окремим графічним елементам функціональних можливостей, зміни якоїсь частини графіки сторінки та зв’язки інтерфейсу із класами де описується сама логіка роботи програми.

«Android Package» або APK (також зустрічається «Android Package Kit») – це стандартний інсталяційний пакет для ОС Android. Він об'єднує скомпільований код додатка, ресурси (іконки, зображення, шрифти, звуки), маніфест з дозволами і метаданими, а також цифровий підпис, що підтверджує цілісність і походження збірки. На практиці це контейнер, який система вміє перевіряти і безпечно встановлювати (іншими словами, аналог інсталяційного файлу *.exe в Windows або *.dmg в macOS, тільки для екосистеми Android) [8]. Визначення є важливим для галузі через створення саме мобільного додатку, а отже варто розуміти і як саме відбувається збірка та встановлення додатків на пристрій.

1.3 Тренди та технології що набувають популярності у галузі

Знову ж таки обмежений доступ до інтернету не є новою проблемою, але із розвитком штучного інтелекту, а особливо популярних мовних моделей все більше і більше застосунків потребують доступ до нього для використання. Деякі тому, що справді використовують ці ж мовні моделі або бази даних чи віддалені сервери. Інші ж суто для монетизації - показу реклами навмисно обмежуючи роботу без мережі або ж навіть вимикаючи увесь додаток через навмисно створені для цього спливаючі вікна. Мови вже не йде про сайти, які ні знайти ні використати без інтернету не можливо. Таким чином щороку кількість випущених застосунків, що можуть працювати без мережі зменшується.

Виникає логічне питання - як їх використовувати не маючи доступу до мережі? І це запитання буде риторичним, бо відповідь на нього - ніяк. Такі додатки без інтернету - просто сміття, що займає місце на пристрої, обмежуючи можливість, скажімо, кількості можливих фотографій чи відео, які може створити користувач. Таким чином, згідно статистики, у Google Play Store та Apple App Store за перший квартал 2025 року 25% випущених додатків вміщали в собі AI агентів [9]. Це означає, що скоріше за все без інтернету ті додатки працювати не могли. І здавалося б це всього четвертина, ніби не так і багато, але насправді окрім цих застосунків інші теж потребують підключення до мережі, просто з інших причин.

Так ми отримуємо наступну статистику: приблизно 80-90% додатків потребують доступу до інтернету для роботи, тоді як всього 10-20% можуть працювати без нього [10].

Така тенденція існує ще й тому, що розробники женуться за новими технологіями не звертаючи уваги на потреби користувачів, адже головне це не те наскільки програма наворочена новітніми функціями, а те що вона зручна у використанні для людей і надає необхідну їм допомогу. І це не означає, що впровадження нових технологій це погано, головне щоб це не було понад міру, не впроваджувати не потрібні але круті у певних колах рішення там, де цього не потрібно.

Це все ми можемо сказати і про патерни проєктування і про різноманітні API, впровадження штучного інтелекту, різноманітні фреймворки та багато іншого. Усі вже давно забули про оптимізацію і зменшення ваги програм і фокусуються тільки на складності, на тому хто крутішу технологію впровадив і це також створює проблеми, бо багато із цих технологій також є залежними від інтернету. Але якщо подумати, то можна було б обійтися і без них.

Таким чином, якщо підсумувати, виходить зараз створюється дуже багато систем, що включають у себе Android-застосунки, які могли б і не називатися тими системами, якби не використовували технології, без яких конкретний додаток міг би і обійтися. І це все робиться виключно для маркетингу, не

зважаючи на звичайні потреби користувача.

Як сказав один відомий засновник всесвітньої компанії, пов’язаної з інформаційними технологіями, спочатку ми, як амбітні розробники дивимося на речі очима «хайпу» і «крутості», додаємо все більше і більше речей, які набувають популярності, про існування яких звичайні користувачі можуть і не здогадуватися. Пізніше ми отримуємо продукт, який або провалюється або не набирає необхідної маси користувачів. А всі відгуки тільки й говорять про те, що програма не комфортна до використання, є багато незрозумілих функцій, словом шлак. І вже пізніше через багато спроб і часу до нас, як розробників доходить, що не кількість «крутих» або «хайпових» технологій визначає те чи додаток хороший, а тільки те чи можна його зручно використовувати за призначенням. Саме таким чином набираються користувачі і тільки так можна забезпечити успіх будь-якої програми. Перш за все потрібно дивитися на потреби людей, а тоді вже на те, наскільки класно робота буде виглядати перед колегами, конкурентами та у галузі загалом. Саме за такими принципами ми і намагалися розробити Unique Names Creator. І якщо таки хотіли впровадити якусь технологію, то перш за все дивилися яким чином буде з нею взаємодіяти користувач, чи буде йому з цього якась користь і чи не заважатиме особливість йому взагалі.

Це все не означає, що ми проти використання новітніх технологій, але тільки не підтримуємо зловживання цим до рівня коли воно буде заважати користувачу. Саме тому ми маємо впроваджене API від AdMob та інше.

1.4 Висновки по розділу

У першому розділі ми розвинули нашу основну проблематику роботи. Розглянули її в цілому в сучасному світі, вияснили основні проблеми та недоліки. Тут було обґрунтовано необхідність нашого в неї втручання із власним застосунком та пояснено чому зараз усе так відбувається. Для нашої роботи, маючи тему «розробка мобільного застосунку з оффлайн-доступом» ми в

основному переглядали теперішній стан мережі та з’єднання, те наскільки воно розповсюджене у світі й що заважає це змінити, що саме робить людство для покращення ситуації. Далі ми розглянули основні терміни та поняття в області, вияснили усі не відкриті до того поняття та детально їх розглянули, зачепили певні вже здійснені для вирішення задачі рішення і основні їхні проблеми та недоліки, що потребують вирішення. Пояснили чому це важливо і що це може дати, як вплине на майбутнє галузі й чому саме зараз саме таке рішення є важливим.

Підсумовуючи викладене, можна зробити висновок, що сучасна індустрія розробки програмного забезпечення дедалі більше орієнтується на впровадження новітніх технологій, хмарних сервісів, штучного інтелекту та різноманітних мережевих рішень. Попри беззаперечні переваги такого підходу, він нерідко призводить до появи залежностей від доступу до мережі Інтернет, збільшення складності програмних продуктів та погіршення їхньої доступності для користувачів у ситуаціях, коли підключення відсутнє або обмежене.

Таким чином у даному розділі ми розглянули проблему та актуальний стан розробки програмного забезпечення у зачепленій галузі. Проаналізували різноманітні дослідження і статистики, а також дані міжнародних організацій. Виявили багато проблем сфери у сучасному світі й пояснили можливості боротьби з ними, оглянули відповідні існуючі рішення і висвітили їх недоліки у цій ситуації, навели точні визначення основних понять та термінів, що пов’язані з галуззю, які є важливими для розуміння предметної області. Також проаналізували тренди та технології, що набувають популярності у галузі й пояснили проблеми, пов’язані з ними, обґрунтували їх доцільність для нашого проєкту, проблематики та галузі в цілому.

РОЗДІЛ 2. ДАНІ ТА ВИМОГИ ДО МАЙБУТНЬОГО РІШЕННЯ, ПОСТАНОВКА ЗАДАЧІ

2.1 Аналіз та збір даних

Для розробки будь-якого рішення із будь-якої галузі спочатку необхідно провести збір та аналіз даних, побажання майбутніх користувачів, а також інші дотичні ідеї, що можуть впливати на подальше рішення. Таким чином команда розробників отримує чітке розуміння вимог майбутніх користувачів і успіх програми при її запуску на ринок значно збільшується.

При зборі та аналізі даних, першим, що необхідно згадати - це сама тема застосунку. Адже темою додатку не може бути просто програма, що може працювати оффлайн. Вона окрім цієї особливості ще повинна мати хоч якусь власну тему. Просто могти працювати без інтернету може і пуста активність. Таким чином темою самого Unique Names Creator є створення унікальних імен, що можна зрозуміти із назви. В ідею впадає генерація слів, що працює оффлайн. Це означає, що ми не можемо для цього використати штучний інтелект. Створюється необхідність реалізації якогось специфічного алгоритму, що буде виконувати основну функцію. Він використовує літери та їх комбінації для створення нових слів разом із відкаліброваним рандомом, таким чином створюючи зовсім нові найменування. Звісно він може створити і вже існуюче, та це трапляється вкрай рідко. І це насправді можна вважати проблемою, якщо не звертати уваги на те, що вона виникає виключно тому, що для своєї роботи додаток не потребує доступу до інтернету. І щоб його більше оптимізувати і він не займав багато місця на пристроях користувача було вирішено не додавати будь-які бази даних чи зв’язки з штучним інтелектом до нього, тим більше, що такі імена трапляються вкрай рідко, шанс на випадіння такого прикладу приблизно дорівнюватиме 0.0026%.

Причиною вибору такої теми самого додатку став факт роботи над Unique Names Creator ще до її початку, а тема просто під нього підійшла. Вже дуже давно ми мали бажання створити щось схоже. Перша ідея зародилася близько десяти років тому під час створення чергового персонажа для гри Mount & Blade: With Fire & Sword [11] у якій було проведено доволі багато часу. І через певну

кількість нових героїв ставало розуміння закінчення фантазії і з’явилися такі найменування персонажів як «На Шашлик», «пан Сосискін» та «містер Олень». Такі персонажі не сприймалися серйозно і гра закидувалася дуже швидко, у них не було певної мети, глобальної ідеї яку хотілося б досягнути, усе було просто для веселощів, а хотілося серйознішого сприйняття. Тоді така ж проблема була помічена і в інших людей навколо. Ба навіть зараз ми можемо спостерігати таку тенденцію у соціальних мережах, на платформах для стрімінгу та перегляду відео й багатьох інших.

Тут і починається перша важлива фаза нашого аналізу. Дослідження того як і наскільки часто люди використовують подібні імена для найменувань своїх героїв та облікових записів. Просто зайшовши до будь-якої онлайн гри або соціальної мережі, увівши в пошуку будь-яке слово англійською, можна побачити таку проблему. Користувачі мають просто однакові назви, але система не дає використовувати їх у одній базі, вони повинні бути унікальними, тому туди додаються різні цифри типу “666”, “4”, “8” та подібні. Вони можуть бути замінами певних літер, а можуть просто стояти в кінці. Тим не менш сильно знищують атмосферність і відчуття унікальності персонажа у іграх та неповторності облікового запису у соціальних мережах.

Іншим фактором є звісно побажання самих користувачів. Зважаючи на те, що ідея не є простою, а попит на ринку ми бачили на власні очі із описаних вище речей, спочатку було створено MVP версію проєкту, а тоді подано чотирнадцятом тестувальникам на пробу. Їх відгуки про основні функції застосунку були позитивними, але турбували деякі частинки інтерфейсу, які було змінено, таким чином ми змогли отримати базове розуміння нефункціональних вимог застосунку.

2.2 Функціональні вимоги

Після проведеного аналізу та опитувань, ми отримали розуміння

функціональних та не функціональних вимог, яким система повинна відповідати. Таким чином було визначено наступні функціональні вимоги.

Першою і найважливішою з усіх вимогою стане надання користувачеві відповідної кількості згенерованих імен при натисканні кнопки створення, як основної функції системи. Сюди також належить правильна робота налаштувань основного алгоритму, так як згідно них користувач повинен отримати відповідний результат. Так якщо людина обере кількість згенерованих імен двадцять - в результаті їх і повинно бути двадцять, якщо ж розмір генерованої частини середній - таким він і повинен бути в кінці. Якщо буде обрано чоловічий рід - то результат має бути відповідним. Це стосується і префіксів та закінчень та інших елементів налаштування алгоритму системи.

Наступною вимогою буде можливість зміни мови системи, що стосується не тільки власне додатку, а ще й мови генерації. Таким чином користувачі будуть отримувати можливість не тільки переглядати застосунок українською та англійською мовами, а ще й мати можливість створювати ними. Це дає не тільки зручність у використанні для користувачів із нашої держави, бо не усі програми, на жаль, підтримують українську, а й дозволяє збільшити кількість потенційних користувачів додатку на людей із інших країн.

Ще однією вимогою стадо додавання рекламних банерів у додаток для монетизації. Таким чином ми, розробляючи продукт, зможемо отримати досвід роботи із відповідними сервісами. Окрім того наша робота може принести якийсь прибуток. З противаг до цих плюсів належить нелюбов користувачів до реклами у Android-застосунках, а тому можливе зменшення охоплення.

Наступною вимогою стала наявність посилання на політику конфіденційності у додатку. В основному ця вимога була зумовлена правилами додавання застосунків у Play Store. Таким чином окрім додавання самого посилання на політику конфіденційності постає завдання ще створення самої політики і написання окремого сайту для неї, виставлення його на хостинг.

Останньою вимогою стала наявність форм про збір та продаж даних при першому заході в додаток і по відповідній кнопці для зміни думки для

користувачів із Європейського Союзу та Сполучених Штатів Америки. GDPR та CCPA відповідно. Вони необхідні за законами цих держав для додатків у яких є показ реклами. Підтвердженням виконання вимог стала діаграма компонентів, зображена на рисунку 2.1.

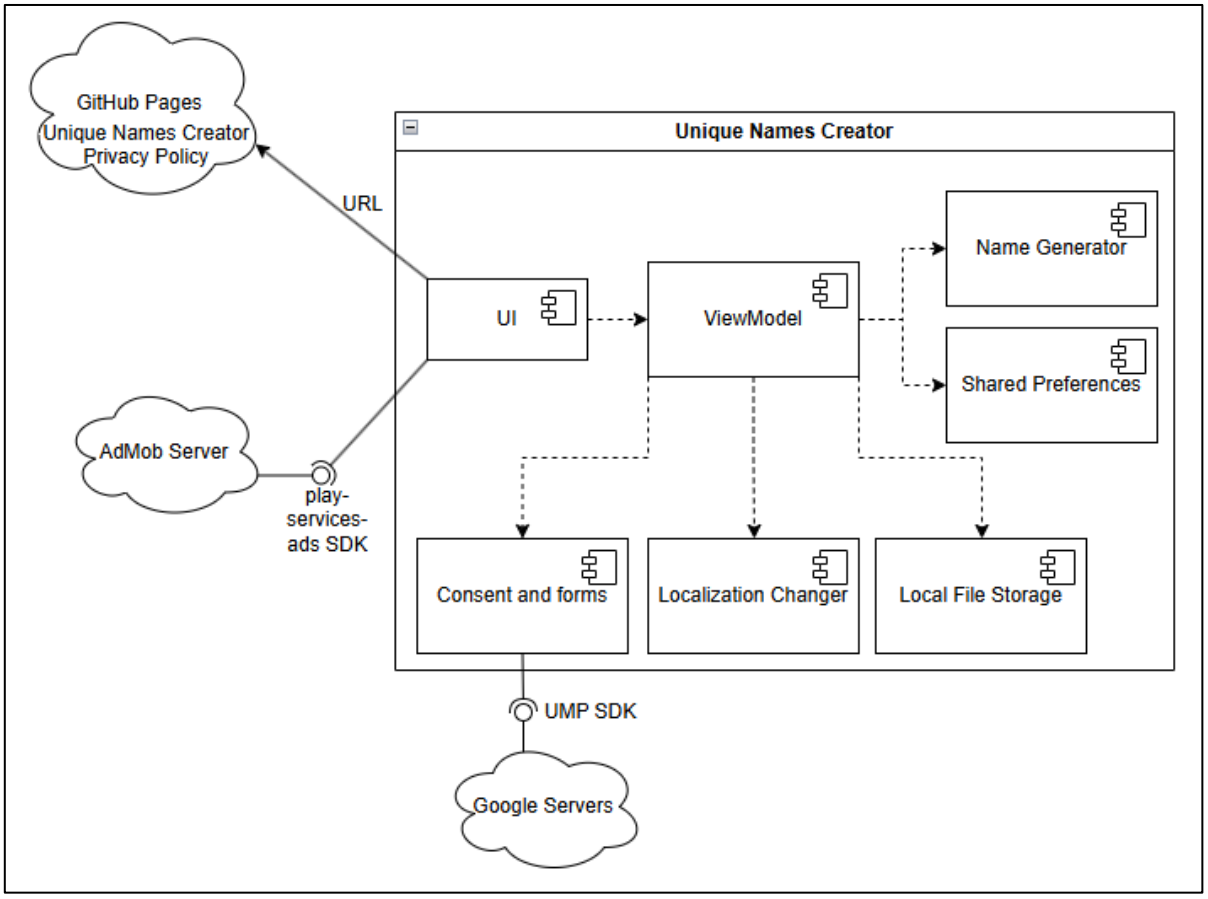

Рисунок 2.1 – Діаграма компонентів

Тут зображені всі компоненти нашої системи та її роботи. Є троє основних частин – UI, ViewModel та бізнес-логіка, що нею використовується, що представляє інші компоненти, оскільки ті до неї належать. Окрім того тут показані з'єднання системи із зовнішніми ресурсами, таким як сайт нашої політики конфіденційності на GitHub Pages, AdMob Server і Google Servers через різноманітні SDK – UMP та play-services-ads. Вони усі забезпечують виконання вимог, що описані вище і навіть додають більше функціоналу та можливостей для майбутнього розвитку системи.

Таким чином функціональними вимогами до Unique Names Creator стали

надання користувачеві послуг із генерації назв та їх налаштувань, наявність двох локалізацій та можливості зміни мови створення, наявність рекламних банерів, політики конфіденційності та форм про збір і продаж даних користувачів.

2.3 Нефункціональні вимоги

Щодо не функціональних вимог, то вони, як і попередні, є неймовірно важливими для будь-якого додатку, адже визначають зручність користування та ефективність його роботи. Нефункціональні вимоги стосуються атрибутів якості системи, які визначають, як вона працює, а не що вона робить [12].

До не функціональних вимог Unique Names Creator перш за все належить коректна й безперебійна робота без підключення до мережі інтернет. Додаток повинен повністю функціонувати без мережі.

Ще однією вимогою є час запуску застосунку. Він не повинен перевищувати трьох секунд на пристроях із середніми характеристиками. Це необхідно для того, щоб люди не переживали про те чи застосунок узагалі робочий і чи у них не завис телефон. А чим більше часу займає завантаження тим вірогідніші описані відчуття.

Третьою вимогою є час показу нових результатів користувачу. Генерація повинна виконуватися достатньо швидко, щоб не виникало запитань до успішності та наявності результату. Користувач повинен не помічати часу генерації.

Наступною вимогою є повноцінна підтримка української та англійської локалізацій застосунком. Це робиться для того, щоб більше людей могло скористатися його функціями та збільшити загальне охоплення додатку, а також дуже важливу зручність використання. При цьому перемикання між локалізаціями не повинно ламати будь-якого функціоналу програми.

Додаток повинен правильно відображатися на пристроях з різними розмірами екранів, як наприклад на планшетах та смартфонах і з великими дисплеями, так і на маленьких телефонах. Сам інтерфейс застосунку не повинен

значно змінюватися і його вигляд має відповідати знімкам екрану з демонстрацією у Play Store.

Використання ресурсів пристроїв додатком повинне бути оптимізоване і не брати забагато й непотрібного. Таким чином щоб і застосунок і пристрій могли лаконічно виконувати роботу без перегрівів та вимкнень, збоїв через нестачу пам’яті. Все повинно працювати стабільно.

Застосунок не повинен вимагати будь-якої реєстрації чи створення облікових записів користувачем. Персональні дані не повинні збиратися додатком. Адже він цього не потребує для виконання всіх необхідних функцій. При цьому дані можуть збиратися AdMob, як рекламодавцем для надання персоналізованих пропозицій.

Додаток повинен працювати стабільно, без аварійних вимкнень чи збоїв. Не перегрівати пристрій і не здійснювати йому будь-якої шкоди. Окрім того інтерфейс користувача повинен бути інтуїтивно зрозумілим для людей, які вперше його бачать. Так щоб користувач відкривши додаток або перейшовши по будь-якій кнопці вперше завжди міг визначити для чого саме цей елемент і за що він відповідає, як повернутися назад.

Додаток повинен підтримувати актуальні версії цільової операційної системи, а саме Android. Рекламні елементи застосунку не повинні перешкоджати його використанню. Налаштування користувача повинні зберігатися між сесіями у застосунку.

Таким чином ми отримали готові функціональні та нефункціональні вимоги до нашої системи. Переглянути діаграму вимог можна на рисунку 2.2. Там зображені усі вимоги до нашої системи, як не функціональні так і функціональні. Вони зручно подані у вигляді списку, тож їх можна без проблем розглянути і зрозуміти, що саме дана система повинна робити. Це зроблено без будь-яких розписувань, бо інакше вона б вийшла доволі громіздкою, тож було вирішено зробити саме так.

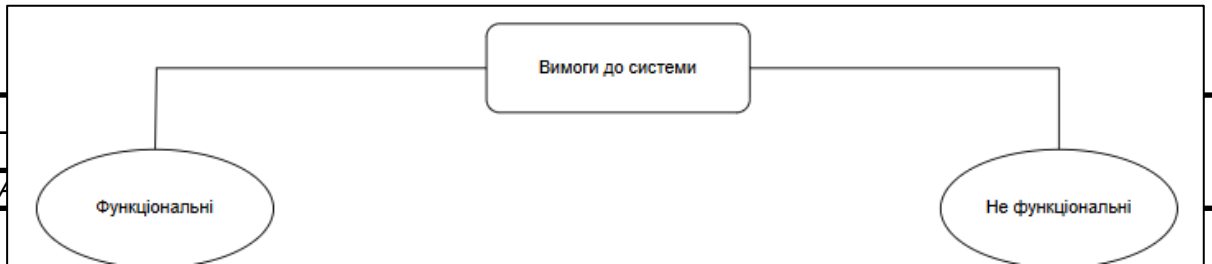

Рисунок 2.2 – Діаграма вимог Unique Names Creator

2.4 Висновки по розділу

Таким чином після аналізу та збору даних, визначення функціональних і не функціональних вимог ми можемо сформувати кінцеве завдання нашої роботи. Зважаючи на все вищеописане Unique Names Creator має стати додатком зі створення імен для користувачів. Застосунок повинен могти повністю працювати без доступу до мережі й повноцінно виконувати своє завдання.

Додатково Unique Names Creator повинен містити рекламні банери, що не заважають у користуванні, але при цьому додають йому можливість монетизації. Для них додатково у застосунку повинні міститися форми згоди для регіонів Європейського Союзу і Сполучених Штатів Америки - GDPR та CCPA відповідно. Вони повинні бути доступними виключно у регіонах, де ці форми необхідні. У інших країнах кнопку з їх відкриттям а також показ при першому

запуску можна приховати.

Застосунок повинен містити повноцінні і робочі англійську та українську локалізації. Зміна мов не має руйнувати систему, можливість прочитання чи які-небудь основні функції, як до прикладу створення додатків, рекламу, систему налаштувань та інше.

Загалом, у другому розділі нашої роботи було описано здійснений аналіз та збір даних для розробники застосунку. На основі них написано базові компоненти технічного завдання, які разом описують, що саме має робити майбутній продукт та наскільки якісно він повинен функціонувати - функціональні та нефункціональні вимоги, а також створено кінцеве завдання. Таким чином ця частина роботи допомагає нам зрозуміти як саме повинен працювати наш додаток, які функції виконувати, що саме він повинен робити і на що варто звертати увагу при його реалізації.

Тут під час аналізу було детальніше розкрито тему саме в контексті основної ідеї, а також можливості роботи оффлайн, було здійснено збір даних та дослідження ефективності й ваги думки в цілому. Щодо нефункціональних вимог, то було визначено певні критерії для зручності використання додатку користувачами, те як саме повинна працювати система, її продуктивність, безпека, масштабованість, дизайн і звісно ж сумісність. Для функціональних же було роз’яснено що саме повинен робити майбутній застосунок, як він повинен поводитись у різних ситуаціях, як реагувати на різні дії користувача. Після цього було сформовано кінцеве завдання, що включило в себе розуміння чим саме має стати Unique Names Creator, яка його основна мета і що саме він повинен робити та як. Воно позначило саму мету роботи, те як вона буде складатися в подальшому.

РОЗДІЛ 3. ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Загальна архітектурна модель проєкту

Архітектура програмного забезпечення - структура, на базі якої

створюється програма, взаємодіють модулі та компоненти всієї програми. За створення гарної архітектури програмісти взялися ще дуже давно, тому наразі нам відомо чимало архітектурних шаблонів [13].

Наразі є ряд архітектурних моделей, популярних у світі розробки. Усім відомі:

- MVC - розділяє програму на три незалежні компоненти для покращення організації коду, масштабованості та спрощення підтримки.
- MVP - розділяє бізнес-логіку та візуальне представлення для спрощення тестування та підтримки коду.
- MVVM - чітко розділяє бізнес логіку, стан програми і візуальний інтерфейс.
- Clean Architecture - розподіл системи на незалежні шари і високою здатністю до тестування, підтримки та масштабованості.
- Client–Server - розділяє завдання між двома компонентами - клієнтом та сервером.
- Microservice - будує великий застосунок через набір невеликих незалежних сервісів, що виконують свою специфічну функцію.

Загалом усі архітектурні модулі націлені на спрощення коду та його підтримки, масштабування проєкту, зменшення зв’язаності між компонентами, полегшення тестування і розуміння розробки командою. Це важливо і людство дійшло до цього не одразу, тим не менш напрямок був очевидним, маючи великі системи які було складно підтримувати ви завжди хотітимете якось їх розбити на менші підсистеми, розділити для роботи над ними окремо. Це значно спрощувало і розуміння таких систем новими працівниками і саму розробку в подальшому і пошук помилок. Людство так влаштоване, що ніхто не хоче мати щось дуже складне й не зрозуміле, адже з часом він і сам забуде що то таке було і вже ніхто тоді не знатиме, а розбиратися треба буде дуже довго. Тому щоб не тримати усе в голові й переживати як би його не забути рішення розбиття на частинки приходить саме собою.

Для розробки Unique Names Creator також необхідно було обрати якусь із

них, оскільки проєкт не є простим, це не односторінковий сайт-візитівка, а повноцінний Android-застосунок із своїми функціями та призначенням, із багатьма активностями та навіть монетизацією. Таким чином для забезпечення повного оффлайн-доступу, архітектура розробки є клієнтською і монолітною. Система реалізована у вигляді Android-додатку, що працює переважно локально на пристрої користувача без будь-якої необхідності підключення до мережі Інтернет. Практично вся логіка Unique Names Creator виконується безпосередньо на пристрої.

У проєкті використовується багатошарова архітектура з розділенням компонентів на рівні представлення, бізнес-логіки та роботи з даними. Такий підхід забезпечує кращу підтримуваність коду, спрощує тестування та дозволяє зменшити зв’язність між окремими компонентами системи. Він має свої переваги, оскільки не тільки спрощує розуміння та підтримку самого додатку, а й подальшу його розробку та ефективність, зменшуючи ресурси пристрою необхідні для виконання задач.

Обраною архітектурною моделлю стала MVVM або Model–View–ViewModel. У ній система складається із трьох основних компонентів: рівня представлення, рівня бізнес-логіки і рівень даних. На першому з них знаходяться всі основні елементи, що відповідають за взаємодію з користувачем та відображення інтерфейсу. До них належать активності, фрагменти, XML-макети та ViewModel. На другому рівні розміщується вся бізнес-логіка, себто генерації результатів, обробки параметрів унесених користувачем у налаштуваннях, правил самого додатку. Останній - рівень даних забезпечує збереження та отримання локальних даних. У Unique Names Creator використовуються SharedPreferences для збереження налаштувань та зміни мови у активностях а також мови генерації, локальні дані, необхідні додатку для роботи, внутрішні структури даних Java, як наприклад масиви чи списки і файлове сховище для збереження результатів роботи основного алгоритму та показу користувачу за запитом.

Перевагами використання такої архітектури є забезпечення роботи

оффлайн, перш за все, через рівень даних, що забезпечує збереження результатів генерації на пристрої без необхідності доступу до віддалених баз даних та хмарних сховищ. Окрім того, це ще й підвищує безпеку та ізольованість даних користувачів, що збільшує безпечність використання додатку. Ще ця архітектура є відносно простою у реалізації та підтримці, має високу швидкодію завдяки локальному виконанню та відсутність залежності від серверів таким чином зменшуючи навантаження на систему й мережу, значно спрощує тестування та налагодження.

Із недоліків ця архітектура має дещо обмежені можливості масштабування, не має можливості синхронізації між пристроями, складніша у реалізації багатокористувацьких функцій та оновлення функціоналу потребує оновлення самого застосунку. Для нашого додатку ці недоліки не є суттєвими, можна сказати що й не є недоліками впринципі, адже ми не будемо їх використовувати й у самої програми суть кардинально їм протилежна, тому можна сказати, що для нас у цій архітектурній моделі немає мінусів.

Таким чином ми обрали MVVM, як архітектурну модель для нашого додатку, вона підходить нам по концепції і не має мінусів, що хоч якось могли б впливати на Unique Names Creator. Вона забезпечує автономність його роботи і відповідність вимогам, покращує структуру коду й полегшує подальший розвиток системи.

3.2 Графічне представлення системи(UML-діаграми)

UML (Unified Modeling Language) — уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів та роботи систем [14]. Вона не є мовою програмування, а тільки списком правил для створення діаграм, для демонстрації можливостей та роботи різних типів систем.

Такі діаграм можна поділити на дві групи - структурні та поведінкові. Структурні показують те, як саме побудована система, які компоненти вона

включає, які містить класи та пакети. До цього типу діаграм належить діаграма класів, діаграма пакетів, діаграма розгортання, профілів, об’єктів, композитної структури та компонентів. Поведінкові діаграми - це вже більше схеми що показують як саме працює система при різних сценаріях і як цих сценаріїв досягти користувачу, до чого має доступ якого саме типу користувач. До цього типу належать діаграми комунікації, послідовності, взаємодії, часу, діяльності, прецендентів та інші [14].

Для нашого додатку і пояснення його системи, можливостей та роботи, ми також розробили кілька діаграм. Це діаграми послідовності, класів, варіантів використання та діяльності. Побачити їх можна на рисунках нижче.

Почнемо із діаграми варіантів використання, як найпростішої з них усіх саме для нашої системи. Вона призначена для показу типів можливих ролей та їх взаємодії із системою, так само як і їхні обмеження у вигляді дій які можуть виконувати користувачі різних ролей. На такій схемі для Unique Names Creator можна побачити, що у системі наявна лише одна роль - звичайного користувача, власника девайсу, який має доступ до усіх функцій. Це пов’язано із тим, що застосунок може працювати оффлайн повноцінно, а також із тим, що системі для повноцінної роботи не потрібно ніяких адміністраторів, користувач усе може зробити самостійно. Також на ній можна побачити всі можливі дії, що можна здійснити у додатку, до них входять: вибір мови, зміна налаштувань генерації, створення імен, перегляд результатів генерації, видалення результатів, перегляд політики конфіденційності та зміна налаштувань приватності. Діаграма варіантів використання для нашої системи представлена на рисунку 3.1.

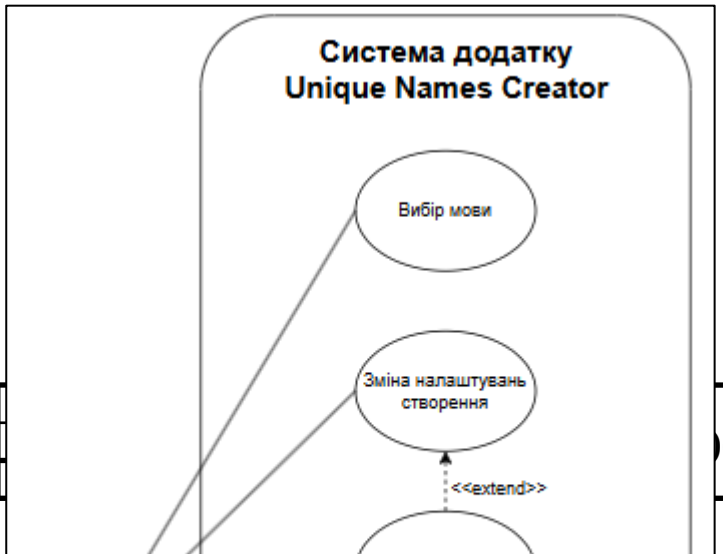

Рисунок 3.1 - Діарама варіантів використання системи Unique Names Creator

Наступною буде діаграма діяльності. Вона виявилася доволі громіздкою для нашого додатку. Схема візуалізує процес використання застосунку й ілюструє потік повідомлень від однієї дії і до іншої. Така діаграма може вважатися розширеним варіантом блок-схем, бо містить синхронізації та паралельне виконання дій. Ми вирішили детально не розписувати кожне із можливих налаштувань генерації, оскільки у світі ІТ прийнято це показувати циклом, тому так вирішили зробити і ми. Тут можна детально побачити шлях для досягнення будь-якої важливої дії в додатку, починаючи від перегляду звичайних налаштувань і закінчуючи найбільшою і найскладнішою частиною – створенням імен. Діаграму діяльності для Unique Names Creator можна побачити на рисунках 3.2 - 3.4. Якщо глянути детальніше, то можна побачити, що на першому з ниє є початкова частина входу в застосунок, логіка зміни мови додатку, а також

використання компоненту Android SDK - Shared Preferences, на наступному можна побачити більше дій вже в системі, що включають перегляд налаштувань, налаштування генерації і різницю між процесами перегляду та створення, на останньому ж показано процес самого показу результатів і те, як до нього дійти.

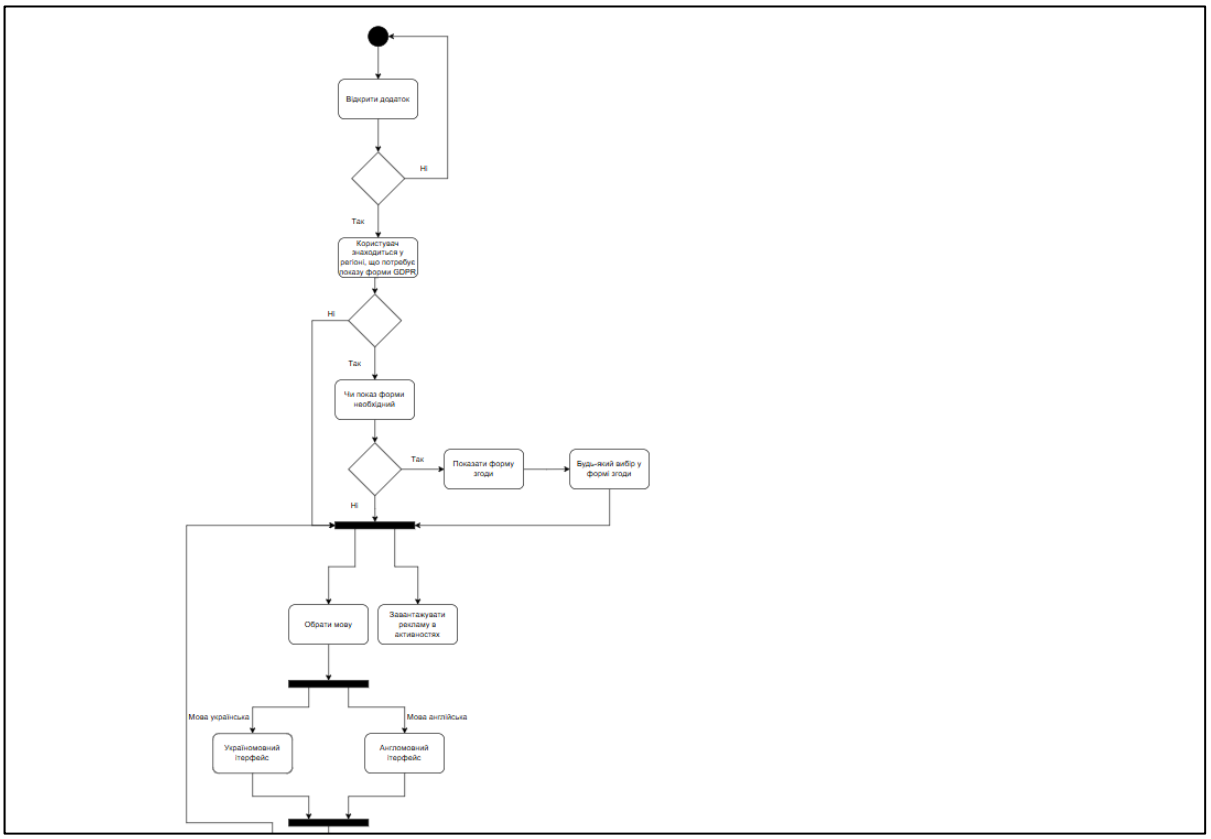

Рисунок 3.2 – Перша частина діаграми діяльності Unique Names Creator

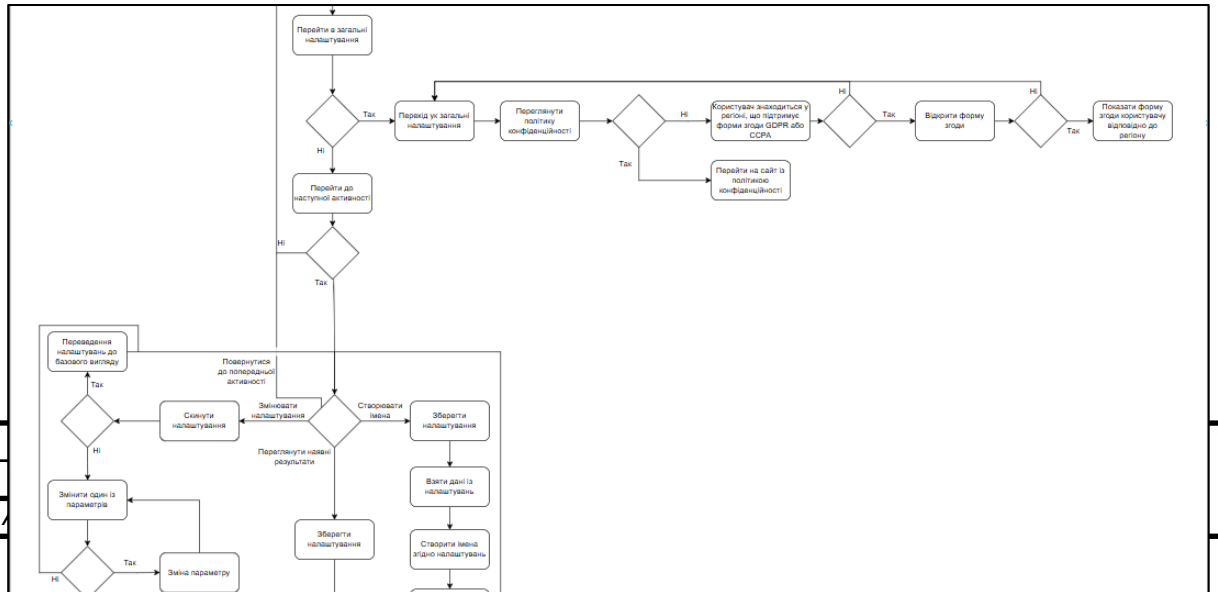

Рисунок 3.3 – Друга частина діаграми діяльності Unique Names Creator

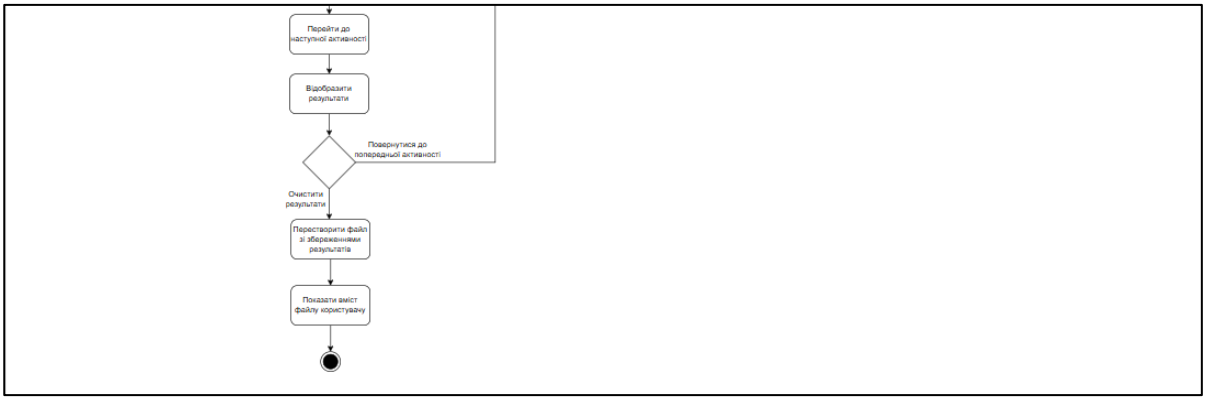

Рисунок 3.4 – Третя частина діаграми діяльності Unique Names Creator

Наступною до розгляду стане діаграма послідовності. Вона показує часові особливості передачі і прийому повідомлень об’єктами системи. При чому дії впорядковуються за послідовністю їх виконання для досягнення певної мети. Таким чином ми створили діаграму послідовності, для основного функціоналу нашого додатку, а саме - створення імен. Її можна побачити на рисунку 3.5 - Діаграма послідовності процесу створення імен Unique Names Creator.

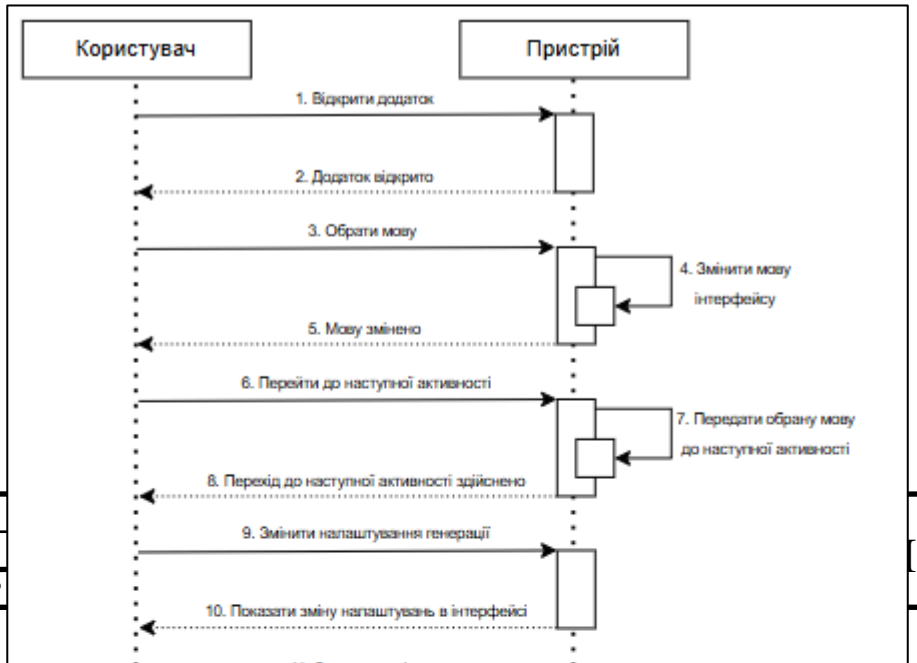

Рисунок 3.5 - Діаграма послідовності процесу створення імен Unique Names Creator.

3.3 Обрані технології та інструменти для розробки рішення

У наш час жоден програміст не може обійтися без середовища розробки. Це просто зручно. Згідно із визначенням середовище розробки - це сукупність інструментів, фреймворків, бібліотек, конфігурацій і систем, які розробники використовують для написання, створення і тестування програмного забезпечення [15].

Це просто програма, що містить у собі все. З нею можна і спростити завантаження різних бібліотек і фреймворків, бо є вже деякі вбудовані, заздалегідь знати де було допущено помилки, зручно тестувати застосунок, змінювати тему та шрифти для зручності перегляду коду, додавати різні аддони для перевірки грамотності і структури, використовувати різноманітні підказки, не прописувати методи повністю - середовище усе підтягне. Такі програми містять у собі інструменти для автоматизації складання та відлагодження

програм. Деякі мають навіть вбудовані графічні системи контролю версій, що дозволяють простіше закидувати код на той же GitHub чи GitLab. Зараз у подібні програми почали інтегровувати навіть штучний інтелект, що фокусується на аналізі коду та його написанні, що значно спрощує розробку. Сучасного програмування просто не існує без таких помічників.

Середовища розробки або IDE можуть поділятися на [16]:

- IDE для розробки систем баз даних.
- Безкоштовні IDE.
- IDE з GUI-редакторами.
- Мовні верстаки - IDE для створення власних мов програмування.
- IDE вбудовані у Linux.
- Набір IDE підкатегорії Microsoft Visual Studio [17].
- Онлайн IDE.
- Педагогічні IDE.
- IDE індустрії відеоігор.
- IDE вбудовані у Windows.
- IDE для розробки мобільних застосунків.
- IDE для конкретних платформ.

Для створення Unique Names Creator ми використовували Android Studio, версії Ledybug Feature Drop | 2024.2.2. Вона не є найновішою, але достатня для виконання необхідної роботи. Ми можемо віднести цей інструмент для розробки одразу до кількох категорій, таких як: безкоштовні IDE, IDE з GUI-редакторами та IDE для розробки мобільних застосунків.

Одним із найважливіших елементів будь-якої IDE є її інтерфейс. Адже саме через зручність користування їх і було винайдено. У Android Studio є два варіанти інтерфейсу користувача - стара і нова версії. Використовувати можна будь-яку однаково ефективно, але при цьому новіша версія ховає в собі новіший інтерфейс, який замість того щоб показувати користувачу усі можливі вкладки управління, показує тільки вікно проєкту і іконку бургера, за якою можна

тимчасово відкрити усі вкладки для використання. Це не дуже зручно, якщо ви повинні їх використовувати доволі часто, але якщо такої необхідності немає, то програма ховає увесь зайвий інтерфейс, що значно покращує фокусування на кодуванні, замість того, щоб відволікатися на непотрібний, у цей момент, інтерфейс.

Однією із основних переваг для вибору IDE була також вартість. Оскільки, як ми вже згадували, Android Studio є безкоштовною, то і вибір доволі швидко впав на неї. Ще одним критерієм була повноцінна підтримка мови програмування Java, хоча тенденції зараз показують те, що Kotlin переймає її роль у програмуванні Android.

Третьою і справді вагомою критерією стала наявність вбудованого GUI-редактора - Layout Editor [18] та можливості завантажень емуляторів через SDK. Ця особливість дозволяє розробникам переглядати й змінювати інтерфейс одразу в IDE без необхідності запуску програми, та й обирати різні розширення екрану для різних пристроїв від зовсім маленького екрану й аж до екрану ноутбуків. Це включає в себе як можливасті зміни у самому коді, атрибутів таких, як наприклад зміни за допомогою миші на графічному інтерфейсі. І це стосується навіть прив’язок до елементів. З мінусів якщо це робити не правильно, то лейаут може прикріплюватися до самого фону виключно через координати осей - x та y. Тоді зміни відображаються у самому редакторі, але не працюють для самих застосунків вже на пристрої. Просто такі більш утилітарні тимчасові рішення для перегляду проміжного результату замість повноцінного рішення на тривалий час. Вигляд такого редактора, під час роботи над Unique Names Creator можна побачити на рисунку 3.6 - вигляд інтерфейсу Android Studio.

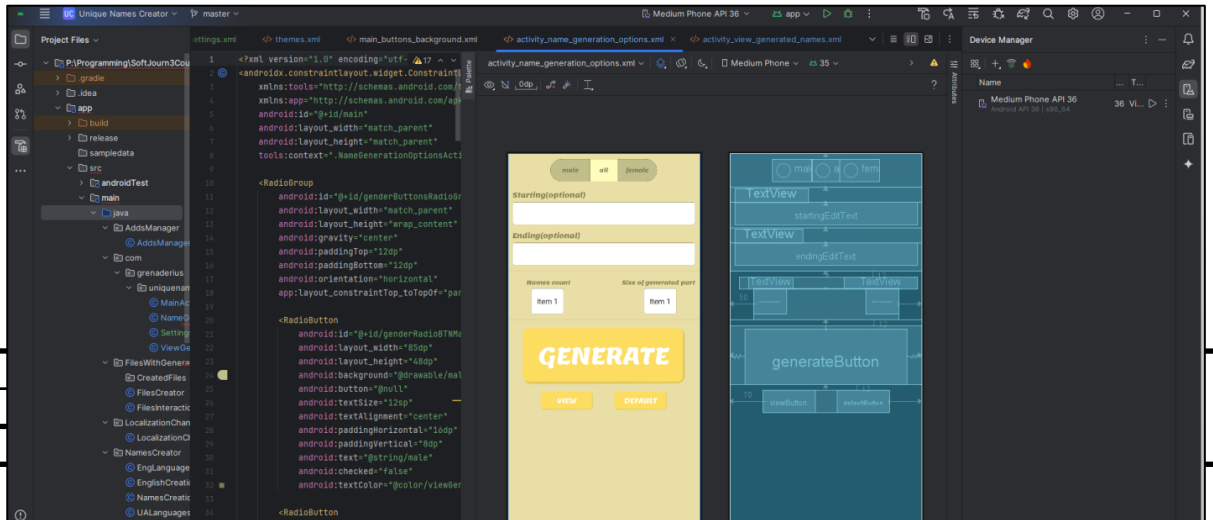

Рисунок 3.6 - Вигляд інтерфейсу Android Studio

Також на цьому фото видно, що використовується саме новий інтерфейс програми. Окрім того є три різні можливі версії роботи з інтерфейсом - виключно з кодом, виключно із графічною версією та комбінований. На фото використовується саме комбінований, оскільки він виявився найбільш зручним для одночасного перегляду результатів своєї роботи. Натомість режим редактора виключно із кодом дає можливість сфокусуватися саме на ньому і мати набагато більше простору для роботи саме з ним, а відповідно і зручності. Layout Editor також дає можливість працювати виключно із графічною версією редагування UI. Цей режим має навіть можливість додавання нових елементів через спеціальне меню, а також зміни їх розташування як почергового так і на сторінці загалом.

На рисунку 3.6 - вигляд інтерфейсу Android Studio у правій стороні зображення, можна також побачити вікно для запуску емуляторів - програм, що виконують функції, які зазвичай реалізує певний зовнішній пристрій[24], у цьому випадку - це саме смартфон. IDE дає можливість використання багатьох різних типів емуляторів від зовсім маленьких телефонів й аж до екранів ноутбуків. Програма також дає можливість вибору різних API для таких пристроїв, оскільки їх особливості залежно від версії Android сильно різняться, так наприклад функція зміни кольору функціональної панелі на пристроях після п’ятнадцятої версії операційної системи була змінена автоматичною зміною кольору під колір застосунку, тож тепер, якщо розроблювана програма має навігаційну панель відмінного від фону кольору, колір інформаційної панелі буде все-одно фоновим, що дещо руйнує дизайни деяких додатків.

Для тестування нашої програми ми також, окрім власних пристроїв, використовували й емулятори різних версій операційних систем. Таким чином було помічено та виправлено деякі недоліки у дизайні інтерфейсу на ранніх стадіях розробки, перевірено роботу певних функцій на пізніших. Останній емулятор використовуваний нами мав саме п’ятнадцяту версію Android, тому на ньому можна буде побачити і приклад із відсутністю функції зміни кольору інформаційної панелі в горі. Приклад використання такого емулятора для перевірки роботи нашої програми можна побачити на рисунку 3.7 - приклад роботи емулятора Android Studio.

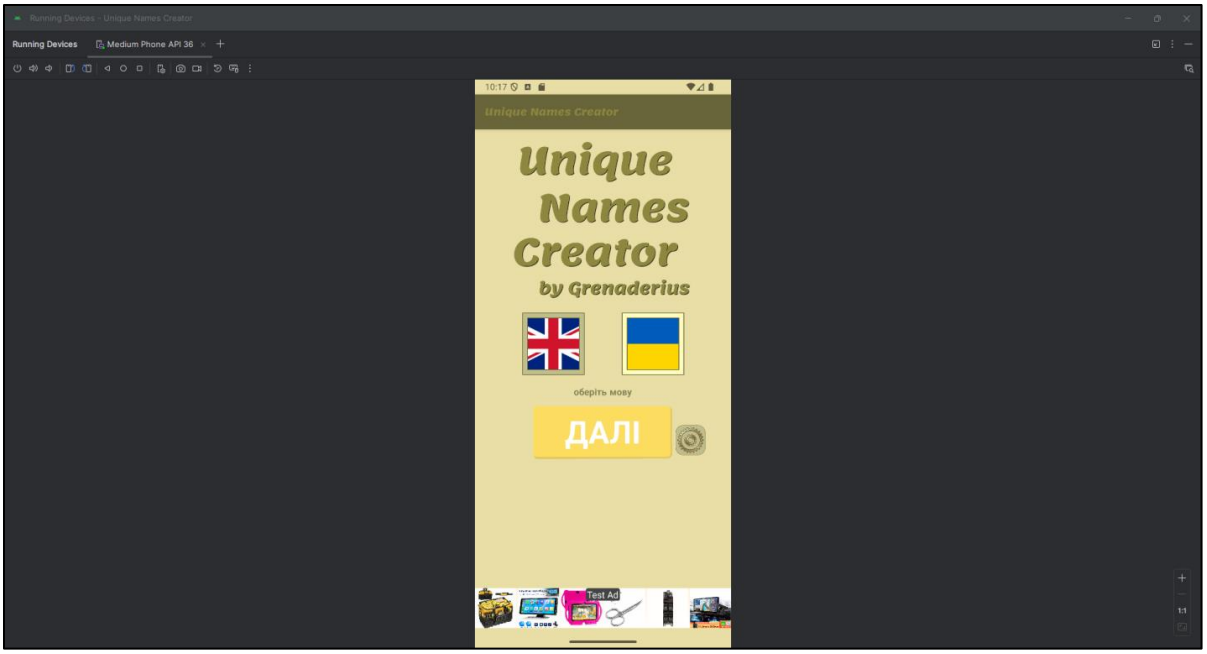

Рисунок 3.7 - Приклад роботи емулятора Android Studio

При виборі мови програмування для майбутнього проєкту ми стикнулися із двома дуже хорошими варіантами - Kotlin та Java. Перший із них є більш рекомендованим у сучасній розробці додатків і зараз усі переходять саме на нього. Причини доволі прості - це простота розуміння, мова дуже схожа на наш другий варіант, але із трохи зміненим, покращеним функціоналом. Наступною перевагою є зменшений шанс на створення багів через простішу та легшу базу коду мови [19], окрім того під час компіляції компілятор може без проблем виявити будь-які потенційні помилки. В результаті Kotlin стає більш безпечною альтернативою Java. Мова є простішою у підтримці готового функціоналу та має

підтримку дуже великої кількості IDE, включаючи описану нами вища Android Studio. Сумісність із Java - це ще один плюс, оскільки дві мови можна використовувати одночасно, тим не менш ми вирішили не користуватися такою особливістю. Має більш логічний, зрозумілий та лаконічний синтаксис і може оптимізувати загальну продуктивність команди у розробці. До недоліків Kotlin належить мала кількість розробників на світовому рівні, що зменшує можливості до навчання і можливої допомоги під час розробки програмного забезпечення. Тим не менш зараз існує дуже багато, навіть вбудованих у IDE чат-ботів та ШІ-агентів, які можуть допомогти із навчанням чи виправленням коду. Наступним недоліком є швидкість компіляції, яку Java показує набагато більшу. Ще одним з мінусів є унікальність самої мови, оскільки після вивчення її розробникам дуже складно переключитися на будь-яку іншу.

З переваг над порівняною новинкою, Java має набагато більше ком'юніті розробників, що спрощує пошук та навчання. Мова налічує дуже багато фреймворків для різних цілей, ще більшу кількість бібліотек. Сюди також можна віднести підтримку від різноманітних IDE та навіть системи для програмування виключно на Java, як наприклад IntelliJ IDEA від JetBrains.

З недоліків - це відсутність функцій розширення, що дозволяють збільшувати класи або інтерфейси новими функціональними можливостями, такими як додаткові властивості або методи, без необхідності створювати новий клас. Відсутність функцій в один рядок, підтримки делегацій, безпеки від отримання помилок типу NullPointerException та інші.

Загалом Kotlin та Java є двома дуже потужними мовами програмування, одна трохи покращує іншу, але при цьому забирає від неї деякі функції. Коли ми зіткнулися із вибором для нашого проєкту ми доволі довго вирішували яку ж із мов усе-таки використати й зупинилися на Java. Причина цьому доволі проста - ми вже маємо певний досвід розробки на цій мові, коли Kotlin довелося б вивчати з нуля і так як це стане першим нашим повноцінним Android-проєктом, що доживе до релізу ми вирішили не ризикувати й обрати більш безпечний і знайомий варіант, що і підвищить швидкість розробки і не дасть нам як

розробникам перегруз у плані новизни та необхідності вивчення нового, таким чином допустившись якоїсь помилки ми будемо знайомими хоча б із якоюсь частиною проєкту, що спростить її визначення і виправлення.

Далі обговоримо систему збереження у нашому проєкті. Це важливо оскільки для збереження інформації, а особливо згенерованого тексту ми не використовували бази даних чи якісь хмарні сховища. Все відбувається на пристрої користувача для забезпечення повного доступу оффлайн. І тут можна було б дати проєкту створювати внутрішню маленьку БД для власних потреб, але саме із бажання оптимізації ми вирішили цього не робити, а зберігати все у звичайному файлі із розширенням .txt. Таким чином ми значно зменшуємо місце, яке займає додаток на пристрої, економимо його ресурси через відсутність необхідності розгортання бази даних та даємо можливість користувачу дуже просто керувати збереженими даними із повним контролем над ними та їх наявністю загалом.

Щодо використаних API, то у роботі ми застосували декілька з них, для реклами, збереження тимчасових даних, зміни вигляду інформаційної панелі та інші. Першим із них для опису буде Shared Preferences - компонент Android SDK, що призначений для збереження невеликих обсягів даних та налаштувань користувача. По суті це механізм для збереження пар даних типу ключ - значення у окремому файлі типу XML для їх використання у різних активностях, так як дані між ними у Android не передаються. При цьому файлів можна мати багато і для різних цілей, вони також можуть бути приватними. Зазвичай використовуються для збереження обраної мови, теми інтерфейсу, налаштувань, визначення першого запуску програми для налаштування додатку відповідно, збереження стану перемикачів та радіо-кнопок, локального кешування даних, запам’ятовування виборів користувача. У Unique Names Creator API використовується для визначення першого запуску додатку та виставлення відповідних налаштувань, збереження і передачі типу мови для перекладу активностей та роботи алгоритму на певній мові, кешування і збереження даних про налаштування застосунку для передачі даних до алгоритму генерації та їх

збереження навіть при вимкненні додатку, це ж саме працює і для мов.

Ще одним використаним API є Edge-to-Edge, що дозволяє додатку відображати інтерфейс на всю площу екрану пристрою, яке використовується для коректного показу інтерфейсу на сучасних пристроях із системними панелями. Наступним є Window Insets API. Воно також використовується для інтерфейсу, а саме для правильного розташування його елементів у порівнянні із системними відступами, які є різними на різних пристроях від різноманітних виробників. Це важливо для правильного розташування самих елементів та уникнення перекриття контенту статус-баром або навігаційною панеллю. Якщо простіше - це адаптація інтерфейсу під різні типи екранів. Window API - це ще один механізм, що використовується для зміни кольору статус-бару та навігаційної панелі. Це робиться для створення повноцінного враження користувача від інтерфейсу застосунку. Наступним є ActionBar API, що застосовується для стилізації тексту, змін кольору розміру та формату, особливо у верхній навігаційній панелі, яка не належить до інтерфейсів сторінок і може бути налаштована виключно через активності. Fonts API Android також є тут використаним для підключення кастомних шрифтів та їх використання у додатку. Ми також імплементували Screen Orientation API для фіксації конкретної орієнтації екрану, а саме портретної, під яку і розроблявся застосунок. AndroidX AppCompat API - також для зміни стилів, більш комплексний, надає можливість використання багатьох бібліотек AndroidX.

Перейдемо до технологій, встановлених для монетизації та роботи реклами й пов’язаного з нею функціоналу типу форм згод та повідомлень про збір даних. Це також забезпечує дотримання вимог законодавства різних країн та їх об’єднань для показу реклами у Android-застосунках. Для цього ми використовуємо сервіси компанії Google AdMob та Google User Messaging Platform (UMP) SDK.

Google AdMob – це продукт компанії Google, що допомагає розробникам заробляти гроші на своїх додатках. AdMob добирає оголошення для додатка на основі критеріїв, які вибирає розробник. Оголошення створюють і оплачують

рекламодавці, які хочуть рекламувати свої товари та/або послуги [20]. По суті це рекламна платформа від компанії Google, що призначена для монетизації додатків через показ реклами різного типу від звичайних банерів і до реклами за нагороду, кастомізованої реклами та повноцінних відео.

Система забезпечує:

- Завантаження рекламних оголошень.
- Автоматичний підбір реклами.
- Взаємодію з рекламодавцями.
- Статистику переглядів та доходу.
- Керування рекламними блоками.

Принцип роботи AdMob полягає в:

1. Отриманні запитів від додатків до серверів.
2. Підборі відповідного оголошення.
3. Передачі реклами на завантаження у додаток.

4. У разі взаємодії користувача із рекламою фіксуються покази кліки та переходи.

Для інтеграції реклами від цієї системи у Unique Names Creator застосовується Google Mobile Ads SDK, що є офіційним SDK від Google для AdMob, Ad Manager та рекламної аналітики. Для його підключення необхідно додати відповідні залежності у файл Gradle або Maven, що імплементує бібліотеку play-services-ads, яка і містить SDK та забезпечує роботу рекламних API. Також до файлу маніфесту додається ID застосунку й користувача, що ідентифікує додаток у системі AdMob. Також необхідно до самих рекламних банерів додати їх ідентифікаційний номер, що теж надає системі розуміння що це за банер і яку рекламу можна туди дати. У нашій системі ми використовуємо чотири таких банера, по одному для кожної з активностей. Приклад вигляду такого банеру можна побачити на рисунку 3.8 - приклад активності із рекламним банером.

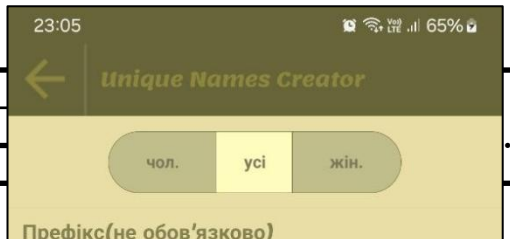

Рисунок 3.8 - Приклад активності із рекламним банером

Google User Messaging Platform SDK є інструментом, що надає розробникам можливість виконувати вимоги певних країн та їх об’єднань до показу реклами корситувачам, а саме надає інструменти для налаштування та управління формами згоди користувача GDPR [21] та CCPA [22], що використовуються для забезпечення їх приватності. Він є частиною Google Mobile Ads SDK.

Підключення відбувається також за допомогою відповідної бібліотеки - user-messaging-platform. Приклад вигляду такої кнопки у нашому додатку можна побачити на рисунку 3.9 - налаштування Unique Names Creator. Варто зауважити, що її не буде видно для користувачів у країнах де така згода не є необхідною.

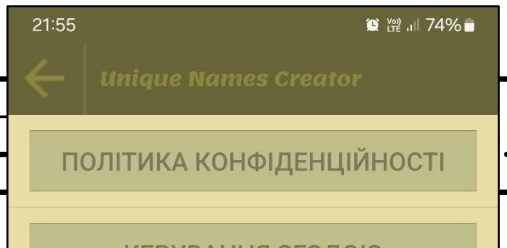

Рисунок 3.9 - Налаштування Unique Names Creator

Також через необхідність відповідати закону будо додано завантаження форми GDPR. У тому документі йдеться про те, що у користувачів із європейського союзу має бути запитаний дозвіл на обробку персональних даних, а й про те, щоб вони могли у будь-який момент змінити своє рішення за допомогою відповідної кнопки у налаштуваннях [21]. Також йде мова про те, щоб вони завжди могли від збору даних відмовитись, що можливо завдяки хрестику в горі. У нашому додатку, згідно правил, таке повідомлення надається користувачу із європейського союзу при першому вході у застосунок. Змінити вибір можна за кнопкою “Manage consent” або українською “Керування згодою” Приклад повідомлення GDPR можна побачити на рисунку 3.10 - повідомлення GDPR.

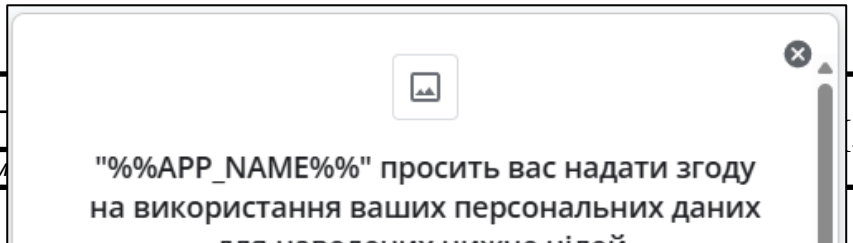

Рисунок 3.10 - Повідомлення GDPR

Також це повідомлення містить можливість налаштувати персональний показ реклами за відповідною кнопкою “Налаштування” або “Settings”. Тут можна налаштувати не тільки сам показ реклами, а й обирати постачальників із доступних фірм, а також різноманітні категорії та типи за бажанням. Окрім того є багато інших можливостей, як наприклад перегляд правил згоди, список партнерів, політику конфіденційності додатку та інше, тож по суті користувач може сам повноцінно керувати оголошеннями, що показуються йому у додатку за допомогою банерів. Приклад вигляду налаштувань зображено на рисунку 3.11 - налаштування GDPR.

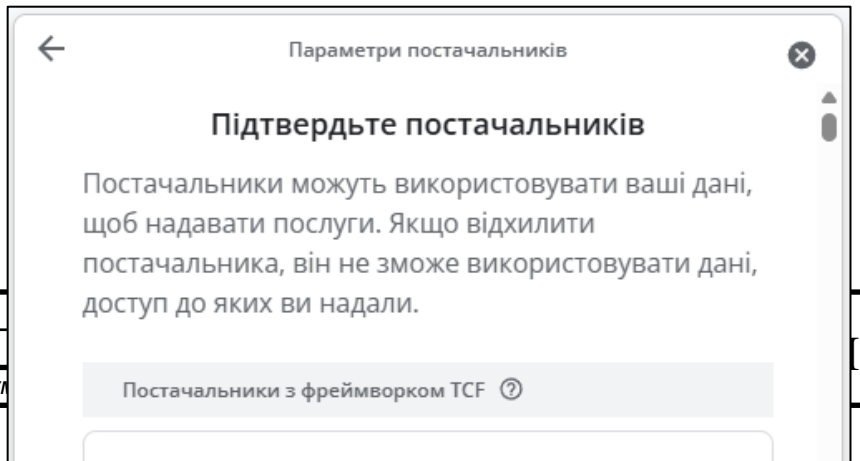

Рисунок 3.11 - Рекламні партнери GDPR

Окрім повідомлення для європейців GDPR, було також додано і форму для відмови від обробки персональних даних для мешканців Сполучених Штатів Америки - CCPA. Це закон штату Каліфорнія, затверджений першого січня 2020 року, про конфіденційність споживачів, який став першим всеосяжним законом про захист даних у США надавши жителям штату Каліфорнія нові права щодо їхньої персональної інформації та встановивши нові обов'язки для компаній, які обробляють ці дані [22]. З часом цей закон прийняло більше штатів і тепер аж 20 його підтримують. На відміну від GDPR повідомлення не повинно автоматично вискакувати при першому запуску програми, а може бути викликаним за певною кнопкою і містить у собі саме відмову, тобто вважається, що користувач уже зарання погодився на збір і обробку даних, а тепер може просто від цього відмовитися. Вигляд CCPA для нашого застосунку можна побачити на рисунку 3.12 - Повідомлення CCPA.

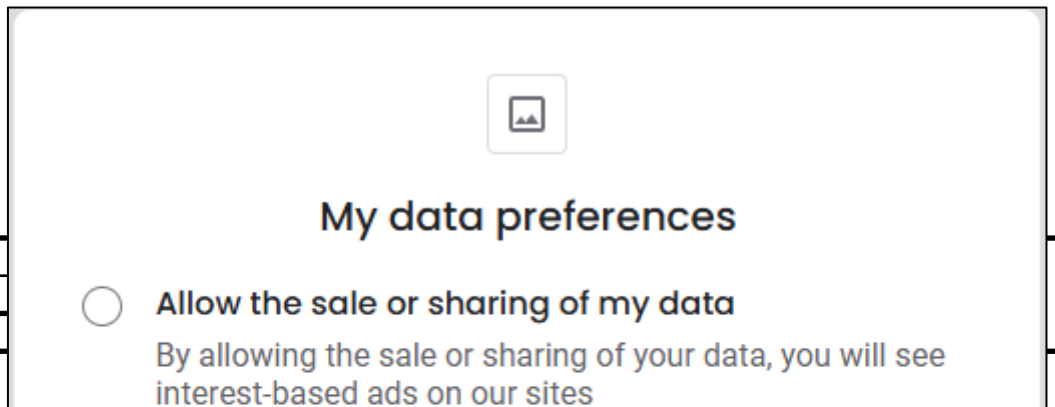

Рисунок 3.12 - Повідомлення CCPA

3.4 Висновки по розділу

У цьому розділі було проведено комплексний аналіз архітектурних підходів, що використовуються під час розробки сучасного програмного забезпечення для мобільних платформ. Було розглянуто найбільш поширені архітектурні моделі, визначено їх основні принципи побудови, особливості організації компонентів та сфери практичного застосування. Окрему увагу приділено порівнянню їхніх переваг і недоліків з точки зору підтримуваності програмного коду, масштабованості, повторного використання компонентів та зручності подальшого супроводу програмного продукту. На основі проведеного аналізу для розроблюваного застосунку було обрано архітектурну модель MVVM, оскільки саме вона найбільш повно відповідає вимогам проєкту, забезпечує чітке розділення логіки інтерфейсу та бізнес-логіки, спрощує підтримку програмного коду та покращує можливості подальшого розширення функціональності застосунку.

Крім теоретичного аналізу архітектурних рішень, у розділі було представлено графічне моделювання системи. Використання UML-діаграм дозволило візуально описати структуру та поведінку програмного забезпечення,

що значно покращує розуміння принципів його роботи як для розробників, так і для осіб, які безпосередньо не брали участі у створенні проєкту. Було побудовано та проаналізовано діаграми діяльності, діаграми послідовності та діаграми варіантів використання, що відображають взаємодію користувача із застосунком, послідовність виконання окремих процесів, а також логіку функціонування системи в різних сценаріях використання. Представлені графічні матеріали на рисунках 3.1–3.5 дають цілісне уявлення про структуру застосунку, основні функціональні можливості та зв’язки між його окремими компонентами.

Також було розглянуто основні програмні засоби, технології та інструменти, що використовувалися під час розробки мобільного застосунку Unique Names Creator. Проведено аналіз середовища розробки Android Studio, обґрунтовано причини його вибору та описано можливості, які воно надає для створення, тестування й налагодження Android-застосунків. Окрім того, було розглянуто особливості використання емуляторів та графічного редактора інтерфейсів Layout Editor, які дозволили значно спростити процес розробки та перевірки працездатності програми на різних типах пристроїв.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОЄКТУ

4.1 Загальна організація та структура коду

Зараз у світі IT усі намагаються досягнути чистої структури коду, називати правильно класи методи та змінні, мати хорошу загальну архітектуру з організованими пакетами. Це все робиться для того, щоб інші розробники могли зрозуміти код, якщо ви працюєте в команді, щоб додаток можна було підтримувати тривалий час в майбутньому. Але для чого ж тоді це робити соло-розробникам? Здавалося б можна просто писати код як завгодно, якщо ти працюєш один і нічого тобі ніхто про це сказати не зможе. І ніби то і є так, окрім декількох нюансів. По перше тобі пізніше можливо доведеться переглядати той код для додавання якихось нових особливостей, покращення самої системи чи

рефакторингу, а по друге, якщо такий стиль використовувати постійно, можна до нього і звикнути і тоді абсолютно жодна команда розробників не візьме тебе на роботу. З описаних вище причин при написанні Unique Names Creator ми намагалися організовувати наші файли та код відповідно до загальноприйнятих стандартів.

Спочатку пройдемося по структурі репозиторію. Вона складається із декількох секцій. Першою з них є коренева секція, тут знаходяться усі модульні пакети, будівельна система та конфігурація IDE, додаткові файли із переліками скриптів запуску, додаткових параметів і локальних налаштувань, файл .gitignore. Тут нас найбільше цікавить головний модуль app, що і містить основний код та ресурси застосунку. В середині є каталог src, далі вже йде поділ на main із файлами самого додатку та тестові каталоги де знаходяться файли модульного тестування. Main поділяється на два каталоги - java та res, у першому знаходиться увесь код, коли у другому стилі, шрифти та фото, іконки що використовуються у застосунку. Також у res є папка layout де знаходяться усі активності, а точніше основна візуальна їх частина, каталоги values і values-uk для англійського та українського перекладів відповідно і файли зі стилями самого додатку.

У java ж знаходяться усі класи написані на цій мові програмування, окрім тих, що використовуються для тестування. Це такі пакети, як AddsManager, FilesWithGenerations, LocalizationChanger, NamesCreator і com/grenaderius/uniquenamescreator. У першому знаходиться клас AddsManager.java, відповідальний за роботу рекламних оголошень та їх завантаження. У другому - FilesCreator.java і FilesInteractions.java, що призначені для створення та дій, що пов’язані з файлами відповідно. LocalizationChanger - пакет із класом LocalizationChanger.java, який бере на себе міну мов інтерфейсу за запитом користувача. Далі йде пакет NamesCreator, що містить файли, які й створюють основний алгоритм програми. Це абстрактний клас NamesCreationAlgorithmRunner.java та EnglishCreationsAlgorithm.java і UkrainianCreationsAlgorithm.java, тим часом, як

UALanguagesLettersAndCombinations.java і EngLanguagesLettersAndCombinations.java містять якраз символи та комбінації для використання цим алгоритмом. Детальніше цей код можна переглянути у додатку А, а також опис його роботи у підрозділі 4.2 - Основні алгоритми та структури даних, що було реалізовано.

У каталозі java також є один із основних пакетів цієї програми - com/grenaderius/uniquenamescreator - цей пакет також є використаним для ідентифікації застосунку у Google Play Store. Він є унікальним простором імен, де com - верхній домен огранізації, grenaderius - назва компанії або розробника, uniquenamescreator - назва додатку. Так ці застосунки ідентифікуються у Play Store і їх простіше розрізняти розробникам. Два додатки на одному пристрої не можуть мати однакову назву пакета. В середині цього пакета знаходяться класи-контролери активностей. Це класи саме взаємодії, вони визначають як саме і що саме будуть робити елементи інтерфейсу, як він буде демонструватися користувачу та багато іншого. Так тут є такі файли як MainActivity.java, NameGenerationOptionsActivity.java, SettingsActivity.java і ViewGeneratedNamesActivity.java. Так ми розглянули основну структуру нашого проєкту, детальніше на ню подивитися можна на рисунку 4.1 та 4.2.

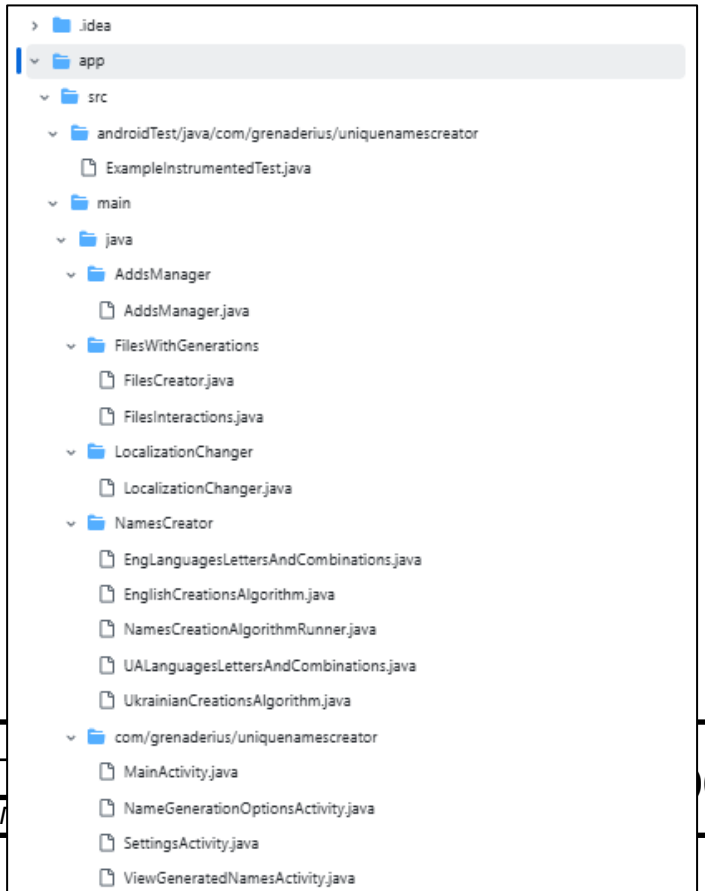

Рисунок 4.1 - Перша частина структури системи додатку на GitHub

Зрозумілий код — це естетично гарний, структурований і не перевантажений код, який має єдине форматування, оптимальну структуру та логіку, тестабельний, легко розширюваний, має чіткі та компактні коментарі щодо логіки чи вимог конкретної функціональності, відповідає принципам SOLID [23].

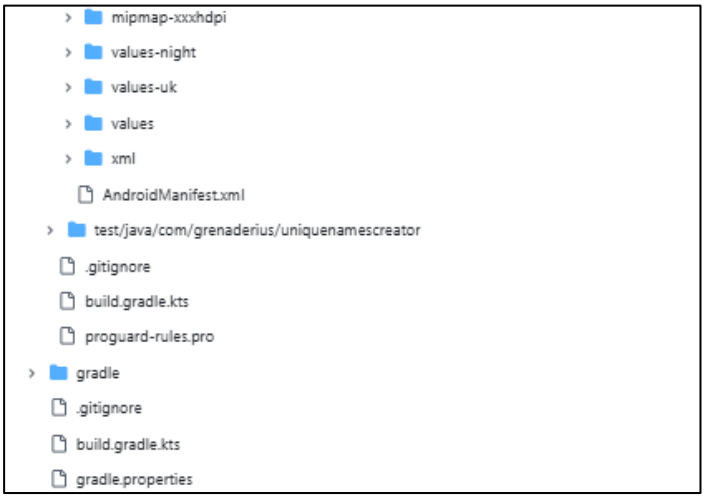

Рисунок 4.2 - Друга частина структури системи додатку на GitHub

Це визначення було знайдено на одному із форумів DOV. Під темою багато програмістів написали свої думки щодо неї і стало зрозуміло, що не можливо просто написати код із сказати, що він написаний добре, саме

структурно, а не в плані розв’язку певної задачі, оскільки його зрозумілість визначається не якимись загальними правилами, а командою розробників, що з ним працює. Таким чином виходить, що правильного і загального підходу до цього питання не має і всі принципи є теоретичними, на практиці усе ж залежить від уміння адаптуватися під стиль певної команди.

Отримавши такий висновок із прочитаного й взявши до уваги ранішездобуті знання про правила найменування класів, методів та змінних, при створенні Unique Names Creator, ми намагалися досягнути зручності у коді саме для нас, так щоб той код ми ж могли пізніше й прочитати. І відверто кажучи, спираючись на те, що застосунок розробляється уже більше року, можемо сказати, що нам вдалося цього досягти. При його створенні не раз доводилося повертатися і розбиратися із тим як саме був написаний той чи інший метод і ми не мали проблем із цим.

Загалом, якщо зібрати усе до купи, ми можемо виділити наступні критерії написання коду та найменування змінних для нашого додатку. Першою буде використання загальноприйнятого стандарту найменування змінних CamelCase, що полягає у написанні першого слова назви із малої літери й наступних із великої, що чітко розмежовує слова між собою і не перетворює назву на набір символів. При цьому ми не використовували однобуквенні назви, транслітеровані слова чи написання кирилицею, не застосовували спецсимволів і ключових слів та не починали назви із цифер. Константи в основному називали подібно звичайним змінним переходячи дорогу таким чином стандарту написання їх капсом і через нижнє підреслення. Намагалися використовувати змінні лише для певного призначення і називати їх відповідно.

Для класів ми використовували загальноприйнятий PascalCase або UpperCamelCase, що означає початок імені із великої літери так само як і кожне нове слово, це не сильно відрізняється від принципу усіх великих літер, але дає зрозуміти де клас, а де змінна, при цьому намагалися їм давати найменування іменниками із вміщеним дієсловом, як наприклад FilesCreator.java, де це було звісно можливо, так, читаючи назву класу одразу стає зрозуміло для чого він

існує.

Методи у нас також названі в загальноприйнятому стилі CamelCase, ми намагалися починати їх із дієслів, що позначають виконувану дію, і звісно правильно називати методи отримання або зміни, відомі гетери й сетери. Ось приклад назви одного із наших методів - getConsonantCombinations(), сам метод можна знайти у додатку А.

4.2 Основні алгоритми та структури даних, що було реалізовано

Тепер перейдемо до пояснення основних алгоритмів програми та структур. Адже жодна програма не може існувати без частини коду, яка виконує основну функцію. Навіть застосунки, що надають послуги якогось API і більше ні для чого не є призначеними мають свій основний алгоритм. Зазвичай він показує як саме дані отримані з ресурсу повинні оброблятися, як саме система подає на них запит, як отримує їх та демонструє.

У Unique Names Creator усе простіше. Наша система основною метою ставить для себе роботу оффлайн, а тому таки має виражений свій власний алгоритм. Цей код є представленим у додатку А. Перейдемо до його детального пояснення. Перш за все варто зрозуміти, що тут ми використовуємо абстракцію із поліморфізмом через абстрактний клас NamesCreationAlgorithmRunner.java, який є абстракцією для реалізацій алгоритмів української та англійської версій. Це класи EnglishCreationsAlgorithm.java та UkrainianCreationsAlgorithm.java відповідно. Вона наслідують наш абстрактний клас та всі разом імплементують інтерфейс Seriazable для можливості передачі об’єктів між активностями та відновлення попереднього стану об’єкта. У даній програмі ця імплементація використовується для передачі об’єктів через Intent для наявності у інших активностях. Загалом це відбувається так, об’єкт класу NamesCreationAlgorithmRunner.java створюється у першому загальному класі та ініціалізується як об’єкт одного із класів-спадкоємців у методі onClick() кнопок для зміни локалізації. Тоді той об’єкт вже передається далі у наступну активність

із налаштуваннями для генерації імен обраною мовою. Адже вибір мови у нашому додатку впливає не тільки на локалізацію, а й мову створення. Логікою в цьому є розуміння, що якщо користувач не знає англійської достатньо для взаємодії з інтерфейсом він не потребуватиме й результатів на тій мові. Те ж працює і для української, якщо користувач не володіє мовою, то йому й не потрібно на ній результатів. Таким чином ми уникаємо необхідності додавання двох окремих наборів кнопок для зміни локалізації й інтерфейсу і алгоритму й маємо ситуацію “вбивства двох зайців одним пострілом”.

Повертаючись більше до пояснення роботи основного алгоритму, варто розуміти, що поділ на два окремі класи став необхідним рішенням для досягнення хороших результатів для обох мов. Щоб детальніше зрозуміти треба мати враження про саму ідею цього коду. Вона не є дуже складною і полягає у створенні повністю нових імен шляхом використання символів та їх комбінацій для певної мови. Для цього ми також маємо два окремі класи - UALanguagesLettersAndCombinations.java і EngLanguagesLettersAndCombinations.java для українських та англійських літер та символів відповідно. Ми не вживаємо тут кириличних і латинських через все-таки різницю між ними та алфавітами самих мов. Таким чином у кожному класі ми отримуємо масиви окремо голосних, приголосних, комбінацій приголосних, комбінацій голосних, закінчень для імен чоловічої статі із першим символом - голосною і так само із першим символом - приголосною. Таким чином у класі EngLanguagesLettersAndCombinations.java ми маємо вісім масивів різних типів символів та їх комбінацій для різних статей, а також гетери до них. У UALanguagesLettersAndCombinations.java ситуація є трохи іншою, клас містить одинадцять масивів, що на три більше, а також гетери до них. Різниця полягає у наявності масиву окремо для м’якого знаку, а також літер "ґ", "ж", "ш", "ч", "щ" і "ї", "є", "ю", "я", які є певною мірою унікальними. Їхню особливість ми виявили під час мануального тестування, коли їх було просто занадто багато в одному слові й такі результати були не читабельними, а вимовити їх було ще складніше, тому було вирішено, що їх варто винести в окремі категорії й дати окремий шанс на

появу.

Тепер детальніше щодо основних методів цього алгоритму. Основним його методом, успадкованим від абстрактного класу є метод generateNames() типу масиву String, який містить у собі параметри для отримання налаштувань, обраних користувачем, щоб створити відповідний результат. Після отримання параметрів та запуску методу відбувається створення ще одного масиву типу String із розміром налаштованої користувачем кількості імен та ініціюється об’єкт рандому для використання його в середині для вибору типу масиву символів який буде отримано і вибору самого символу, для цього після кожного використання об’єкт і оновлюється. Далі йде оголошення та ініціалізація двох змінних типу int, які викликають методи getMaxCountOfElements() і getMinCountOfElements() для отримання мінімальної та максимальної можливої кількості елементів у слові відповідно. Тут все залежить від типу розміру обраного генерованого слова, так якщо обрати “малий” - максимальною кількістю елементів буде три, а мінімальною - один. Не забуваємо, що до цього може ще додаватися закінчення і ця “кількість” елементів не є справжньою кількістю, оскільки якщо випаде комбінація символів замість одного їх кількість у імені зросте.

Повернемось до нашого основного методу. Після отримання максимальної і мінімальної кількостей символів, хоч правильно це буде називати кількостями випадків випадіння масивів. Далі йде цикл від нуля і до розміру масиву із усіма іменами. У ньому першою з’являється змінна для отримання випадкового числа між максимальним і мінімальним елементами для того, щоб мати вже величину самого слова. Далі також створюється об’єкт StringBuilder для вміщення вже самого слова. Він був обраний через зменшення кількості зайнятих ресурсів, оскільки він створює String, який можна змінювати, це один запис у пам’яті замість потенційних десяти для окремого слова. Після цього йде ще один цикл, вже якраз до кількості саме імен даного слова. У ньому до об’єкту StringBuilder додається результат роботи методу letterGenerator(). Він повертає випадкову змінну з обраного рандомом масиву літер. Із шансом п’яти відсотків це може бути масив комбінацій, інші шанси ж призначені для звичайних змінних. Методи

getLettersArr() і getLettersCombinationArr() є доволі схожими у своїй роботі. Вони з певними шансами і обмеженнями у кількості співпадінь одного і того ж типу літер повертають якийсь із них. Ну а там вже з нього береться певний елемент і додається до слова. Далі, після завершення другого циклу, за схожим принципом до слова додається і закінчення відповідної статі, якщо таке було обрано, якщо ж ні, то закінчення додається через описаний вище метод letterGenerator().

Після цих всіх махінацій ми отримуємо готове ім’я, але це ще не все, до слова додається початок, якщо такий був уведений користувачем, а також перша літера згенерованого слова стає великою, якщо у префіксі останнім символом є пробіл, або початку немає взагалі. Далі вже додається префікс і закінчення якщо таке є. В кінці ж до масиву вже імен додається новостворене і перший цикл починається з початку. Приклад роботи нашої системи генерації, що тут була описана можна побачити на рисунку 4.3. Тут було використано спільне закінчення «рон», уведене користувачем, тому ми отримали дещо саме такі результати. Загалом, як можна побачити алгоритм працює справно і демонструє читабельні й різноманітні, можна навіть сказати унікальні результати, що можуть задовольнити користувачів.

Рисунок 4.3 – Приклад роботи основного алгоритму системи

Серед структур даних, як уже згадувалось раніше, в основному ми використовували масиви типу String. Окрім того, самі імена ми зберігаємо у файлі у каталозі програми, щоб користувач мав можливість за бажанням його очистити. Також ми застосовували збереження інформації за допомогою API Android із назвою SharedPreferences, що використовує файли типу FXML для збереження та передачі даних між активностями, а також для збереження налаштувань користувача навіть при повному закритті та відкритті програми з нуля. Це звісно, що не працює на очищення даних у програмі, але налаштування зберігаються при очищенні кешу.

4.3 Модульне тестування та налагодження системи

Модульне тестування або англійською unit testing — це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. В об'єктно-орієнтованому програмуванні модулем вважають метод[34]. Тестування програми таким методом є неймовірно ефективним і дозволяє виявити дуже багато помилок без використання мануального і додаткових ресурсів саме тестувальників. Також воно сильно економить час через можливість перегляду багатьох сценаріїв одночасно, тоді коли вручну це може займати дуже багато часу, особливо якщо методи включають елементи рандому і не завжди є можливість з першого разу отримати необхідний результат. Більше того модульне тестування дозволяє запускати велику кількість тестів одночасно, тоді коли в ручну це буде відбуватися набагато довше. І це ми вже не згадуємо про

швидкість відкриття комп’ютером окремих вкладок та натискання на кнопки, що є набагато швидшим за людську через відсутність необхідності у візуалізації, а тільки програмному знаходженні елементів.

Не зважаючи на всі ці переваги модульного тестування для нашого застосунку ми його не використовували для тестування нашого застосунку оскільки воно було паралельним дослідженням сфери та можливостей створення імен, удосконалення самого алгоритму. Але цей спосіб ми також не залишили осторонь, оскільки саме за допомогою нього тестувався сайт політики конфіденційності нашого додатку, що є опублікованою на GitHub Pages і доступною за кнопкою у застосунку [24]. Таким чином ми використали цей спосіб саме для тестування сайту. З його допомогою було перевірено роботу усіх елементів інтерфейсу з якими можна взаємодіяти, а також можливі переходи за посиланнями. Так ми протестували роботу випадних списків навігаційного меню, а також їх елементів і та їх наявності. Крім того було протестовано переходи за наявними посиланнями. Це все був опис безпосередньо створених нами функцій для модульного тестування. Результати проведеного нами тестування для сайту політики конфіденційності за допомогою фреймворку Nightwatch v3 можна побачити на рисунку 4.4.

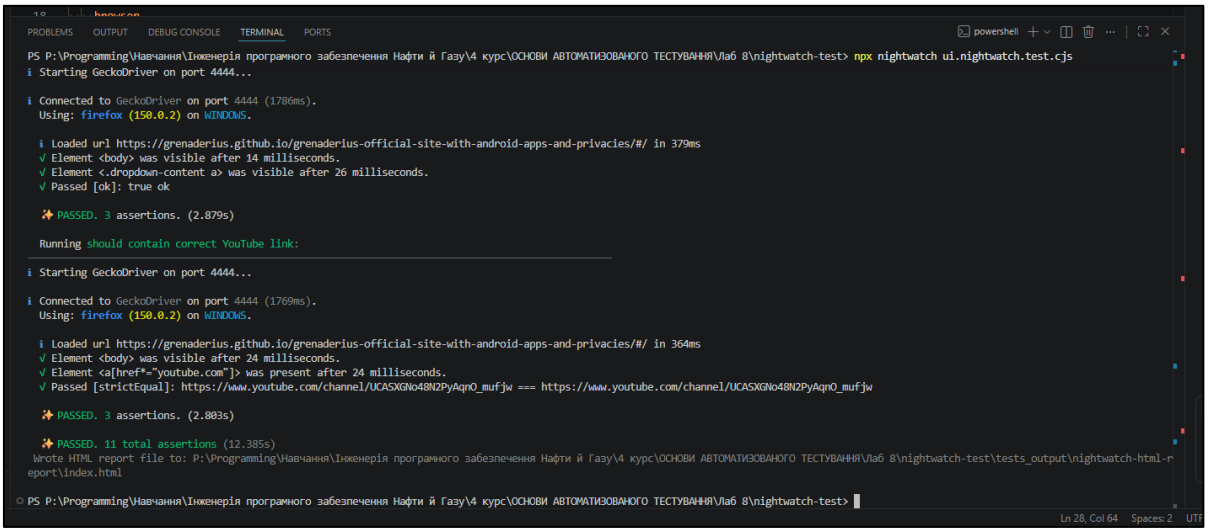

Рисунок 4.4 – Результати модульного тестування сайту політики конфіденційності за допомогою фреймворку Nightwatch v3

Перейдемо до макетів об’єктів, або більш відомих як моки, англійською

Mock Object - об'єкти, що імітують поведінку справжніх об'єктів контрольованими способами, тобто, реалізують інтерфейси справжніх об'єктів, але не мають власної реальної функціональності [25]. По суті вони є клонами реальних об’єктів, або цілих частин програми, які ми використовували під час тестування для спрощення його реалізації. Так наприклад ми за допомогою моків здійснювали імітацію сторінки без її відкриття через бібліотеку JavaScript для Node.js - Puppeteer. По суті було створено фейковий об’єкт сторінки. Це робилося перш за все для швидкості роботи тестів. Відбувалася перевірка назви сторінки. Також макети об’єктів було використано для імітації DOM-елементу. І для перевірки наявності об’єктів у випадаючих списках навігаційної панелі.

CI або англійською Continuous Integration — це неперервна інтеграція, що забезпечує послідовну й автоматизовану збірку, упаковування і тестування застосунків [26]. Сам же пайплайн є послідовністю дій, поділених за категоріями на логічні одиниці. Разом вони презентують автоматизований процес, що перевіряє код розробників та об’єднує зміни в один репозиторій, запускає збірку програми й проводить її по автоматизованим тестам. Процес має кілька типових етапів, як наприклад збірка - компіляція або пакування коду підготовлюючи його до виконання, статичний аналіз - перевірка на дотримання стилю коду й наявності вразливостей без запуску програми і саме тестування - автоматичний запуск юніт- та інтерграційних тестів, для перевірки правильності роботи логіки. При цьому, якщо будь-який із цих кроків завершується з помилкою, пайплайн блокується, а розробник одразу отримує сповіщення. Це дозволяє командам розробників виявляти баги на ранніх стадіях роозробки і не ламати основної версії. Після цього зазвичай запусчкають CD-пайплайн, що готує код до релізу й оновлює робочі сервери.

При створенні нашого проєкту ми не користувалися CI/CD-пайплайнами через відсутність такої необхідності, оскільки додаток розробляється однією людиною, нами, то й такі перевірки коду нам не є потрібними. Це відбувається тому, що перед кожним комітом ми перевіряємо внесені зміни власноруч, за допомогою мануального тестування, на власному девайсі або за допомогою вбудованих

інструментів обраної IDE Android Studio.

4.4 Висновки по розділу

У цьому розділі ми описали структуру нашого коду та показали реалізацію архітектури репозиторію, що можна побачити на рисунках 4.1 і 4.2. Окрім цього ми описали основні алгоритми та структури даних системи, пояснили наш стиль написання коду і порівняли його зі світовими стандартами. Тут було докладно переглянуто роботу нашого основного алгоритму і причини чому він може повноцінно виконувати усі функції оффлайн, пояснено призначення усіх його класів і основних методів. Докладно глянути його код можна у додатку А.

Тут було проведено аналіз структури репозиторію, описано призначення основних каталогів, пакетів і класів, а також їхню роль у забезпеченні функціонування застосунку. Особливу увагу приділено організації вихідного коду відповідно до загальноприйнятих практик розробки програмного забезпечення, що дозволяє підтримувати зрозумілість, масштабованість та зручність подальшого супроводу проєкту.

Також було розглянуто підходи до написання якісного та зрозумілого коду, проаналізовано основні принципи найменування програмних сутностей та особливості їх застосування у межах даного проєкту. На основі вивчених рекомендацій і власного практичного досвіду було сформовано набір правил, яких дотримувалися під час розробки системи. Зокрема, для найменування змінних та методів використовувався стиль CamelCase, для класів — PascalCase, а самі назви підбиралися таким чином, щоб максимально точно відображати призначення відповідних програмних елементів.

Окрім того, було продемонстровано, що навіть у випадку індивідуальної розробки дотримання стандартів програмування залишається важливим чинником успішної підтримки та розвитку програмного продукту. Правильно організована структура проєкту та зрозумілий код значно спрощують процес

внесення змін, пошуку помилок, рефакторингу та подальшого розширення функціональних можливостей системи.

У результаті було сформовано впорядковану структуру програмного забезпечення, яка забезпечує логічний поділ відповідальностей між компонентами, полегшує навігацію кодовою базою та створює надійну основу для подальшого розвитку й супроводу застосунку Unique Names Creator.

Далі було розглянуто процес модульного тестування і налагодження системи. Тут було згадано розроблювану нами політику конфіденційності [24], а також те як ми тестували сайт. Було описано CI/CD-пайплайни, макети об’єктів, або моки, що використовуються для перевірок і модульне тестування. Таким чином у цьому розділі було розглянуто структуру нашого репозиторію, описано основний алгоритм програми і причину чому він може працювати оффлайн, а також розглянуто модульне тестування для сайту політики конфіденційності нашого застосунку [24]. Описали важливість цієї сторінки і його роль у випуску додатку.

РОЗДІЛ 5. ТЕСТУВАННЯ ГОТОВОЇ СИСТЕМИ, ЗАГАЛЬНА ОЦІНКА РОЗРОБЛЕНОГО РІШЕННЯ

5.1 Використані стратегії та методики тестування

Вже давно відомим фактом є необхідність перевірки роботи перед випуском. І це стосується усього, від автомобілебудування чи запікання і до створення повноцінних застосунків. Для цього є багато різних підходів та методів, що підходять під різні випадки, показують різні речі й виконують перевірку різного функціоналу. До типів тестування у галузі програмного забезпечення належать: функціональне, інтеграційне, навантажувальне, безпеки, модульне, системне, приймальне, за моментом внесення змін, функціональне, нефункціональне, регресії, локалізаційне, сумісності та інше. Звінсо для нашого застосунку не потрібно було проводити їх усіх, адже деякі перевіряють, наприклад, роботу застосунку із віддаленим сервером чи базою даних, деякі

призначені для сайтів, інші для десктопних програм, а ми ж маємо Android-застосунок.

Таким чином для Unique Names Creator, впродовж часу розробки, було обрано наступні типи тестувань: функціональне, нефункціональне, за моментом внесення змін, інтеграційне, системне, приймальне і локалізаційне, а також внутрішнє та бета-тестування із користувачами. Окремо для сайту політики конфіденційності було проведено модульне тестування, яке є описаним у підрозділі 4.2.

Почнемо із локалізаційного тестування. Як можна зрозуміти із назви воно стосується мов та, всього, що з ними пов’язано від самих перекладів додатку і до того як певні слова було змінено для різних ситуацій. Ця перевірка для нашого застосунку відбувалася спочатку нами, як розробниками, особисто, а потім і з залученням тестувальників та потенційних користувачів.

Наступним типом описаним методом тестування стане функціональне. І тут уже можна описувати дуже багато, адже методів у застосунку вдосталь, тому скажемо в загальному, при уведенні кожної нової функції ми одразу намагалися виконувати її тестування. Так само як і при додаванні якихось змін. Такі перевірки проводилися неодноразово і часто вказували на зроблені помилки, показували нам успішність нових імплементацій.

Нефункціональне тестування є цілою групою перевірок, що націлені на систему при взаємодії користувача із додатком. Сюди входять стрес тестування, безпекове тестування, юзабіліті тестування, сумісності, продуктивності та навантажувальне. Тут варто зазначити, що не усі з них ми проводили, бо деякі просто не мають сенсу для цього додатку. Так наприклад безпекового тестування не було проведено, адже Unique Names Creator не збирає даних користувачів і навіть не потребує інтернету для повноцінної роботи. Тестування продуктивності та навантажувальне були проведені одночасно, для цього ми обирали найбільші можливі кількість і розмір імен та запускали створення, як стрес-тести, поки не отримували справді багато результатів. Юзабіліті ж було проведено не тільки нами, але й користувачами неодноразово. Також було здійснено тестування

сумісності, способом створення різних емуляторів на завантаження додатку на різних пристроях.

Тепер розберемо тестування за внесенням змін або можливо більш відоме, як тестування регресії. Це є перевірки після додавання будь-якого нового функціоналу. Тут не грає ролі те у яку саме частину програми і наскільки сильно вона її змінює. Це може бути щось незначне, майже непомітне, але у будь-якому разі варте тестування, оскільки додавання чогось одного може непрямо вплинути на щось інше таким чином змінюючи його роботу повністю. І саме тому при кожній такій зміні ми намагалися перевіряти роботу програми знову і знову, щоб знати, що все точно добре. До такого типу тестування ми залучали і користувачів, які при закритому або бета-тестуванні завантажували нові випуски на своїх пристроях і пробували їх. Це ж робилося і на власних пристроях та емуляторі.

Неодноразово під час розробки Unique Names Creator нами було проведено й інтеграційне тестування. Воно включало в себе перевірки міжмодульних взаємодій, що стосувалися активностей, раніше згаданих SharedPreferences, алгоритму генерації імен та UI. У такого типу перевірках зовнішні тестувальники не брали участі, оскільки не були знайомими із кодом, а могли тільки перевіряти результат у себе на пристрої.

Наступним типом тестування, що ми проводили, стало системне. Воно є це рівнем тестування, який перевіряє повний та повністю інтегрований програмний продукт. Метою системного тестування є оцінка комплексних системних специфікацій [27]. Воно є важливим, адже показує чи працюють реальні робочі процеси відповідно до бізнес-вимог, виявляє проблеми інтеграції, продуктивності та середовища, створює стабільну базову основу для тестування прийняття користувачами. Простими словами - це перевірка усієї системи в цілому, як вона працює і чи відповідає вимогам, чи з нею все добре. Тестування такого типу було проведено нами до прийняття рішення початку публікації застосунку та до здійснення альфа та бета тестувань, для того, щоб дати потенційним користувачам готовий продукт, що працює згідно задуму і виконує необхідні функції.

Призначене для користувача приймальне тестування, або User Acceptance Testing (UAT), є заключною перевіркою перед тим, як програмне забезпечення або систему буде остаточно надано замовникові або випущено на ринок. Цей етап важливий для забезпечення того, щоб усі функціональні та бізнес-вимоги, висунуті до продукту, були повністю виконані. Під час UAT особливу увагу приділяють тестуванню продукту з погляду кінцевого користувача, щоб упевнитися в його готовності до реальної експлуатації [28]. Основною його метою є підтвердження того, що робоча система відповідає поставленим їй, специфічним бізнес-вимогам. Також увага приділяється зручності та інтуїтивності призначеного для користувача інтерфейсу і користувачі намагаються наблизити умови тестування до реальних умов експлуатації. Це тестування проводилося нами кілька разів, спочатку власноруч, а пізніше із зовнішньою допомогою у внутрішньому та закритому тестуваннях. До нього були залучені користувачі, що не бачили застосунку до того і могли дати свій відгук щодо роботи із додатком, який вони відкрили вперше.

Альфа-тестування (alpha testing) – це вид приймального тестування, яке зазвичай проводиться на пізній стадії розробки продукту і включає імітацію реального використання продукту штатними розробниками або командою тестувальників. Зазвичай альфа-тестування полягає в систематичній перевірці всіх функцій програми з використанням технік тестування «білого ящика» і «чорного ящика» [29]. Цей тип тестування був проведений нами ще до початку приймального тестування і включав у себе нас як розробників і додатково запрошених користувачів для перевірки всіх функцій додатку і прийняття рішення про перехід до бета-тестування із більшою кількістю користувачів. Воно забезпечує краще уявлення про надійність програмного забезпечення, допомагає імітувати поведінку користувача і виявити допущені помилки.

Бета-тестування (beta testing) – інтенсивне використання майже готової версії продукту з метою виявлення максимального числа помилок в його роботі для їх подальшого усунення перед остаточним виходом (релізом) продукту на ринок, до масового споживача. Бета-тестування є реально працюючою версію

програми з повним функціоналом. І завдання бета-тестів – оцінити можливості і стабільність роботи програми з точки зору її майбутніх користувачів [29]. Почнемо з того, що сюди вже було залучено доволі багато користувачів, у кількості чотирнадцяти штук різномантіних людей та пристроїв. Це було здійснено для максимізації отриманих відгуків і розуміння до яких варто прислухатися, а які є простими побажаннями. Воно тривало доволі довго, приблизно протягом трьох тижнів та під час нього було перейдено від першої версії додатку і до робочої на даний момент - 1.2.2 [30].

5.2 Результати проведеного тестування, статистика

При виконанні локалізаційного тестування, як особисто, так і з користувачами, було виявлено низку неточностей у перекладі системи. Так наприклад при виборі величини створеного слова для англійської локалізації для середнього результату початково нами було обрано слово “middle”, а правильним було б “medium”, що правильно відзначили користувачі, що мали можливість спробувати додаток до релізу.

За результатами функціонального тестування було неодноразово вдосконалено основний алгоритм програми, проведено його аналіз і досліджено вплив кількості різних типів символів у різних мовах на створене слово. Таке ж тестування було проведено і для інших функцій системи, але основний вплив воно все ж мало на основну. Залучення користувачів до цього типу перевірок не проводилося, вони в основному допомагали із інтерфейсом та іншими речима, що стосуються результатів і налаштувань.

Проводячи нефункціональне тестування ми отримали доволі позитивні результати щодо роботи нашого додатку. Таким чином було виправлено певні помилки сумісності із пристроями, що мають різну ширину екранів, окрім того було виявлено проблему навантаження застосунку при наявності великої кількості результатів у файлі збереження, що сповільнювало швидкість генерації та час переходу на активність перегляду, відповідно знижуючи продуктивність.

Юзабаліті ж показало хороші результати, програма була доступною для використання і робочою на усіх тестових пристроях та емуляторах, не зважаючи на мінорні баги із інтерфейсом. Усі відповідні помилки, окрім навантаження при великій кількості результатів було виправлено. Так було вирішено, бо на активності показу є наявною кнопка очищення, що дає йому можливість видаляти все з файлу при бажанні. Можливо пізніше буде також додано автоматичне видалення останніх результатів при створенні нових, що можна буде налаштовувати, але до кінця ще не вирішено, чи додаток цього потребує, оскільки і без цього ним можна зручно користуватися, а додавання такої функції може зруйнувати досвід деяких користувачів, що хочуть мати всю історію своїх генерацій збереженою.

Виконуючи тестування за внесенням змін ми отримували багато різноманітних результатів, які складно описати у одному абзаці. Це відбувалося через його впровадження щоразу після внесення будь-якої зміни. Та ми не можемо сказати що воно не було того варте, адже ми щоразу таким чином могли переконатися у правильності роботи нашої програми або виявити якісь помилки. Так само і наші користувачі, що були залучені до внутрішнього та бета-тестування неодноразово повідомляли нам про переваги або недоліки у внесених змінах. Таким чином ми були переконані у створенні додатку із мінімальною кількістю багів, адже створити все одразу без помилок не можливо, навіть великі компанії часто випускають сирі продукти, як наприклад усім відомий Cyberpank 2077 [31], а тоді стикаються із критикою користувачів і їм доводиться усе переробляти.

За допомогою інтеграційного тестування було неодноразово виявлено проблеми при вдосконаленні роботи між модулями та змінах у їх взаємодії. Таким чином виявлялися одні із найбільш вагомих помилок, що стосувалися роботи налаштувань та їх збереження, а відповідно і роботи усього основного алгоритму. Виправлення таких помилок є ключовим у процесі тестування будь-якого додатку, адже вони прямо впливають на його функціонал і можуть негативно змінити досвід користування програмою.

При здійсненні системного тестування не відбувалося залучення зовнішніх користувачів, все було здійснено особисто нами, щоб не псувати початкове враження у разі невідповідності системи. Таким чином проведена перевірка показала нам повну готовність додатку до випуску і переконала у його можливості на кінцевому етапі розробки. Показала відповідність як функціональним так і нефункціональним вимогам.

За допомогою приймального тестування було виявлено ряд недоліків у інтерфейсі системи, що ускладнювала роботу із додатком користувачів, оскільки елементи не були інтуїтивно зрозумілими. Тому до початку повного випуску було перероблено значну частину користувацького інтерфейсу додатку у відповідності до бажань та пропозицій тестувальників. Пізніше, після проведених змін, таке тестування було здійснено повторно для того, щоб переконатися у здійсненому прогресі та можливості релізу. Таким чином ми упевнилися у готовності Unique Names Creator до випуску на рівні вдоволення користувачів.

Під час проведення альфа тестування було отримано та зібрано позитивні відгуки про застосунок та його функціонал, він повноцінно працював на використаних пристроях та емуляторах. Тим не менш використання пристроїв із великим екраном дало можливість зрозуміти необхідність адаптації частини інтерфейсу користувача під подібні машини. Тому ми були змушені подовжити його на певний час, унести зміни, провести повторно і аж тоді змогли таки переконатися у можливості переходу до наступного етапу, а саме бета-тестування.

Бета тестування стало останнім етапом перевірок застосунку перед релізом, під час нього було отримано ряд відгуків від користувачів і дуже сильно змінено графічний інтерфейс застосунку згідно їхніх побажань, виправлено помилку із вискакуванням форми згоди при кожному вході у застосунок а також ряд інших мінорних помилок. Воно було найбільшим стрес-тестом, що було успішно пройдено. Під час нього ми також перевіряли краші нарізних пристроях і було виявлено їхню відсутність не залежно від того як сильно навантажувався

Unique Names Creator. Відповідний графік із відомостями про збої можна побачити на рисунку 5.1.

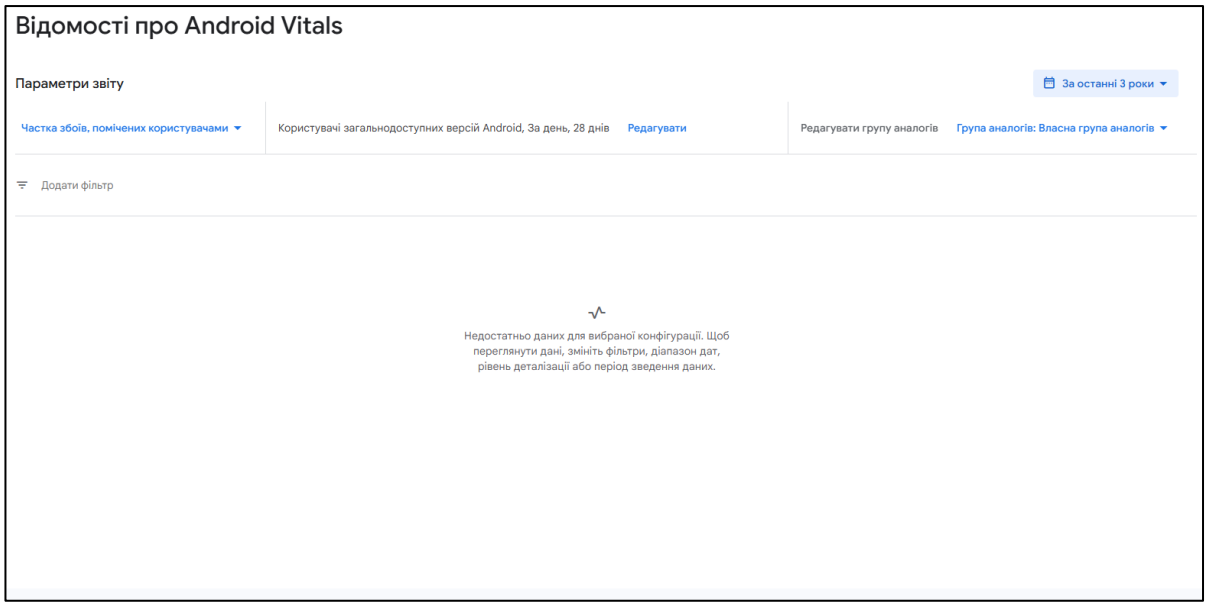

Рисунок 5.1 - Відомості про збої Unique Names Creator

Тут було б зображено відомості про збої за увесь час випуску додатку відповідно до фільтрів. Надпис “Недостатньо даних для вибраної конфігурації. Щоб переглянути дані, змініть фільтри, діапазон дат, рівень деталізації або період зведення даних.” означає відсутність відповідних даних, що є позитивним у цьому ключі.

5.3 Впровадження в експлуатацію готового рішення

Реліз — це процес випуску нової версії програмного забезпечення або оновлення в продакшн-середовище. Реліз включає в себе всі необхідні етапи для того, щоб нові функції, виправлення помилок або покращення стали доступними кінцевим користувачам або внутрішнім тестувальникам [32].

Випускати Unique Names Creator ми вирішили у Google Play Store. На це вплинули популярність застосунку та кількість його користувачів, а отже і потенційних наших. До того ж маркетплейс є з коробки на кожному смартфоні із операційною системою Android, а отже кількість можливих користувачів повинна приблизно дорівнювати кількості користувачів ОС. Тут варто розуміти, що не все так добре як воно здається, адже наявність магазину й інтернету для

завантаження застосунків не гарантує успіху продукту, що в ньому знаходиться. Для цього потрібно займатися якимось маркетингом, популяризацією додатку, а не просто опублікувати його. Та й до того ж не всім буде він до вподоби і далеко не усі можуть бути його цільовою аудиторією.

У відеоігри грає 3,1 млрд осіб, це близько 40% населення Землі. Такого висновку дійшли експерти після проведеного компанією DFC Intelligence дослідження в липні 2020 року. В ігри на ПК грають 1,5 млрд або 48% із них [33]. Від 1.5 до 2.5 мільярда грають саме в RPG, а інша частина зазвичай у інші жанри. Також згідно статистики, у 2020 році п’ятдесят мільйонів людей стабільно грають в DnD [34]. Якщо говорити про рольові настільні ігри загалом, то цю цифру мабуть можна множити вдвічі, а можливо й в тричі. Також багато хто цікавиться якимись командними іграми, а також звісно онлайн та будь-які ентузіасти із соціальних мереж. Таким чином з’ясовується, що цільовою аудиторією нашого застосунку може бути чи не 75% планети. Залишається тільки опублікувати й розрекламувати додаток правильно, так щоб зацікавити хоч когось. Таким чином, для нас, як не досвідчених розробників успіхом вважатиметься хоч сотня завантажень за перший місяць, але потенційно можливо отримати мільйони.

Що ж необхідно для випуску додатку у Play Store? Відповідь доволі проста, а завдання не дуже - пройти усі етапи випуску та перевірки від Google для отримання доступу до випуску в робочу версію, але перед цим необхідно ще створити обліковий запис розробника, що коштує 25$. Таким чином шлях до публікації складається із наступних етапів [35]:

1. Створити акаунт Google Play Console.

2. Опублікувати сайт із політикою конфіденційності.

3. Підготувати оформлення сторінки у вигляді іконок, скріншотів, банерів, трейлера й описів.

4. Підготувати підписаний APK або APK-бандл.

5. Пройти ряд опитувань щодо роботи додатку, його призначення, вартості, наявності реклами та іншого.

6. Надати власні контактні дані.

7. Створити випуск для внутрішнього тестування та додати тестувальників.

8. Наповнити сторінку додатку контентом.

9. Створити випуск для закритого тестування.

10. Знайти 12 тестувальників і провести закрите тестування впродовж двох тижнів.

11. Впевнитись у відповідності вимогам Google для випуску застосунків [36].

12. Подати заявку на доступ до робочої версії.

13. Опублікувати робочу версію, яка і є справжнім релізом.

Ці всі етапи було пройдено для публікації нашого застосунку і тепер він є доступним у Play Store [30]. Пройдемося детальніше, по найбільш цікавих із них або таких про які варто розписати. Так сайт із політикою конфіденційності [24] було створено ще заздалегідь через знання вимог ознайомлення із вимогами до публікації. Так само у версії застосунку 1.2.1 було додано до нього окрему активність із налаштуваннями додатку, де є кнопка, що перекидає користувача до цієї політики, відповідно до правил Google [36]. Так само туди було додано і кнопку для повторного відкриття форм згоди GDPR [21] та CCPA [22] згідно тих же вимог, адже користувач має мати змогу відмовитися від збору або продажу його даних коли забажає.

Щодо підготовки банерів та іконок, то основний логотип додатку у нас був уже давно, правда із часом він був трохи змінений. Від початку і до кінця виконувався за допомогою ресурсу Canva [37]. Побачити логотип застосунку можна на рисунку 5.2.

Рисунок 5.2 - Логотип Unique Names Creator

Банери ж були підготовлені окремо, для двох версій сторінок - української та англійської відповідно за допомогою ресурсу Figma [38]. При їх створенні ми намагалися відповідати усім загальноприйнятим правилам дизайну. Таким чином було створено по вісім банерів для кожної із сторінок. На них було описано основні можливості програми та продемонстровано її застосування користувачу. Так ми маємо продемонстровану основну сторінку із закликом вибору мови, сторінку налаштувань із закликом до їх використання. Активності результатів із показом результатів створення із обраними налаштуваннями. Приклади генерації електронних адрес та налаштувань для них, а також приклади створення із доданими префіксами чи закінченнями. Ми б хотіли додати набагато більше зображень з описами можливостей, але, нажаль, кожна сторінка є обмеженою до восьми максимум. Створені нами банери для Unique Names Creator є продемонстрованими на рисунку 5.3.

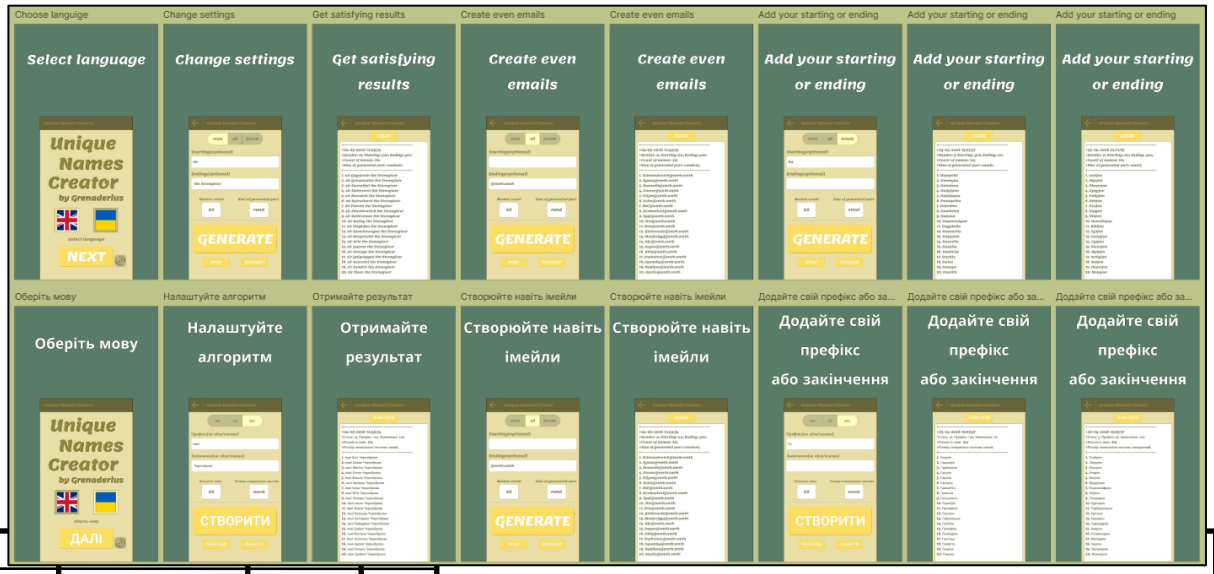

Рисунок 5.3 - Банери Unique Names Creator

Окрім банерів на сторінці також було створено окремий графічний елемент, що можна побачити під час пошуку додатку. Він є важливим для залучення відвідувачів Play Store, що використовують великі дисплеї типу комп'ютерів або планшетів та схожого. Його приклад для нашого Unique Names Creator є на рисунку 5.4.

Рисунок 5.4 - Вигляд feature graphics при пошуку застосунку

Щодо підготовки APK-бандлу, то це робиться доволі просто в середині IDE. Процес був здійснений нами неодноразово, оскільки навіть для створення випусків для тестування необхідно мати підписаний додаток. Спочатку необхідно створити файл з підписом для конкретного застосунку, а далі ввести паролі і відповідно таким чином створити підписаний бандл. Ця процедура придумана не просто так і служить своєрідним забезпеченням того, що ви

зможете випускати оновлення для додатку у майбутньому, адже він підтверджує автора, а також те, що програма не була зміненою після збірки у APK або бандл.

Щодо опитувань, то тут усе просто - чим простіший додаток, тим легшими і меншими є опитування. Тобто якщо у застосунку відсутні будь-які реєстрації, необхідності підключення до інтернету, реклама, зовнішні сховища, збір даних, то подібні тести - це просто формальність, де практично всюди необхідно відповісти ні. Якщо ж щось із цього є присутнім, то все ускладнюється. У нашому випадку наявності реклами потрібно було у середині пройти ще три етапи опитувань, загалом це було приблизно шість окремих тестів, на кожен з яких необхідно було виділити принаймні тридцять хвилин. При чому дуже важливо відповідати чесно і правильно, оскільки на етапі цих опитувань застосунок можуть відхилити.

Щодо внутрішнього та закритого тестування, то воно було проведено, як було згадано у підрозділі 5.1 та 5.2 із чотирнадцятьма тестувальниками протягом двох тижнів і за допомогою них було виявлено та виправлено доволі багато проблем і недоліків, що в кінцевому покращило застосунок. Тут варто згадати, що під час цього тестування обрані користувачі повинні не видаляти гру із свого пристрою і не змінювати основний обліковий запис на інший, бо усе може злетіти і таймер двох тижнів початися спочатку. Між цими двома етапами необхідно було наповнити сторінку застосунку вищепоказаними фото та описами. Тут варто зазначити, що насправді є набагато більше різних фото для показу на різних пристроях, просто ми націлювалися саме на смартфони і в портретному режимі, а тому маємо виключно вище показані зображення.

Щодо описів як короткого так і більшого, тут варто зазначити, що обидва впливають на пошук додатку в магазині, але короткий опис користувач бачить одразу і зазвичай саме за ним обирає чи використовувати цей застосунок чи пропустити, звісно після назви й логотипу. З мінусів він є дуже обмеженим у величині, доступно тільки вісімдесят символів, тоді як у великому - чотири тисячі. І насправді це є проблемою, бо коли ти можеш красиво описати роботу додатку англійською і вмістити туди достатню кількість ключових слів, то

українською потрібно учетверо більше місця, щоб воно звучало логічно і правдоподібно. Якщо порівнювати наші описи іноземною: “Create unique names for your in-game and tabletop characters, emails & much more” та рідною: “Створюйте унікальні імена для своїх ігрових і настільних персонажів та іншого” мовами, можна побачити цю різницю. Набагато більше і зрозуміліше усе виходить саме англійською. Це зумовлено будовою мови, але принаймні у нашому випадку не все так погано, як коли іноземці перекладають на нашу мову ці описи через перекладач або штучний інтелект.

Після всіх цих етапів, отримавши дозвіл від Google на публікацію і пройшовши фінальне опитування ми нарешті отримали доступ до робочої версії та опублікували Unique Names Creator у Play Store [30]. Шлях був не простий, але ми зуміли це зробити і є справді задоволеними даним результатом на цей момент. Із рекомендацій від платформи отримали розблокування лендскейпного режиму в застосунку та відмову від деяких використаних API для зміни кольорів, через припинення їх підтримки у новітніх версіях Android і використання штучного інтелекту для ранішез гаданих функцій, чим ми і будемо займатися найближчим часом, а ще звісно маркетингом, необхідно ж повідомити світу про наш реліз.

За час наявності застосунку у Play Store, на момент написання роботи це чотирнадцять днів, сама публікація була здійснена 14.05.2026, ми маємо сімнадцять завантажень Unique Names Creator, а також кількість установлень 94,4%, що позначає кількість активних пристроїв зі встановленим додатком за останні тридцять днів. Це все підтверджує статистика, зображена на рисунку 5.5.

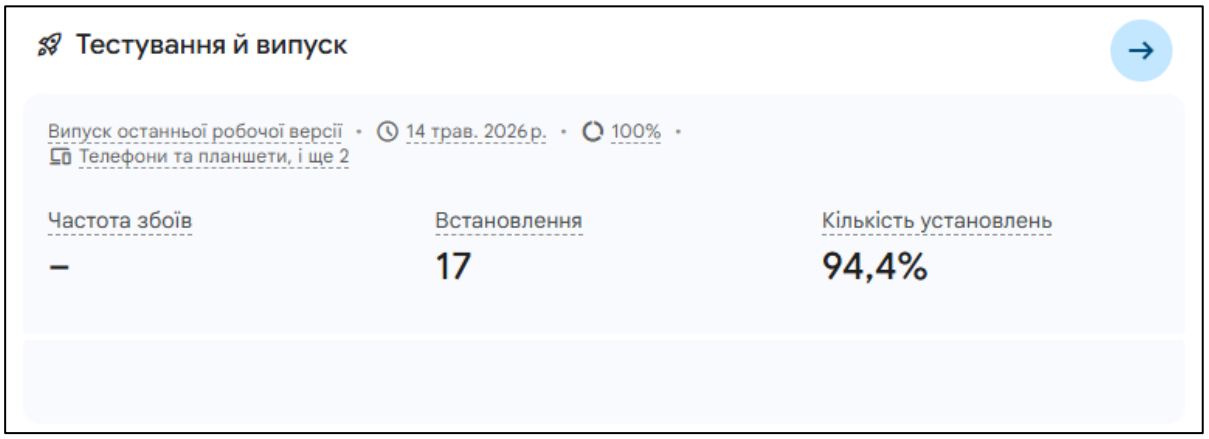

Рисунок 5.5 - Статистика випуску Unique Names Creator

Також на цьому рисунку можна побачити відсутність будь-яких збоїв на пристроях користувачів, показуючи надійність та відсутність критичних помилок у застосунку. Окрім того, якщо глянути іншу статистику, можна також побачити відсутність помилок ANR, що означає відсутність помилок із часом очікування відповіді на запити у системі. У нас таких помилок і не може бути, адже додаток не має ніякого допоміжного серверу чи чогось подібного й може працювати повністю автономно, без підключення до мережі. Це ж підтверджує і графік зображений на рисунку 5.6.

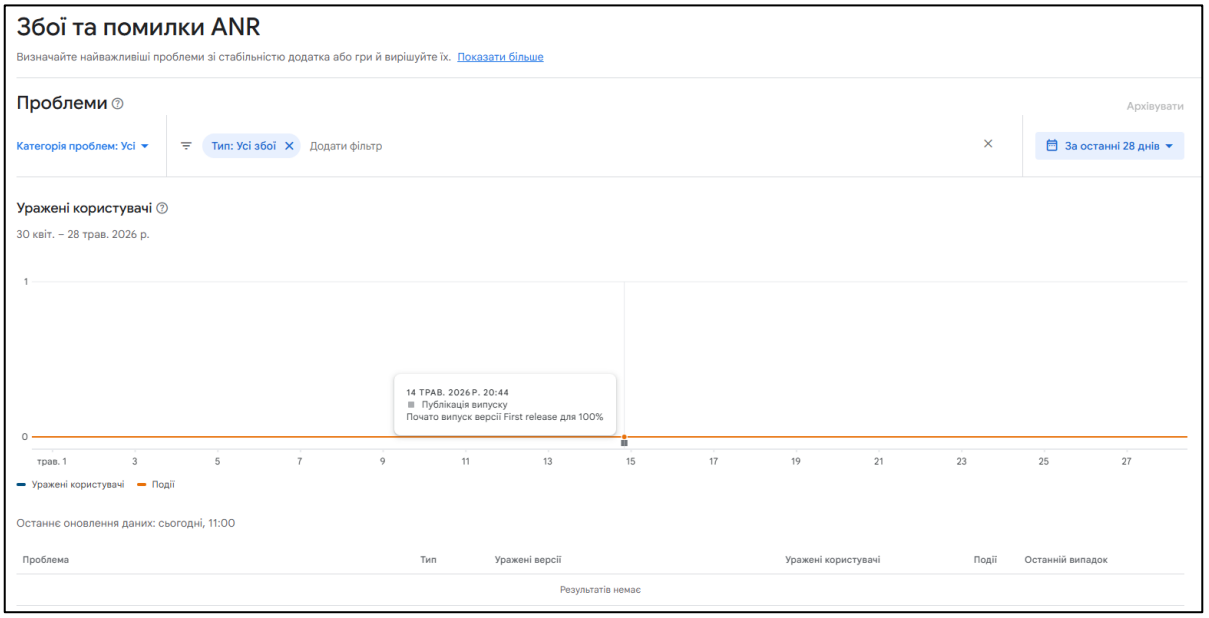

Рисунок 5.6 - Статистика збоїв та помилок ANR Unique Names Creator

Якщо поглянути на відгуки та оцінки користувачів, то наш застосунок поки їх немає. На нашу думку це відбувається тому, що ми не додавали ніякого повідомлення користувачу із проханням оцінки та написання відгуку, як робить багато додатків. Як часто ви самі вирішуєте написати відгук до будь-якого застосунку, яким користуєтесь? Особисто ми робимо це неймовірно рідко, маючи буквально кілька відгуків за більш ніж десятирічний досвід користування Play Store. І це не зважаючи та те, чи розробник просить про них. Відгуки хочеться

писати тільки тоді, коли справді є до того бажання, а як немає, то й писати нічого. Те ж стосується і оцінок. Підтвердження відсутності будь-яких коментарів можна побачити на рисунках 5.7, де дуже добре видно повідомлення: “У вашого додатка немає відгуків”. Якщо ж глянути на рисунок 5.8, то там знаходиться графік зміни оцінок додатку за день місяця. Сірим квадратом позначено дату випуску. На ньому добре прослідковується відсутність будь-яких ліній чи точок, а отже і оцінки є відсутніми.

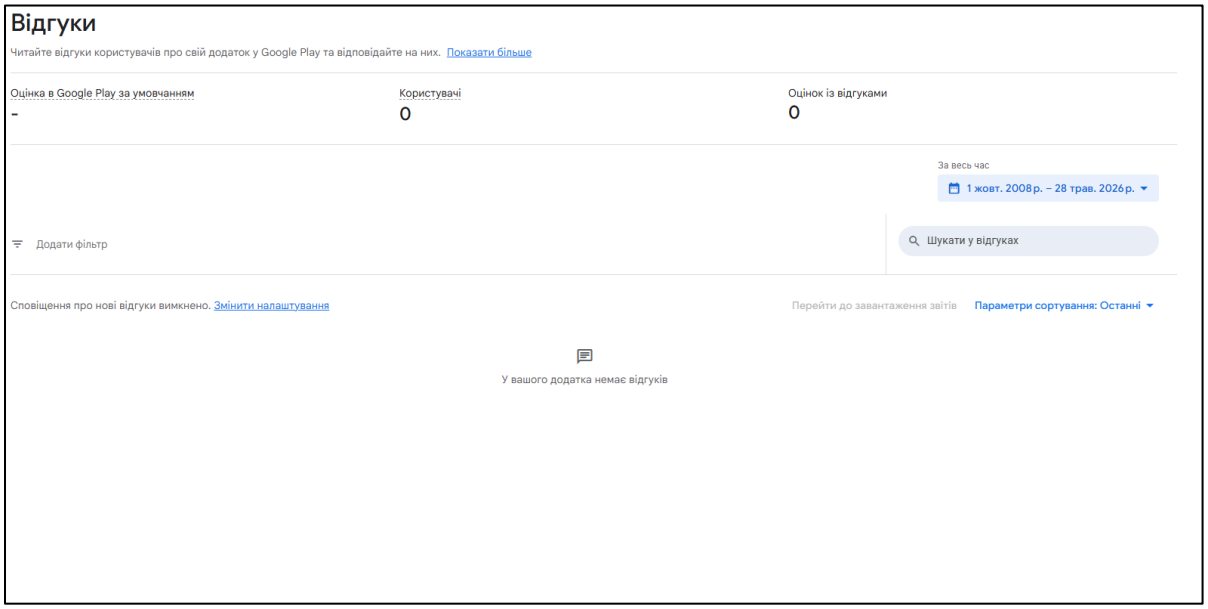

Рисунок 5.7 - Відгуки додатку Unique Names Creator

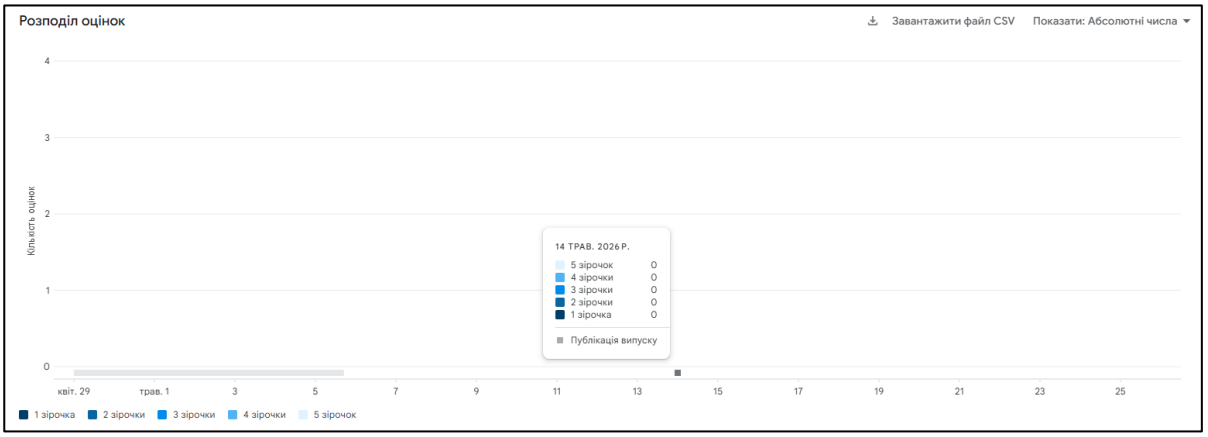

Рисунок 5.8 - Розподіл оцінок додатку Unique Names Creator

Далі візьмемо реальні завантаження застосунку. Станом на 28.05.2026 їх кількість становить вісімнадцять, при конверсії у 65,4%, що можна побачити на рисунку 5.9.

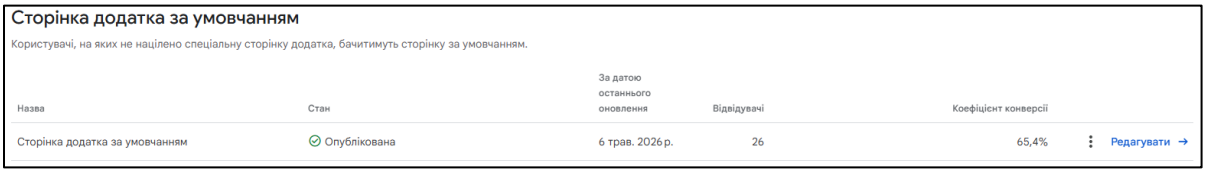

Рисунок 5.9 - Конверсія Unique Names Creator

Це означає, що із двадцяти шести користувачів, які відкрили сторінку додатку вісімнадцять його завантажило, таким чином отримуємо потенційно вісім користувачів, що вирішили не завантажувати додаток, або принаймні шість, якщо не рахувати наші входи для перегляду сторінки. Цю статистику на даний момент можемо вважати дуже навіть позитивною, адже із чотирьох потенційних клієнтів додатку троє його завантажують всього по фото й опису, навіть без наявності трейлеру. Порівняно із категорією інструментів, де знаходиться Unique Names Creator середня активність користувачів за день становить 31,25%, коли в аналогів близько 10%, що є дуже позитивною статистикою, зважаючи на те, що Unique Names Creator не призначений для щоденного використання. Підтвердженням вище написаному є графіки на рисунках 5.10 та 5.11.

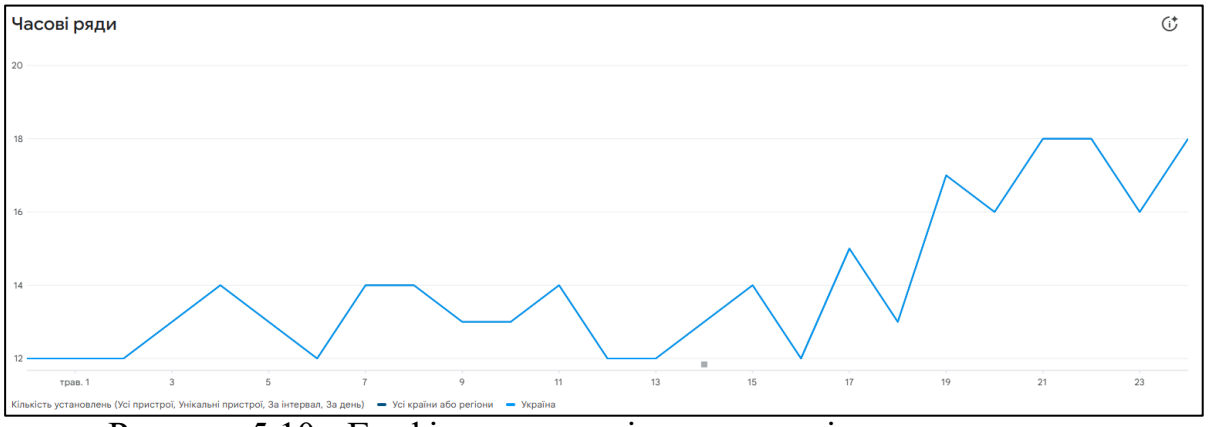

Рисунок 5.10 - Графік активності користувачів додатку за день

Рисунок 5.11 - Графік активності користувачів за день порівняно з аналогами для застосунку Unique Names Creator

Повернемося до рекламних оголошень, що показуються у нашому додатку. Ми їх сюди додавали для монетизації, а отже, не зважаючи на порівняно низьке охоплення у перші два тижні, без будь-якого маркетингу поглянемо на зміну графіків доходу, адже це також є вагомою віхою досягнених результатів нашого Unique Names Creator. Таким чином у загальному спостерігається позитивна статистика у доходах за місяць, не зважаючи на невеликий спад за останній тиждень. Таким чином ми можемо спостерігати, що запити на рекламу отримують оголошення у 63,2%, що є позитивним показником. Приблизний дохід за місяць становить 0,05$, а eCPM - фактична ціна за тисячу показів [39] для Unique Names Creator становить 0,16$, що не є поганим показником маючи чотири баннера у застосунку, а отже, якщо тисяча користувачів переглянуть бодай по одній рекламі із кожного, то ми будемо мати 0,64$, а зважаючи на те, що зазвичай користувач буде по кілька разів переходити від активності налаштувань і до активності із результатами, то ми можемо дійти і до більшої кількості аніж 1$ за тисячу користувачів тільки із звичайних банерів, що не заважають їх досвіду використання застосунку. Правда до можливості виводу, а саме сотні доларів дійти буде дещо складно, адже у такому випадку необхідно сто тисяч людей, а отже використання такої реклами не стане релевантним і не принесе реального доходу, тим не менш ми будемо намагатися покращувати результати у подальшому, шляхом популяризації уже самого додатку серед цільової аудиторії. Переглянути описану нами вище статистику можна на рисунках 5.12 та 5.13, де зображені

графіки зміни приблизного доходу, запитів, показів, частоти збігу та зафіксованої eCPM, а також статистика по доходам нашого Unique Names Creator за місяць.

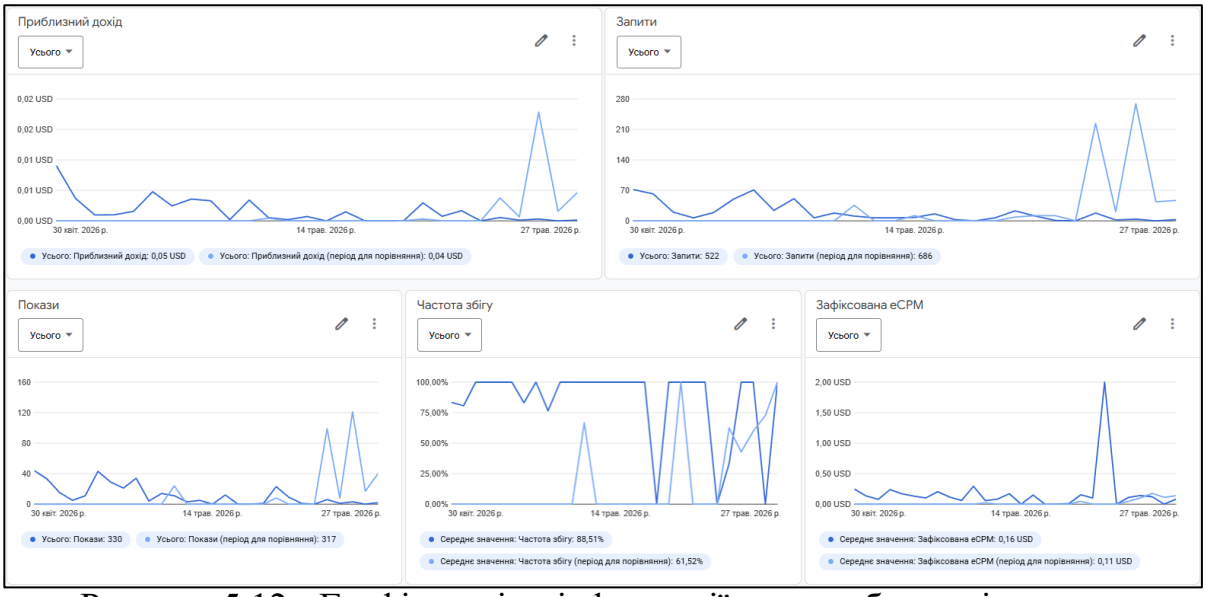

Рисунок 5.12 - Графіки зміни інформації про прибуток від реклами

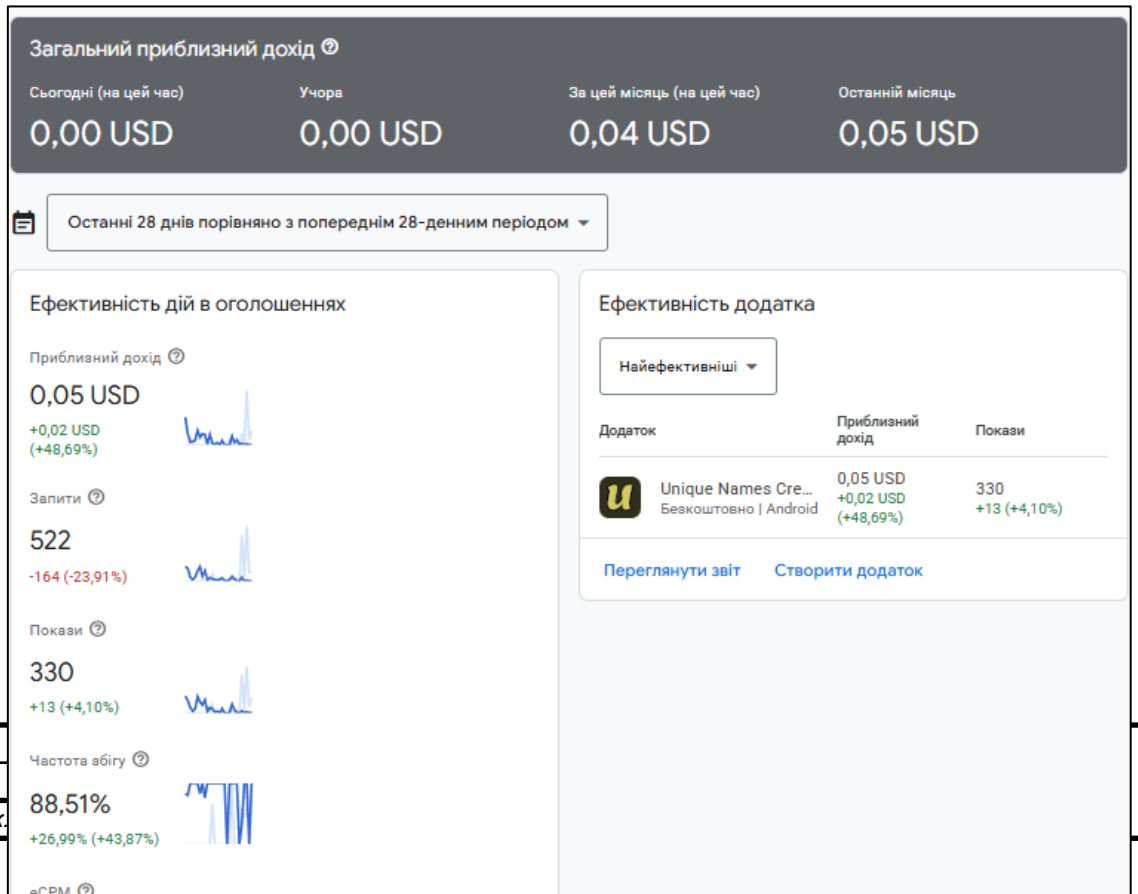

Рисунок 5.13 - Графіки місячної статистики ефективності оголошень для застосунку Unique Names Creator

5.4 Висновки по розділу

У цьому розділі ми розглянули процеси тестування нашого додатку та методи які ми для цього використовували, розповіли про його публікацію і її результати. Загалом під час перевірок додатку було використано дуже багато різних способів, як наприклад функціональне, нефункціональне, за моментом внесення змін, інтеграційне, системне, приймальне і локалізаційне, а також внутрішнє та бета-тестування із користувачами. Також результати тестування підтвердили коректність роботи основного алгоритму генерації назв, правильність функціонування механізмів локалізації, збереження налаштувань користувача та взаємодії між окремими компонентами системи.

Під час публікації застосунку ми повинні були пройти певні кроки для підтвердження його якості та важливості для користувачів. Це включало різноманітні відповіді на питання, перевірки відповідності правилам випуску у Google Play Store, а також внутрішнє і закрите тестування із фіксованою кількістю користувачів та часу. Окрім того необхідно було створити банери для сторінки у маркетплейсі й наповнити її описами - великим та маленьким. Випустивши Unique Names Creator ми отримали доволі хороші результати, двадцять шість різних користувачів змогли знайти його сторінку із яких вісімнадцять його завантажило і це не зважаючи навіть на відсутність будь-якого маркетингу. Щодо монетизації, то ми частково у ній розчарувалися, адже як виявляється для отримання хоч якогось прибутку її має переглянути дуже велика

кількість користувачів, що робить її не дуже релевантною. За цих два тижні ми також не отримали ніяких відгуків чи оцінок, що доволі невтішно, але все ще попереду. Далеко не всі користувачі хочуть писати хоч якісь відгуки до застосунків, якими вони користуються. Таким чином у цьому розділі ми описали завершення розробки нашого застосунку, його тестування і випуск у Google Play Store та його результати.

ВИСНОВКИ

Під час виконання цієї роботи нами було досліджено сучасний стан розробки мобільних застосунків та актуальні тенденції розвитку програмного забезпечення для мобільних пристроїв. Проведений аналіз показав, що значна частина сучасних мобільних сервісів орієнтується на використання віддалених серверів, зовнішніх API та постійного мережевого з’єднання. Такий підхід дає широкі технічні можливості, проте водночас створює залежність від якості підключення до мережі Інтернет та обмежує доступність функціоналу в умовах нестабільного або відсутнього з’єднання. У результаті було підтверджено актуальність створення мобільних рішень, здатних працювати автономно та виконувати свої основні функції без підключення до мережі.

У межах роботи було проведено повноцінний цикл розробки програмного продукту, від аналізу наявних даних, визначення вимог та постановки задачі й до повноцінного його випуску у маркетплейсі від Google – Play Store. При розробці нами було створено основний код програми – алгоритм генерації, що забезпечив основну її особливість та відмінність від інших – унікальність отриманих результатів у поєднанні зі змогою повноцінно працювати оффлайн. Він базується на використанні символів та їх комбінацій у поєднанні із ретельно відкаліброваним рандомом для досягнення не тільки новизни, а й краси, читабельності та ефективності. Проведені тестування із залученням користувачів підтвердило його працездатність і здатність досягати різноманітних результатів

відповідно до отриманих із налаштувань параметрів.

У процесі розробки ми також працювали над монетизацією майбутнього продукту, саме тому до нього були додані рекламні банери від Google AdMob, досягаючи таким чином можливості отримання доходу. Не забували ми й про збереження прав людини та її приватності, саме тому до застосунку було додано форми згоди відповідно до законів певних держав і їх об'єднань. З цієї ж причини було створено й окремий сайт з політикою конфіденційності додатку й додано посилання до нього через кнопку в налаштуваннях, щоб користувач завжди міг переконатися у тому, що ми цінуємо його приватність. У результаті публікації ми не отримали якогось неймовірного охоплення через відсутність будь-якого маркетингу для створеного рішення. Цю проблему плануємо вирішити в подальшому шляхом створення рекламних відео роликів та їх розповсюдження на відповідних платформах. Окрім того варто не забувати, що завжди є куди рухатись далі, а тому за успіху розповсюдження у планах лежить перетворення додатку на повноцінну оффлайн-платформу зі створення імен, адрес електронних адрес, назв компаній, паролей та іншого.

Отримані результати роботи мають високу вагу для її подальшого вдосконалення та покращення користувацького досвіду. Їх наукова новизна полягає у дослідженні та практичній реалізації алгоритму генерації, що функціонує виключно на мобільному пристрої без використання зовнішніх сервісів і забезпечує створення унікальних результатів шляхом комбінування символів різних алфавітів із використанням контрольованої випадковості. Практичне значення роботи полягає у можливості використання отриманих результатів для подальшого вдосконалення алгоритмів автономної генерації, оптимізації мобільних застосунків із повною оффлайн-функціональністю та дослідження взаємодії користувача з програмними продуктами такого типу.

Підсумовуючи результати роботи, можна зробити висновок, що поставлена мета була досягнута повністю: створено працездатний мобільний застосунок, здатний виконувати свої функції повністю оффлайн, без необхідності мережевого підключення. На жаль, він ще не має великої аудиторії, але за її

досягнення на нього є великі плани із покращення його роботи й перетворення на повноцінну платформу. В цілому отримані результати підтверджують можливість реалізації подібних додатків та публікації їх на відомих маркетплейсах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. statcounter GlobalStats: Desktop vs Mobile vs Tablet Market Share Worldwide - May 2026 - [Електронний ресурс]. - Режим доступу: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide- Дата звернення: 12.04.2026.
2. Мій провайдер: Нестабільне з'єднання з інтернетом: причини та шляхи вирішення - [Електронний ресурс]. - Режим доступу: https://surl.li/skywif - Дата звернення: 12.04.2026.
3. Як економити інтернет-трафік користувачам iOS і Android – [Електронний ресурс]. – Режим доступу: https://surl.li/bvlglf - Дата звернення: 13.04.2026.
4. TheGuardian: Universal internet access unlikely until at least 2050, experts say – [Електронний ресурс]. – Режим доступу: https://www.theguardian.com/technology/2019/jan/10/universal-internet-access-unlikely-until-2050-experts-say-lack-skills-investment-slow-growth - Дата звернення: 13.04.2026.
5. Українська бібліотечна енциклопедія: Інтернет – [Електронний ресурс]. – Режим доступу: https://ube.nlu.org.ua/article/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82 - Дата звернення: 13.04.2026.
6. Вікіпедія: Мобільний застосунок – [Електронний ресурс]. – Режим доступу: https://surl.lu/rwunhw - Дата звернення: 14.04.2026.
7. Вікіпідручник: Програмування під Android/Структура Android програми – [Електронний ресурс]. – Режим доступу: https://surl.lu/itatiw - Дата звернення: 16.04.2026.
8. BRANDER: APK-ФАЙЛ [Електронний ресурс]. – Режим доступу: https://brander.ua/glossary/apk-fayl - Дата звернення: 24.05.2026.

9. SQ Magazine: Mobile App Growth Statistics 2026: Downloads, Spending & User Trends – [Електронний ресурс]. – Режим доступу: https://sqmagazine.co.uk/mobile-app-growth-statistics/ - Дата звернення: 16.04.2026.

10. Itransition: Key mobile app statistics for 2026: app usage, revenue & success factors – February 10, 2026 – [Електронний ресурс]. – Режим доступу: https://www.itransition.com/services/application/development/mobile/statistics - Дата звернення: 17.04.2026.

11. TaLeWorLds: Mount&Blade: With Fire and Sword – [Електронний ресурс]. – Режим доступу: https://www.taleworlds.com/en/games/fireandsword - Дата звернення: 18.04.2026.

12. Валера Ф. VISURE: Функціональні та нефункціональні вимоги(з прикладами) – [Електронний ресурс]. – Режим доступу: https://surl.lu/xmdcck - Дата звернення: 26.04.2026.

13. JAVARUSH: Частина 2. Поговоримо трохи про архітектуру ПЗ – [Електронний ресурс]. – Режим доступу: https://javarush.com/ua/groups/posts/uk.2519.chastina-2-pogovorimo-trokhi-pro-arkhtekturu-pz - Дата звернення: 28.04.2026.

14. Каграманова Ю. DOU: Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти – [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/40575/ - Дата звернення: 18.04.2026.

15. TestMatick: Середовище розробки – [Електронний ресурс]. – Режим доступу: https://testmatick.com/uk/glossary/seredovyshhe-rozrobky/ - Дата звернення: 03.05.2026.

16. WikipediA: Category:Integrated development environments – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Category:Integrated_development_environments - Дата звернення: 07.05.2026.

17. WikipediA: Category:Microsoft Visual Studio – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Category:Microsoft_Visual_Studio - Дата звернення: 07.05.2026.

18. Developers: Develop a UI with Views – [Електронний ресурс]. – Режим доступу: https://developer.android.com/studio/views/layout-editor - Дата звернення: 07.05.2026.

19. Kirfy: Advantages and Disadvantages of Kotlin – [Електронний ресурс]. – Режим доступу: https://krify.co/advantages-and-disadvantages-of-kotlin/ - Дата звернення: 07.05.2026.

20. Google AdMob Довідка: Довідковий центр: Посібник з AdMob: Початок роботи. Як працює AdMob – [Електронний ресурс]. – Режим доступу: https://support.google.com/admob/answer/7356092?hl=uk - Дата звернення: 08.05.2026.

21. GDPR TEXT: Загальний регламент про захист даних (GDPR) – [Електронний ресурс]. – Режим доступу: https://gdpr-text.com/uk/ - Дата звернення: 03.05.2026.

22. ВікіпедіЯ: California Consumer Privacy Act (CCPA) – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/California_Consumer_Privacy_Act_(CCPA) - Дата звернення: 03.05.2026.

23. Orlova T. DOU: Як писати код так, щоб він був зрозумілий іншим розробникам – [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/42528/ - Дата звернення: 09.05.2026.

24. GRENADERIUS: UNIQUE NAMES CREATOR Privacy Policy – [Електронний ресурс]. – Режим доступу: https://grenaderius.github.io/grenaderius-official-site-with-android-apps-and-privacies/#/privacy/UniqueNamesCreator - Дата звернення: 10.05.2026.

25. ВікіпедіЯ: Макет об'єкта – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D0%B5%D1%82_%D0%BE%D0%B1%27%D1%94%D0%BA%D1%82%D0%B0 - Дата звернення: 10.05.2026.

26. Філімонов С., Андрєєв М. Що таке CI/CD, як він працює та коли знадобиться на проєкті. Лайфхаки та bad practices – [Електронний ресурс]. –

Режим доступу: https://highload.tech/uk/blogs/shho-take-ci-cd-yak-vin-pratsyuye-ta-koli-znadobitsya-na-proyekti-lajfhaki-ta-bad-practices/ - Дата звернення: 11.05.2026.

27. Гамільтон Т. GURU99: Що таке системне тестування? Типи з прикладом – [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/system-testing.html - Дата звернення: 11.05.2026.

28. Семенів М. LEMON school: Приймальне тестування (Acceptance Testing) – [Електронний ресурс]. – Режим доступу: https://lemon.school/blog/pryjmalne-testuvannya-acceptance-testing - Дата звернення: 11.05.2026.

29. QATestLab: Альфа- та Бета-тестування – [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/alpha-beta-testing/ - Дата звернення: 11.05.2026.

30. Google Play: Unique Names Creator – [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.grenaderius.uniquenamescreator - Дата звернення: 12.05.2026.

31. Cyberpank: GET ULTIMATE CYBERPANK 2077 EXPERIENCE – [Електронний ресурс]. – Режим доступу: https://www.cyberpunk.net/ua/en/ - Дата звернення: 12.05.2026.

32. HIRE1: Що таке Реліз? – [Електронний ресурс]. – Режим доступу: https://hire1.com.ua/glossary/release - Дата звернення: 16.05.2026.

33. Торговий корабель: Як багато людей грають у комп'ютерні ігри? – [Електронний ресурс]. – Режим доступу: https://kamin.goroh.cx.ua/ukraincyam/yak-bagato-lyudey-grayut-u-komp-39-yuterni-igri.html - Дата звернення: 14.05.2026.

34. reddit: r/rpg: How many people play TTRPGs? – [Електронний ресурс]. – Режим доступу: https://www.reddit.com/r/rpg/comments/1enm0b0/how_many_people_play_ttrpgs/ - Дата звернення: 14.05.2026.

35. Play Console Довідка: Як опублікувати додаток – [Електронний ресурс]. – Режим доступу: https://support.google.com/googleplay/android-developer/answer/9859751?hl=uk - Дата звернення: 16.05.2026.

36. Google Play: Центр правил для розробників – [Електронний ресурс]. – Режим доступу: https://play.google/intl/uk/developer-content-policy/ - Дата звернення: 16.05.2026.

37. Canva: Який дизайн ви хочете створити? – [Електронний ресурс]. – Режим доступу: https://www.canva.com/ - Дата звернення: 16.05.2026.

38. Figma: Prompt, code, and design from first idea to final product – [Електронний ресурс]. – Режим доступу: https://www.figma.com/ - Дата звернення: 17.05.2026.

39. Google AdMob Довідка: Довідковий центр: Початок роботи: Як користуватися головною інформаційною панеллю – [Електронний ресурс]. – Режим доступу: https://surl.lt/szukdh - Дата звернення: 18.05.2026.

ДОДАТКИ

Додаток А

Код файлу NamesCreationAlgorithmRunner.java

```
package NamesCreator;

import java.io.Serializable;

public abstract class NamesCreationAlgorithmRunner implements Serializable {
   public abstract String[] generateNames(int countOfNames, String countOfGeneratedElements, String gender, String startingText, String endingText);
}
```

Код файлу UkrainianCreationsAlgorithm.java

```
package NamesCreator;

import androidx.annotation.NonNull;
import org.jetbrains.annotations.Contract;
import java.io.Serializable;
import java.util.Random;

public class UkrainianCreationsAlgorithm extends NamesCreationAlgorithmRunner implements Serializable{
   private static Random random;
   private static int vowelsInRowGenerated = 0;
   private static int consonantsInRowGenerated = 0;
private static int counter = 0;
```

Продовження додатку А

```
   private static final String maleGender = "ч", femaleGender = "ж";
```

```
    @NonNull
    public String[] generateNames(int countOfNames, String
countOfGeneratedElements, String gender, String startingText, String endingText){
        String[] namesArr = new String[countOfNames];
        random = new Random();

        int maxCountOfElements =
getMaxCountOfElements(countOfGeneratedElements);
        int minCountOfElements =
getMinCountOfElements(countOfGeneratedElements);

        for (int i = 0; i < namesArr.length; i++) {
            int nameLettersCount = random.nextInt(maxCountOfElements -
minCountOfElements + 1) + minCountOfElements;
            StringBuilder name = new StringBuilder();

            for (int j = 0; j <= nameLettersCount; j++) {
                name.append(letterGenerator());
                counter++;
            }

            name.append(getEnding(gender));

  if (startingText.isEmpty() ||
String.valueOf(startingText.charAt(startingText.length() - 1)).equals(" ")){
                name.replace(0, 1,
String.valueOf(Character.toUpperCase(name.charAt(0))));
            }
```

```
            name.insert(0, startingText);
            name.append(endingText);
            namesArr[i] = String.valueOf(name);
        }

        return namesArr;
    }

    private static String letterGenerator() {
        random = new Random();
        String[] lettersArr = random.nextDouble() <= 0.95 ? getLettersArr() :
getLettersCombinationArr();

        random = new Random();
        return lettersArr[random.nextInt(lettersArr.length)];
    }

    private static String[] getLettersArr() {
        if (random.nextDouble() <= 0.5 && consonantsInRowGenerated == 0){
            consonantsInRowGenerated++;
            vowelsInRowGenerated = 0;

  if(random.nextDouble() <= 0.05){
                return UALanguagesLettersAndCombinations.getUniqueConsonants();
            }

            return UALanguagesLettersAndCombinations.getConsonants();
        } else if (vowelsInRowGenerated == 0){
```

```
            consonantsInRowGenerated = 0;
            vowelsInRowGenerated++;



            if(random.nextDouble() <= 0.05){
                return UALanguagesLettersAndCombinations.getUniqueVowels();
            }

            return UALanguagesLettersAndCombinations.getVowels();
        }else return getLettersArr();
    }

    private static String[] getLettersCombinationArr(){
        if (random.nextDouble() <= 0.5 && consonantsInRowGenerated == 0){
            consonantsInRowGenerated++;
            vowelsInRowGenerated = 0;
            return UALanguagesLettersAndCombinations.getConsonantCombinations();
        }
        else if(vowelsInRowGenerated == 0){
            consonantsInRowGenerated = 0;
vowelsInRowGenerated++;
            return UALanguagesLettersAndCombinations.getVowelsCombinations();
        }else return getLettersCombinationArr();
    }

    private static String getEnding(String gender){
        random = new Random();
        String []endingsArr;
```

```
        if(gender.equals(maleGender)){
            if(consonantsInRowGenerated == 0){
                endingsArr                                                    =
UALanguagesLettersAndCombinations.getConsonantsStartingMaleEndings();
            }else{
                endingsArr                                                    =
UALanguagesLettersAndCombinations.getVowelsStartingMaleEndings();
            }

            return endingsArr[random.nextInt(endingsArr.length)];
        }else if(gender.equals(femaleGender)){
            if(consonantsInRowGenerated == 0){
                endingsArr                                                    =
UALanguagesLettersAndCombinations.getConsonantsStartingFemaleEndings();
            }else{
                endingsArr                                                    =
UALanguagesLettersAndCombinations.getVowelsStartingFemaleEndings();
            }


            return endingsArr[random.nextInt(endingsArr.length)];
        }

        return letterGenerator();
    }

    @Contract(pure = true)
    private static int getMaxCountOfElements(@NonNull String
```

```
countOfGeneratedElements){
        switch (countOfGeneratedElements){
            case "малий":
                return 3;

            case "середній":
                return 5;

            default:
                return 7;
        }
    }

    @Contract(pure = true)
    private static int getMinCountOfElements(@NonNull String
countOfGeneratedElements){
        switch (countOfGeneratedElements){
            case "середній":
                return 3;

            case "великий":
                return 5;

            default:
                return 1;
        }
    }
}
```

Код файлу EnglishCreationsAlgorithm.java

```
package NamesCreator;

import androidx.annotation.NonNull;
import org.jetbrains.annotations.Contract;

import java.io.Serializable;
import java.util.Random;

public class EnglishCreationsAlgorithm extends NamesCreationAlgorithmRunner implements Serializable{
    private static Random random;
    private static int vowelsInRowGenerated = 0;
    private static int consonantsInRowGenerated = 0;
    private static final String maleGender = "m", femaleGender = "f";

@NonNull
    public String[] generateNames(int countOfNames, String countOfGeneratedElements, String gender, String startingText, String endingText){
        String[] namesArr = new String[countOfNames];
        random = new Random();

        int maxCountOfElements = getMaxCountOfElements(countOfGeneratedElements);
        int minCountOfElements = getMinCountOfElements(countOfGeneratedElements);
```

```
        for (int i = 0; i < namesArr.length; i++) {
            int nameLettersCount = random.nextInt(maxCountOfElements -
minCountOfElements + 1) + minCountOfElements;
            StringBuilder name = new StringBuilder();

            for (int j = 0; j <= nameLettersCount; j++) {
                name.append(letterGenerator());
            }

            name.append(getEnding(gender));

            if (startingText.isEmpty() ||
String.valueOf(startingText.charAt(startingText.length() - 1)).equals(" ")){
                name.replace(0, 1,
String.valueOf(Character.toUpperCase(name.charAt(0))));
            }

            name.insert(0, startingText);
            name.append(endingText);
            namesArr[i] = String.valueOf(name);
        }

        return namesArr;
    }

    private static String letterGenerator() {
        random = new Random();
        String[] lettersArr = random.nextDouble() <= 0.95 ? getLettersArr() :
```

```
getLettersCombinationArr();

        random = new Random();
        return lettersArr[random.nextInt(lettersArr.length)];
    }

    private static String[] getLettersArr() {
        if (random.nextDouble() <= 0.5 && consonantsInRowGenerated == 0){
            consonantsInRowGenerated++;
            vowelsInRowGenerated = 0;
            return EngLanguagesLettersAndCombinations.getConsonants();
        } else if (vowelsInRowGenerated == 0){
            consonantsInRowGenerated = 0;
            vowelsInRowGenerated++;
            return EngLanguagesLettersAndCombinations.getVowels();
        }else return getLettersArr();
    }

    private static String[] getLettersCombinationArr(){
        if (random.nextDouble() <= 0.5 && consonantsInRowGenerated == 0){
            consonantsInRowGenerated++;
            vowelsInRowGenerated = 0;
            return EngLanguagesLettersAndCombinations.getConsonantCombinations();
        }
        else if(vowelsInRowGenerated == 0){
            consonantsInRowGenerated = 0;
            vowelsInRowGenerated++;
```

```
            return EngLanguagesLettersAndCombinations.getVowelsCombinations();
        }else return getLettersCombinationArr();
    }

    private static String getEnding(String gender){
        random = new Random();
        String []endingsArr;

        if(gender.equals(maleGender)){
            if(consonantsInRowGenerated == 0){
                endingsArr =
EngLanguagesLettersAndCombinations.getConsonantsStartingMaleEndings();
            }else{
                endingsArr =
EngLanguagesLettersAndCombinations.getVowelsStartingMaleEndings();
            }

            return endingsArr[random.nextInt(endingsArr.length)];
        }else if(gender.equals(femaleGender)){
            if(consonantsInRowGenerated == 0){
                endingsArr =
EngLanguagesLettersAndCombinations.getConsonantsStartingFemaleEndings();
            }else{
                endingsArr =
EngLanguagesLettersAndCombinations.getVowelsStartingFemaleEndings();
            }

            return endingsArr[random.nextInt(endingsArr.length)];
```

Продовження додатку А

```
    }

        return letterGenerator();
    }

    @Contract(pure = true)
    private static int getMaxCountOfElements(@NonNull String
countOfGeneratedElements){
        switch (countOfGeneratedElements){
            case "small":
                return 3;
            case "middle":
                return 5;

            default:
                return 7;
        }
    }

    @Contract(pure = true)
    private static int getMinCountOfElements(@NonNull String
countOfGeneratedElements){
        switch (countOfGeneratedElements){
            case "middle":
                return 3;

            case "big":
                return 5;
```

```
            default:
                return 1;
        }
    }
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: “Розроблення мобільного застосунку з офлайн-доступом”

Обсяг пояснювальної записки: 85 аркушів

Дата закінчення роботи червня 2026р.

Підпис _____