Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інструментарію для створення мобільних додатків»

Студента 4 курсу, 10 групи,

спеціальності
122 «Комп'ютерні науки»

_____
підпис студента

Радаловський Вадим Вікторович

Науковий керівник
кандидат фізико-математичних наук

_____
підпис керівника

Філімонова Тетяна Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент

_____
підпис керівника

Демідов Павло Георгійович

Київ 2021
Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«18» грудня 2020р.

Завдання
на випускну кваліфікаційну роботу (проект) студенту

Радаловський Вадим Вікторович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка інструментарію для створення мобільних додатків»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*
2. Строк здачі студентом закінченої роботи *31 травня2021 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: обґрунтування та розробка інструментарію для створення мобільних додатків.
Об'єкт дослідження: процес розробки інструментарію для створення мобільних додатків.
Предмет дослідження: засоби створення мобільних додатків.
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
2	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
3	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Аналітичне дослідження загального (міжнародного) ринку мобільних додатків на операційних системах IOS та Android

1.1 Основні теоретичні відомості та операційні системи мобільних девайсів

1.2 2020 рік в динаміці - основні зміни, тенденції. Детальний аналіз на основі звіту дослідницької компанії App Annie – State of Mobile 2021

1.3 Тренди та прогнози розвитку ринку на 2021 рік

РОЗДІЛ 2. Вибір та створення моделі роботи допоміжного інструменту для розробки мобільних додатків

2.1 Аналіз інструментів для розробки та підтримки

2.2 Розробка моделі роботи допоміжного інструменту для розробки мобільних додатків

РОЗДІЛ 3. Створення допоміжного інструменту для розробки мобільних додатків

3.1 Аналіз використаних в процесі роботи зовнішніх бібліотек Python

3.2 Програмна реалізація допоміжного інструменту для розробки мобільних додатків

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.10.2020*	*01.10.2020*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*18.12.2020*	*18.12.2020*
3	*Вступ*	*03.02.2021*	*26.04.2021*
4	*РОЗДІЛ 1. Аналітичне дослідження загального (міжнародного) ринку мобільних додатків на операційних системах IOS та Android*	*26.02.2021*	*26.02.2021*
5	*РОЗДІЛ 2. Вибір та створення моделі роботи допоміжного інструменту для розробки мобільних додатків*	*06.04.2021*	*06.04.2021*
6	*РОЗДІЛ 3. Створення допоміжного інструменту для розробки мобільних додатків*	*12.05.2021*	*12.05.2021*
7	*Висновки*	*14.05.2021*	*14.05.2021*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*20.05.2021*	*26.05.2021*
9	*Попередній захист випускної кваліфікаційної роботи*	*26.05.2021*	*03.06.2021*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*27.05.2021*	*07.06.2021*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*31.05.2021*	*10.06.2021*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «22» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи (проекту)
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник

Радаловський В.В..

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи (проекту)

Керівник випускної кваліфікаційної роботи (проекту)

_____ 31.05.2020 р.

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота (проект) студента Радаловського В.В.

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Демідов П.Г.

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2021 р.

АНОТАЦІЯ

У випускній кваліфікаційній роботі створено допоміжний інструмент для розробки – систему для фіксації й прорахунку часу, що витрачається на користування додатками для подальшого аналізу ефективності процесу розробки на основі отриманих даних. Здійснено аналіз ринку мобільних додатків у хронології з детальним аналізом ринку станом на кінець 2020-ого року. Проаналізовано інструменти для розробки мобільних додатків за направленням функціоналу та послуг, що вони надають.

Ключові слова: мобільний додаток, ринок мобільних додатків, операційна система, інструмент для розробки.

Anotation

The graduation qualification work is devoted to creation support tool for development a system for recording and calculating the time spent on using applications for further analysis of the effectiveness of the development process based on the obtained data. Analysis of the market of mobile applications in chronology with a detailed market analysis at the end of 2020 has been done. An analysis of application distribution systems was done.

Keywords: mobile application, mobile applications market, operating system, , development tool.

ЗМІСТ

ВСТУП

Актуальність теми: в сучасному світі у абсолютної більшості є власні мобільні телефони. Однак наразі мобільний зв'язок є далеко не єдине на що здатні, і що вже роблять сучасні мобільні девайси. Швидкий розвиток технологій дозволив помістити величезну кількість можливостей у малий корпус й перетворити звичайний комунікатор у «кишеньковий комп'ютер». Але мобільний девайс, сам по собі залишається, в цілому, лише комунікатором. Саме мобільні додатки представляють собою інструменти націлені на вирішення потреб користувачів надаючи величезну кількість послуг – від фінансових послуг та стрічок новин до перегляду та створення медіа контенту за допомогою мобільних телефонів.

У зв'язку з розмірами сформованої користувацької бази ринок мобільних додатків став ринком з дуже високим рівнем конкуренції де, для побудування прибуткового бізнесу, дуже важливим є створення та видання якісного таргетованого додатку швидше за можливих конкурентів.

Особлива актуальність теми пов'язана з пандемією COVID-19, та необхідністю перенесення звичайних побутових активностей (як шопінг та комунальні послуги) в онлайн формат, зручний для користувача.

Мета роботи: обґрунтування та розробка інструментарію для створення мобільних додатків.

Досягнення мети обумовило необхідність вирішення таких завдань:

- комплексне аналітичне дослідження ринку мобільних додатків станом на кінець 2020 року – початок 2021 року.
- дослідження існуючих інструментів для розробки мобільних додатків.
- розробка моделі допоміжного інструменту для оптимізації процесу розробки мобільних додатків.
- створення допоміжного інструменту для оптимізації процесу розробки мобільних додатків.

Об’єкт дослідження: процес розробки інструментарію для створення мобільних додатків.

Предмет дослідження: засоби створення мобільних додатків

Методи дослідження: загальнонауковий аналітичний метод, системний метод, метод програмування.

Інформаційна база дослідження: статистичні дані, що надаються у вільний доступ дослідницькими компаніями, публікації експертів області як технічних, так й економічних, маркетингових спеціальностей, пов’язаний напряму з досліджуваною темою, публікації та новини від діючих компаній на ринку, технічна документація.

Практичне значення: отримані результати дослідження можуть бути використані для створення мобільного додатку, для визначення категорії створюваного додатку, його цільової аудиторії, операційної платформи. Розроблену програму можна використовувати в процесі основної технічної розробки програми для побудови ефективного процесу розробки, командної взаємодії та фінансової ефективності процесу, в залежності від умов використання.

РОЗДІЛ 1. Аналітичне дослідження загального (міжнародного) ринку мобільних додатків на операційних системах IOS та Android

1.1. Основні теоретичні відомості та операційні системи мобільних девайсів

Ринок мобільних додатків є відносно новим й активно розвивається спільно ринком мобільних девайсів. Відправною точкою для створення мобільних додатків стала поява на мобільному телефоні екрану. Перше програмне забезпечення для телефонів представляло собою вбудовані додатки, які призначалися для виконання конкретних функцій телефону і встановлювалися в пристрій самими виробниками.

Час появи перших мобільних додатків, встановлених на телефон поверх вже наявного програмного забезпечення, можна віднести до кінця 90-х років минулого століття. Тоді в програмну оболонку телефонів, крім самих необхідних додатків, виробники стали встановлювати додаткове ПЗ.

Основою для будь якої взаємодії користувача та додатків для девайсів є операційна система. На протязі всього періоду часу з 90-их років по 2021-ий рік були створено й випущені девайси з великою кількістю операційних систем, таких як BlackBerry OS, Symbian, Windows Phone. Однак на кінець 2021-ого року основними є лише 2 операційні системи: Google Android та Apple iOS.

IOS (айос) - це мобільна операційна система, створена і розроблена компанією Apple виключно для своїх пристроїв. На її основі працюють iPhone і iPod Touch, раніше встановлювалася і на iPad до появи iPadOS в 2019 році. Є другою за популярністю операційної мобільною системою відразу після Android (Андроїд) [1].

Операційна система надалі кожен рік розвивається додаючи нові функції та зручності для користувача. Останньою версію є iOS 14, що вийла у Вересні 2020-ого року. Операційна система має наразі низку особливостей [2]:

- Швидкодія – ОС гарно оптимізована, тому, незважаючи на більш «слабкі» технічні компоненти, у більшості випадків ОС працює без збоїв та має відмінну швидкість роботи.
- Зручність користування – має мінімум необхідних налаштувань для зручного використання.
- Інтуїтивний інтерфейс та легке управління – зручний інтерфейс з малими можливостями до кастумізації.
- Безпека та конфіденціальність – є закритою системою та неможливістю впливу встановлених програм на систему та файлу користувача без прямої згоди від користувача.
- Підтримка людей з обмеженими можливостями – має велику кількість функції для підтримки людей з обмеженими можливостями, одна з яких – голосовий помічник Siri за допомогою якого можна керувати всією системою та її функціями.

Однак варто додати, що iOS дуже консервативна платформа, що вкрай рідко має значні зміни в структурі і можливостях. Також операційна система закрита, тобто не є можливим встановлення програм через сторонні ресурси, лише через App Store. Та має малу кількість налаштувань – пристрій та його функції неможливо налаштувати «під себе». Незважаючи на великий вибір програм в App Store – вони є обмеженими в можливостях й не здатні в значній мірі внести зміни у основні попередньо встановлені додатки, що йдуть в комплекті з самим пристроєм.

App Store є скороченням від Application Store, що буквально означає магазин додатків. Особливості операційної системи мобільних пристроїв Apple

такі, що установка програм можлива тільки з одного місця - магазину додатків, в якому зібрані попередньо перевірені фахівцями Apple програми [3].

Android - це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в основному для сенсорних мобільних пристроїв, таких як смартфони та планшети. Android розробляється консорціумом розробників, відомим як Open Handset Alliance [4]. Він був представлений у листопаді 2007 року. Альянс об'єднував великих бездротових операторів, виробників пристроїв і виробників чіпсетів, а, крім самої Google, в нього увійшли HTC, Motorola, Samsung, T-Mobile і Qualcomm [5].

Операційна система Android використовується в своїх пристроях багатьма брендами. Основні з них (відсортовані по рейтингу від найпопулярніших): Samsung, Huawei, Xiaomi, OPPO, Vivo, Motorola, Lenovo, LG, Nokia [6].

На відміну від Apple iOS Android OS є відкритою платформою. Це й є основною як перевагою, так і недоліком операційної системи.

Переваги та недоліки такого підходу:

- + Широкі можливості налаштувань, як інтерфейсу так і функцій девайсу.
- - Готові пакети від виробників для налаштувань користувачеві часто не мають великого спектру можливостей, а додатки, що ставляться «зверху» на основну систему додатково завантажують систему.
- Можливість використовувати сторонні ресурси для завантаження та встановлення додатків.
- - Сторонні ресурси часто є розповсюджувачами шкідливого ПО.
- + Операційна система гнучко налаштовується самими виробниками й значно відрізняється у кожного окремого виробника. Об’єднує їх одна

структура системи та набір функцій. Немає обмежень, щодо вдосконалення операційної системи самим виробником девайсу.

- - Не кожний виробник має ресурси для відладженого запуску свого продукту, тому якість кінцевої Операційної Системи, що отримує користувач може значно відрізнятися в залежності вид виробника девайсу.

Основним магазином на операційній системі Android є Google Play Store. Google Play, що раніше називався Android Market, - це сервіс цифрового розповсюдження, що управляється та розробляється Google. Він служить офіційним магазином додатків для сертифікованих пристроїв, що працюють під управлінням операційної системи Android, що дозволяє користувачам переглядати та завантажувати програми, розроблені за допомогою набору для розробки програмного забезпечення Android (SDK) та опубліковані через Google. Google Play також служить магазином цифрових медіа, пропонуючи музику, книги, фільми та телевізійні програми [7].

Окремо від Google Play на операційній системі Android існують також інші джерела мобільних додатків. Це зумовлено відкритістю платформи та можливістю встановлення додатків з інших ресурсів, окрім магазину Google. Це можуть бути, як окремі файли для встановлення додатків в інтернеті, так й повноцінні магазини додатків від інших компаній, такі як: Aptoide, F-Droid, Samsung Galaxy Store, Huawei AppGallery, GetApps [8 - 10]

За даними Kantar розподіл використання мобільних операційних систем на кінець 2020-ого року є таким (Таблиця 1.1) [11]:

Розподіл використання мобільний операційних систем у 2020 році по країнам

Таблиця 1.1

	США	Мексика	Бразилія	Великобританія	Іспанія	Франція	Італія	Німеччина	КНР	Японія	Австралія	**Середнє**
Android	51,70%	96,20%	93,60%	53,30%	84,80%	75,40%	82,10%	74,40%	83,30%	50,60%	53,30%	**72,61%**
BlackBerry	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,20%	0,00%	0,00%	0,00%	**0,02%**
iOS	48,30%	3,50%	5,30%	46,70%	15,00%	24,50%	17,70%	25,30%	16,60%	45,90%	46,70%	**26,86%**
Windows	0,00%	0,00%	0,80%	0,00%	0,20%	0,10%	0,20%	0,10%	0,00%	0,00%	0,00%	**0,13%**
Інші	0,00%	0,30%	0,30%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	3,60%	0,00%	**0,38%**

Кількість користувачів iPhone у вересні 2020 перейшло за 1 млрд [12].

Не маючи чітких даних, що до кількості користувачів Android (бо операційна система використовується одночасно великою кількістю виробників мобільних телефонів) – можемо припустити, що їх кількість – 2,7 млрд

В результаті на кінець 2020-ого року користувацька база

iOS (iPhone) – 1 млрд

Android – 2.7 млрд

Це не точні дані, але вони показують в загальному обсяги користувацьких баз на сучасних мобільних платформах.

1.2. 2020 рік в динаміці - основні зміни, тенденції. Детальний аналіз на основі звіту дослідницької компанії App Annie – State of Mobile 2021 [13]

На відміну від ринку мобільних девайсів, де у 2020 році відбувся помітний спад, ринок мобільних додатків мав зріст одразу багатьох показників. За рахунок карантинних заходів значно збільшився час, що користувачі проводили у мобільних додатках, а за цим показником зросли й інші.

- Час користування додатками в день на користувача на девайсах з ОС Android (Daily Time Spend Per User) – 4.2 години – на 20% більше ніж у 2019 році (Рис.1.3).
- Нові завантаження додатків (New App Downloads) – 218 млрд – на 7% більше ніж у 2019 році

- Прибуток від додатків – прямі продажі додатків та покупки всередині додатків (App Store Spend) – 143 млрд $ – на 20% більше ніж у 2019 році
- Прибуток від реклами (Mobile Ad Spend) – 240 млрд $ – на 26% більше ніж у 2019 році
- Венчурний капітал у мобільні технології (Venture Capital to Mobile Tech) – 73 млрд $ – на 27% більше ніж у 2019 році.

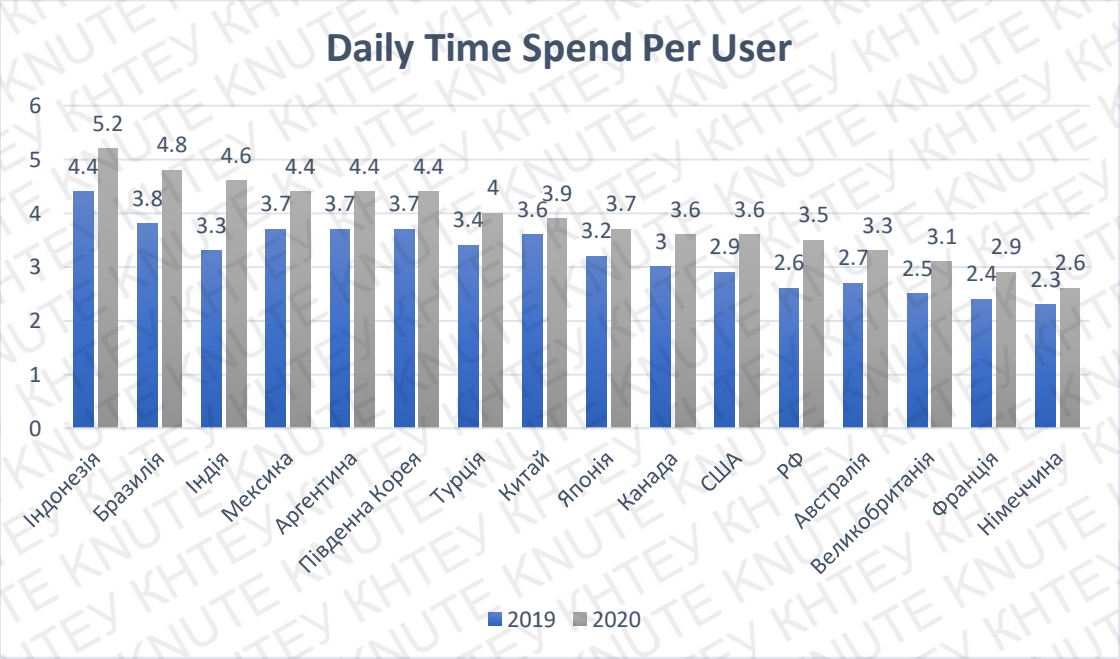

Рис.1.3 Кількість використаних годин на користувача в день по країнам за 2020 рік

На відміну від 2019 року, коли був одночасно й спад попиту на мобільні девайси, у 2020 році сума інвестицій у технології мобільних додатків збільшилась на 27%. Основними інвесторами були фінансові сервіси, комерційні, транспортні та торговельні компанії. Основні напрямки для капіталовкладень – інноваційні споживчі та підприємницькі технології з використанням гео-локації користувачів, хмарні сервіси та Штучний Інтелект (Artificial Intelligence) (Рис.1.5).

Рис.1.5. Інвестиції в технології мобільних додатків з 2011 по 2020 рік

За віком серед користувачів додатків виділяють 3 групи:

- Покоління Z (Gen Z) – віком від 16 до 24 років.
- Мілленіали (Millennials) – від 25 до 44 років.
- Покоління X (Gen X) – від 45 років.

В діаграмі нижче наведений їх розподіл, в залежності від країни (Рис.1.6):

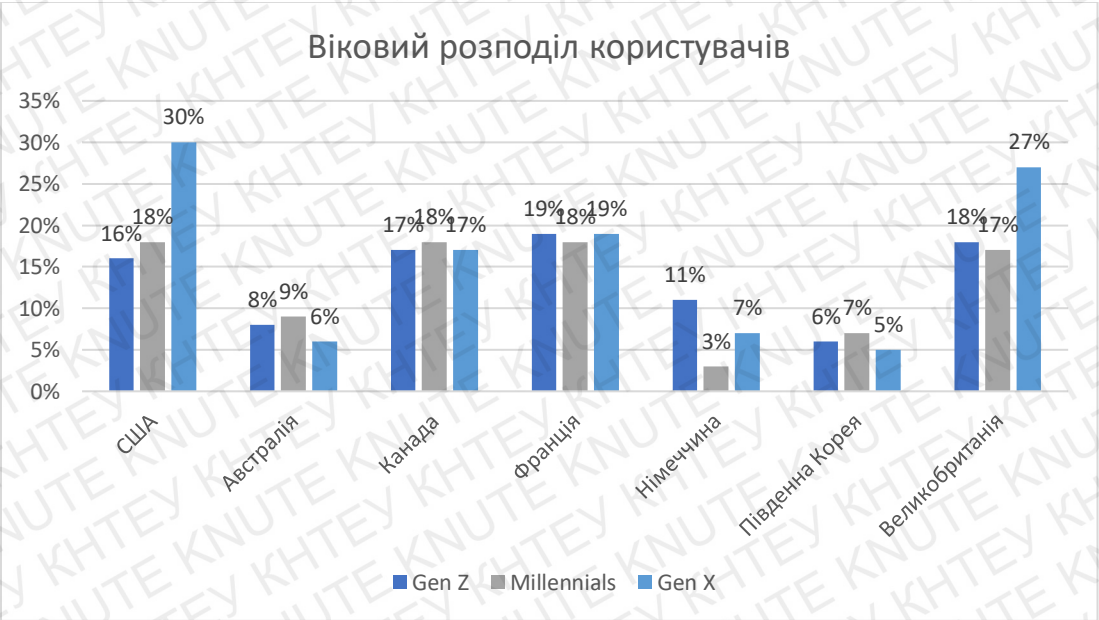

Рис.1.6 Віковий розподіл користувачів по групам за даними 2020 року

Розглянемо загальні тенденції в розрізі вікових груп у наведених країнах за рейтингами додатків з найбільшою кількістю користувачів.

Покоління Z

Одним з найпопулярніших додатків є соціальна мережа Snapchat. Вона знаходиться на першій позиції в рейтингу за кількістю користувачів в віковій категорії 16-24 одразу в декількох країнах – США, Австралія, Канада, Німеччина та на другій позиції в Великобританії. Велику популярність має стримінговий сервіс Twitch – він займає 2 позиції в рейтингу Німеччини, Південної Кореї, США, Канади та перше місце в Великобританії. Також варто виділити соціальну платформу TikTok (3 позиція в США, 4 у Франції та 5 у Великобританії) та сервіс для онлайн комунікації Discord відеоігрового напрямку (1 позиція в Південній Кореї, 2 в Австралії та Франції, 3 в Канаді та Німеччині). Варто додати, що, незважаючи на початкову цільову аудиторію, на протязі 2020 року компанія вела активне поширення свого додатку серед нової аудиторії, як додатку для робочої та учнівської комунікації у період карантину.

Мілленіали

Серед цієї вікової групи більшість додатків відрізняються в залежності від регіону, але основна тенденція – це додатки робочого направлення та фінансові додатки. Виділити варто соціальну мережу LinkedIn, що має бізнес направлення й займає 1 місце в рейтингу у Канаді, 2 місце у США, Франції, Австралії та 5 місце у Великобританії. Та додаток Discord, що зайняв першу позицію в США за 2020 рік.

Покоління X

Характеризується використанням допоміжних додатків та сервісів – додатки «розумного будинку», новини, погода, тощо.

Якщо розглядати саме відеоігровий сегмент, то в ньому типи додатків діляться на декілька категорій:

Казуальні ігри (Casual) – ігри спрощені за свою структурою за вимогами до навичок користувача, націлені на максимальну аудиторію.

Хардкорні (Hardcore / Core) – мають функціонал, що вимагає певних навичок від користувача. Зазвичай націлені на свою нішову аудиторію, в залежності від жанру, але є й виключення.

Мобільні азартні ігри (Casino / Mobile Gambling) – азартні ігри, що використовують мобільні девайсу, як платформу.

Казуальні ігри мають найбільший охват аудиторії, але мають особливість невеликих фінансових вкладень користувачів, не зважаючи на високі показники проведеного часу в іграх. Але варто вказати, що незважаючи на меньші фінансові вкладення ніж у Хардкорних ігор, за рахунок кількості користувачів й проведененого ними часу в додатках є гарною цільовоє для зовнішньої реклами вмонтованої у самі додатки.

Також важливо, що не зважаючи на дуже малу аудиторію мобільних азартних ігор – на них випадає 11% від усіх користувацьких витрат в мобільних іграх (Рис.1.7).

Рис.1.7. Доля додатків за категоріями по показникам завантажень, користувацьких витрат, та часу в додатках

Додатки для виконання фінансових операцій (банкінг, додатки для сплати податків, дорогих покупок, такі як квартира, машина, тощо та додатки для інвестицій) мали помітний зріст у 2020 році. Під час дії карантинних обмежень все більше людей надають перевагу дистанційним онлайн методам виконання звичних фінансових операції. Спад завантажень відбувся лише в Китаї по причині змін в законодавстві, що до однорангового кредитування (Рис.1.8).

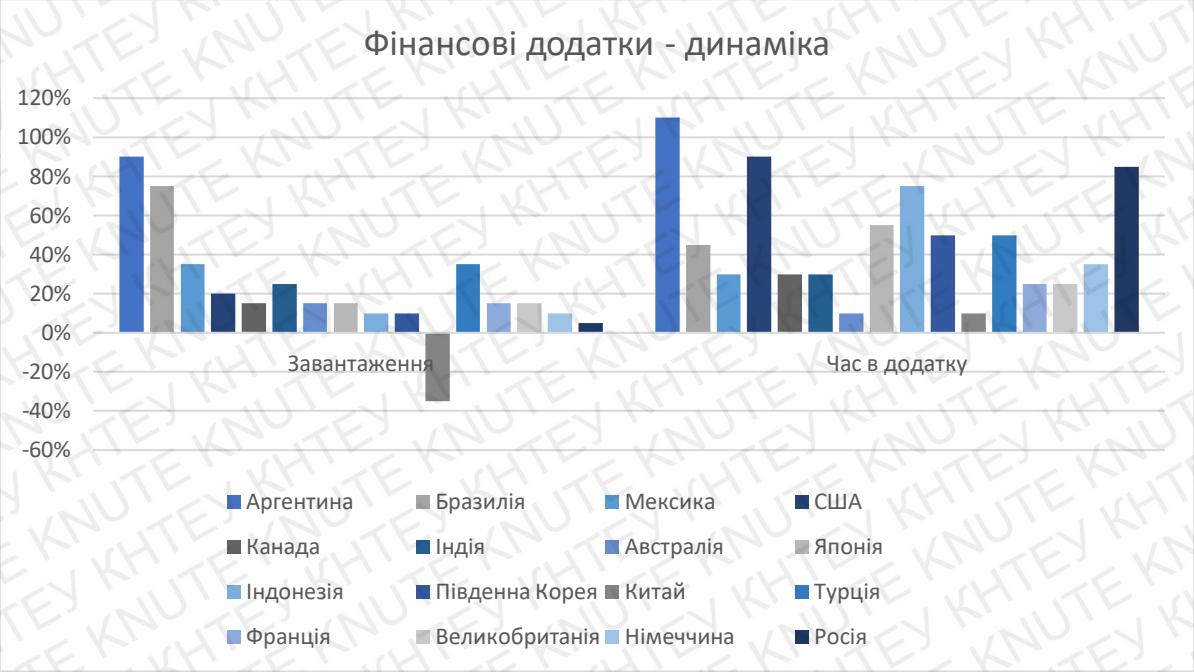

Рис.1.8. Зміни в категорії фінансових додатків 2019-2020 роки

Час, що в середньому витрачався користувачем на мобільні соціальні мережі збільшився майже для всіх додатків, без залежності від регіону та платформи.

Лідарами цієї групи додатків є TikTok, Instagram, Telegram, WhatsApp, Twitter, Discord, Facebook, Pinterest, Twitter. Саме ці додатки загалом ділять між собою п’ятірку найбільш популярних мобільних соціальних мереж, в залежності від країни (за виключенням Китаю) (Додаток 2).

Також 2020 рік є роком з найбільшими показниками мобільного шопінгу. За даними CNBC за час Фестивалю Шопінгу (Shopping Festival) через платформи компанії Alibaba та інші торгівельні платформи були здійснені покупки загальною сумую у $115 млрд. Фестиваль тривав з 1 по 11 листопада.

В США за період з 1 листопада по 9 грудня через сервіси мобільного шопінгу було зроблено покупок на $53,2 млрд, що на 55% більше у порівнянні з 2019 роком.

Загальний час проведений в мобільних додатках для шопінгу на платформі Android виріс на 30% у порівнянні з 2019 роком. В Американському регіоні першою країною за цим показником були США, в Азії – Китай, та Росія у Європі.

Основні платформи для соціальної комерцій (комерційна діяльність, що активно виокристовує соціальні мережі не тільки для маркетинга, але й для прямих продаж через соціальні мережі) – Instagram та Pinterest мали приріст:

Instagram – на 20% з 414 млн завантажень до 495 млн

Pinterest – на 49% з 129 млн до 193 млн (Рис.9).

Активно продовжив розвиватися сегмент лайв-шопінгу. Формат live-commerce - новий стиль шопінгу через стріми, модель, яка об'єднує пряме включення продавця або блогера з можливістю робити покупки в реальному часі. Лідерами цього напрямку є Китай з додатком TaoBao Live – кількість завантажень зросла на 101% з 3,90 млн до 7,87 млн завантажень (Рис.1.10).

За прогнозами technavio вже в 2024 році соціальна та лайв комерцій матиме загальний ринок об’ємом у $2 трлн.

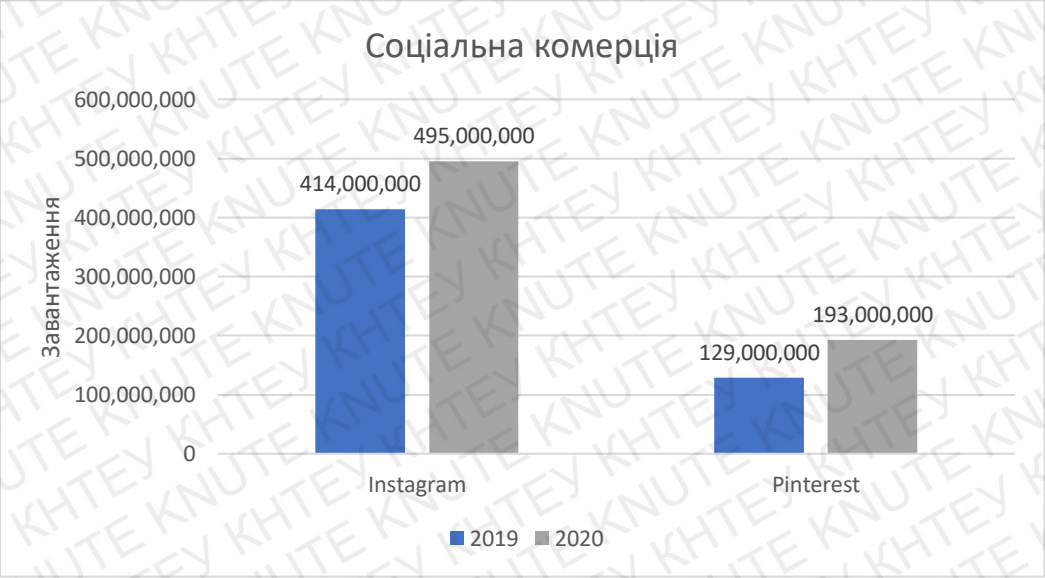

Рис.1.9. Дані по кількісті завантажень Instagram та Pinterest за 2019-2020 роки – найбільш популярних майданчиків соціальної комерції

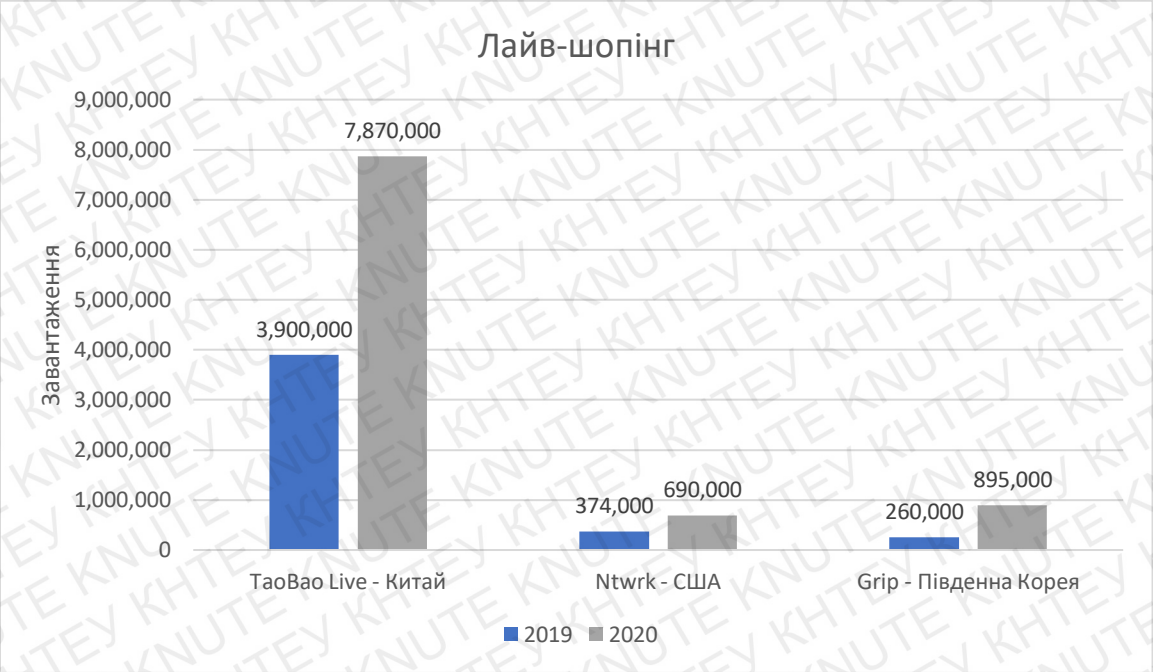

Рис.1.10 Дані по кількості завантажень найбільш популярних додатків лайв-шопінгу

Сервіси с доставки їжі активно розвивалися вже в 2019 році, але за умов, створених пандемією темпи лише збільшились. В більшості країн кількість сесій у додатках цієї категорії почали стрімко збільшуватись у другому та третьому кварталах 2020 року й досягли максимум у четвертому кварталі.

В США, Аргентині, Великобританія, Індонезії та Росії ріст в порівнянні с четвертим кварталом 2019 року дорівнював 60%, 65%, 70%, 80% та 105% відповідно.

Однак була й негативна тенденція. З початком пандемії через логістичні обмеження був помітний спад попиту в таких країнах як Індія, Китай, Турція, Великобританія, Франція. Найбільш значний спад відбувся в Індії. Хоч Індія як була так і залишається лідером з попиту на послуги доставки їжі – показники користувацьких сесій на кінець 2020 року все ще не досягли показників до початку пандемії. Кількість користувацьких сесій в Індії на платформі Android у

період з 05.01.2020 по 12.01.2020 – 883 млн. У період з 20.12.20 по 27.12.20 – 660 млн (Рис.11).

Рис.1.11 Динаміка популярності додатків з доставки їжі за 2020 рік

Цікава ситуація склалася серед додатків для подорожей. Під впливом пандемії люди почали шукати альтернативні місця для відпочинку – ця тенденція дуже помітна в США, де кількість завантажень на recreation.gov(національні парки) та Geocaching (пошук точок інтересу з соціальними елементами) зросла на 255% та 75% відповідно. Сумарна кількість завантажень цих додатків склала 2.7 млн

У Південній Кореї показується інша тенденція, а саме – альтернативні види транспорту. Додатки Xing Xing та LimeBike спрямовані на оренду електричних скутерів та велосипедів, мали приріст більший за 300%. В сумі кількість їх завантажень становить 2.6 млн

Великобританія поєднала одразу два тренди, тут, хоч з відносно невеликим приростом у порівнянні з 2019 роком лідерство займаються додатки OS Maps:

Walking & Bike Trails (велосипедні та піші маршрути для прогулянок) та Santander Cycles (оренда електровелосипедів) (Рис.1.12).

Рис.1.12. Приріст завантажень додатків для подорожей у порівнянні з 2019 роком

Ще декілька таблиць з рейтингами додатків та компаній-видавників на кінець 2020 року вкладені до роботи у вигляді додатків (Додаток 3, додаток 4, додаток 5).

1.3 Тренди та прогнози розвитку ринку на 2021 рік

2020 рік вніс дуже багато змін я у структуру ринку в загальному, так й у сабі внутрішні процеси щодо розробки та реалізації. Пандемія COVID-19, однозначно є дуже значущим чинником цих змін, однак не єдиним.

Одним з вже наявних наслідків судового процесу між Apple та Epic Games є зниження комісії App Store для додатків з річним прибутком меншим за 1$ млн – до 15% від прибутку замість 30%, що були й залишаються для компаній з прибутком від $1 млн. За прогнозами варто очікувати збільшення кількості малих компаній/стартапів й, відповідно, посилення рівня конкуренції на

платформі iOS. Також з посиленням конкуренції можуть зрости й рекламні ставки, що, в результаті, призведе до того ж рівня чистого прибутку малих додатків, що й був до внесення змін в систему комісії.

Були також внесені зміни у політику безпеки та конфіденційності на девайсах Apple. Для ідентифікації користувача рекламною компанією й збору даних використовується IDFA – унікальний номер, що допомагає зв'язати встановлення додатку з конкретним девайсом. Починаючи з 2021 року з новим оновленням операційної системи буде обов'язковим запит на дозвіл використання ідентифікатору у користувача, а додатки, що не будуть виконувати вимоги будуть видалені з магазину. Як наслідок – збільшення цін на рекламу в Apple Search Ads та збільшення ролі ASO – оптимізації додатку з ціллю його максимальної видимості в пошуковій системі. Одночасно з цим збільшиться роль якості додатків. А поява нової пошукової системи в магазині від Apple буде вимагати швидкої реакції та дій для адаптації пошуку по ключовим словам.

2021 році недостатньо просуватися тільки через App Store, Google Play і використовувати стандартні рекламні канали (такі, як Facebook і Google Ads). Щоб залучити аудиторію і отримувати більше установок, потрібно використовувати альтернативні рекламні майданчики і магазини додатків. Як можна побачити за статистикою 2020 року – дуже популярними є нові соціальні мережі, такі як Tik-Tok, які не є стандартними майданчиками для реклами додатків/брендів. Дуже важливо буде інвестувати в аудиторію і створювати ком'юніті навколо програми: вкладати в SMM, email-маркетинг, сайт [14].

Зі сторони самої розробки додатків можна виділити такі тренди [15]:

Інтеграція з Інтернетом Речей (Internet of Things / IoT).

IoT - далеко не нова концепція. Але зростання кількості мобільних телефонів у широкому діапазоні секторів та категорій створило, здавалося б, безмежні можливості для Інтернету Речей. Люди звикли використовувати

технології для поліпшення свого повсякденного життя. IoT описує зростаючу мережу пристроїв, підключених до Інтернету, забезпечуючи зручність та автоматизований контроль для споживачів. Технологія розумного будинку є прекрасним прикладом зростання розвитку Інтернету Речей та розвитку мобільних додатків. Мобільні програми можна використовувати для регулювання температури в будинку з віддаленого місця, блокування або розблокування вхідних дверей та підключення до систем домашньої безпеки. Холодильники та інші побутові прилади також можна підключити до мобільних додатків. Очікується, що глобальний ринок Інтернету Речей досягне $222 млрд у 2021 році. $161 млрд з цих оцінок надходитиме від програмного забезпечення, наприклад мобільних додатків.

Програми для складних пристроїв.

У 2019 році вийшли складні пристрої, такі як Samsung Galaxy Fold, Huawei Mate X та нова Motorola Razr. Ці смартфони складаються, щоб стиснути або розширити розмір екрану залежно від уподобань користувачів. Наприклад, користувач може зателефонувати із закритим пристроєм, але переглянути відео на великому екрані, розгорнувши пристрій. З точки зору розробки додатків, торговельні посередники та творці вмісту повинні враховувати ці пристрої під час створення або оновлення програми. Ідея полягає в тому, що програма повинна плавно регулювати свій дисплей у міру складання або розгортання екрана. Зараз складні пристрої - це лише частинка загальної частки ринку смартфонів. Але це зміниться в найближчі роки. Згідно з дослідженням USA Today за 2019 рік, 17% користувачів iPhone і 19% користувачів Android раді придбати телефон зі складним дизайном. За даними Statista, в 2019 році було відправлено приблизно 3,2 мільйони складних телефонів. Очікується, що цей прогноз досягне 50 мільйонів одиниць до 2022 року.

Мобільна комерція

Ця тенденція домінує у 2019, 2020 і продовжуватиме процвітати в 2021 році. Функціональність мобільної електронної комерції - найкраща функція для торгових посередників мобільних додатків для демонстрації під час зустрічей з клієнтами. Щодня запускаються нові програмами для стимулювання продажів. Ми ще не на цьому етапі, але ми майже досягли моменту, коли компанії потрібен додаток для мобільної комерції, щоб залишатися конкурентоспроможним. До кінця 2021 року понад 72,9% загального обсягу продажів електронної комерції надходитиме з мобільних пристроїв. Програми відіграють значну роль у поточному та майбутньому успіху мобільної комерції.

Штучний інтелект (Artificial Intelligence/ AI)

Штучний інтелект та машинне навчання проникли у розробку мобільних додатків багато років тому. Але лише зараз почались створюватись платформи для зручного впровадження AI до додатків. Так минулого року Apple випустила Core ML 3. Ця остання версія механізму машинного навчання iOS була створена, щоб допомогти розробникам вбудувати технологію AI у свої програми. Приклади функцій AI, які можна впровадити в мобільний додаток, включають: розпізнавання зображень, розпізнавання обличчя, класифікація тексту та зображень, розпізнавання та класифікація настрою, розпізнавання мови прогнозне обслуговування.

Мобільні гаманці

Мобільні гаманці, такі як Apple Pay, Google Pay і Samsung Pay, мають тенденцію до зростання. У 2019 році було здійснено транзакцій з мобільних гаманців на $6,1 млрд. До 2022 року цей показник, як очікується, досягне $13,98 млрд. Мобільні гаманці повинні враховуватися при розробці додатків у 2021 році. Інтеграція гаманця повинна стати стандартною функцією для кожного додатка, який обробляє транзакції. Хоча наразі це не відповідає дійсності, однак з розвитком самих гаманців буде розширюватись й їх підтримка додатками.

Мобільні додатки для замовлення послуг.

Такі програми, як Airbnb та Uber, показали, наскільки успішними можуть бути програми в цьому просторі. Користувачі витрачають $57,6 млрд на рік, користуючись послугами для замовлення послуг. Декілька прикладів в яких сферах можна використовувати цей тип додатків : виклик лікарів, віртуальні викладачі та тренери, доставка їжі, прибирання будинку, послуги з технічного обслуговування, запис на фітнес, догляд за домашніми тваринами, перукарні та салони краси.

РОЗДІЛ 2. Вибір та створення моделі роботи допоміжного інструменту для розробки мобільних додатків

2.1 Аналіз інструментів для розробки та підтримки

Для створення та підтримки мобільних додатків використовують велику кількість вже готових рішень, що значно спрощують процеси компанії для запуску продукту.

Це можуть бути як системи для зручного написання додатку, так і аналітичні інструменти для збору даних о користувачах, їх поведінці.

Мобільні аналітичні інструменти

Для створення успішного продукту також є дуже важливим отримання зворотного зв’язку о роботі вже випущеного додатку. Аналітика мобільних додатків допомагає ASO-фахівцям знати трафік конкурента, розуміти, які ключові слова краще впровадити, проаналізувати кількість відтоку користувачів, з'ясувати причину, маркетологам - спланувати етапи просування, зрозуміти, які канали трафіку розумніше задіяти, розробникам - знати, які сильні і слабкі сторони є у додатків. Це допоможе зрозуміти що потрібно ще оптимізувати: швидкість завантаження, візуальні елементи, систему бонусів.

Трекінг мобільних додатків дає можливість: відслідковувати позиції софта в Google Play і App Store, порівняти ефективність каналів, що в кінцевому підсумку допоможе вибрати правильну стратегію і заощадити гроші на рекламу; відстежити поведінку користувача, статистику завантажень додатків, скільки часу користується інструментом, що конкретно його цікавить; визначити, наскільки сервіс популярний в своїй ніші, подивитися трафік конкурента, позиції у видачі; моніторити кількість помилок програми [16].

Фреймворки для розробки мобільних додатків

Фреймворки характеризуються як комплексне середовище розробки програмного забезпечення. Вони включають в себе багато компонентів, яких основна робота - допомогти у створенні додатків. Фреймворки поставляються з рядом наборів інструментів, програм відладки, компіляторів, з бібліотеками кодів, програмними інтерфейсами і іншими компонентами. Працюючи разом, всі вони призначені для того, щоб полегшити роботу розробникам додатків. Фреймворки є основою розробки додатків, а їх використання робить процес розробки простіше [17].

Кросплатформені інструменти розробки додатків

Кросплатформені мови програмування призначені для створення одного коду, який буде зрозумілий різним операційним системам. Іншими словами, створюється один додаток, що працює на різних пристроях. Принцип дії платформ досить простий.

Кросплатформена розробка мобільних додатків має свої переваги: швидкість розробки набагато вище, ніж у нативних додатків, так як потрібно створити всього один код, і він сам буде адаптуватися під різні пристрої; вартість розробки також знижується; більшість фахівців впораються з новими фреймворками, тому без зусиль зможуть створити якісні програми; додатки на різних пристроях будуть дотримуватися єдину стилістику, а значить, підвищується впізнаваність бренду.

Але, незважаючи на переваги, кросплатформені додатки мають і ряд недоліків: інтерфейс програми не рідний, тому є необхідність підключати дизайнерів для розробки зовнішнього вигляду; деякі функції досить складно реалізувати через недоробки фреймворка; знижена швидкість роботи програми [18].

Конструктори мобільних додатків

На відміну від фреймворків та класичних інструментів для створення мобільних додатків – основна функція конструкторів – це можливість створювати додатки не маючи глибоких знань у технічній стороні створення додатків. Однак існує ряд недоліків. За використання сервісу доведеться платити, причому кожен місяць, а сума залежить від обраного тарифного плану та замовленого набору інструментів. Може додатково стягуватися плата за інтенсивність використання продукту і кількість його установок. Користувач залишається прив'язаним до сервісу, який може збільшити вартість діючих тарифів, або зовсім заблокувати аккаунт без особливих на це причин. Зміна і можливість доопрацювання додатків можуть бути ускладнені через обмеженість пропонованих сервісом шаблонів і наборів компонентів. До недоліків ще можна також додати уповільнення швидкості роботи створеного додатку [19].

Інструменти AR (доповненої реальності)

Доповнена реальність – є одним з досить нових методів взаємодії користувача додатку з контентом, коли контент додатку проектується на реальне середовище за й відображається на екрані девайсу. Зазвичай це відбувається за допомогою екрану телефону та його камери, однак існують й більш складні пристрої, як окуляри доповненої реальності.

За допомогою камери девайс розпізнає простір навколо себе, з'ясовує габарити і опорні точки, знаходячи горизонтальні поверхні, на яких можна розташувати віртуальний об'єкт - заздалегідь зібрану 3D-модель або навіть 2D-об'єкт. Наприклад спеціально розроблену картинку або зняте відео.

Є 3 типи додатків з доповненою реальністю:

Доповнена реальність, прив'язана до маркера. Коли для вбудовування цифрового контенту в реальний світ додаток повинен точно знати, на що дивиться користувач.

Доповнена реальність, НЕ прив'язана до маркера. Безмаркерний AR дозволяє розмістити об'єкт віртуальної реальності в конкретній точці. Приклад: одне з найзнаменитіших у світі комерційних AR-додатків Ikea Place працює саме за таким принципом. Користувач може розмістити віртуальні меблі у своїй вітальні, додаток зрозуміє, що це приміщення, орієнтуючись на точки підлоги, стін і стелі, і дозволить розставити тривимірні крісла і столи так, що віртуальний світ ідеально впишеться в реальний.

Доповнена реальність, прив'язана до конкретної локації. В цьому випадку AR-додаток пов'язує контент доповненої реальності з певним місцем розташування, визначаючи його за допомогою GPS, компаса або систем комп'ютерного зору [20].

Інструменти AI (штучного інтелекту)

Використання штучного інтелекту відкриває дуже великі можливості для автоматичного аналізу даних та використання певних алгоритмів для покращення користувацького досліду. Найбільш вживаними методами використання штучного інтелекту в мобільних додатках є:

Персоналізований маркетинг. Показ клієнту персоналізованої реклами підвищує його зацікавленість, допомагає підвищити лояльність і, відповідно, покращує продажу.

Автоматизація взаємодії з клієнтами. Більшість взаємодій з клієнтами, такі як відправка електронної пошти, листування в чаті або соціальних мережах, телефонні дзвінки, в даний час вимагають участі людини. ШІ ж дозволяє компаніям автоматизувати всі ці способи комунікації.

Аналіз даних і автоматизація. Найбільша перевага використання штучного інтелекту на основі хмарних обчислень - він може швидко виявляти важливі та актуальні результати під час обробки великих даних.

Прогноз результатів. Ще однією перевагою ШІ є здатність з високою точністю прогнозувати результати на основі аналізу даних [21].

Рекламні інструменти для мобільних додатків

Реклама є одним з найбільш прибуткових методів монетизації додатків, як з безкоштовною та й з платною моделлю розповсюдження. Розповсюдженою є як реклама інший додатків (навіть тієї ж категорії, що й додаток в якому реклама запущена) так й реклама зовнішніх товарів та послуг.

Використання реклами в додатку є ряд переваг як для власника реклами, так й, відповідно, для видавника додатку, а саме:

- Більше можливостей для монетизації
- Користувачі витрачають більше часу у програмах
- Більш залучена аудиторія
- Вища вірогідність конверсії
- Більша точність
- Більше персоналізації
- Немає рекламного блокування

Однак також існують й недоліки -це висококонкурентний канал. Оскільки все більше рекламодавців прагнуть відображати свою рекламу в додатках, це стає дедалі більш конкурентною та дорогою маркетинговою платформою. Залежить від бізнес-ніші. Якщо цільовий клієнт додатку не є постійним користувачем смартфона або не любить використовувати вміст додатків, тоді оголошення для мобільних додатків можуть не найкращим чином відповідати бренду – буде низька кількість як кліків, так й конверсії [22].

Інструменти для здійснення мобільних платежів

Одним з важливих інструментів для додатків сфери онлайн комерції – є власні системи мобільних платежів, коли користувачу, для здійснення оплати, нема необхідності переходити на зовнішні ресурси для оплати. Важливо додати, що для прямої монетизації додатків (покупок цифрових товарів та послуг, що стосується саме додатку) – використовуються платіжні системи Google Play та App Store, що стягують комісію за користування. Використання власних платіжних систем у такому випадку, щоб уникнути комісії, є порушенням користувацьких правил магазинів.

Варіантів реалізації платіжних систем декілька: підключити послуги агрегатора або шлюзу і задовольнятися адаптивної версією або - створити власний додаток з продуманим механізмом оплати.

Інструменти відправки push-повідомлень

Push-повідомленя - це короткі повідомлення, які веб-ресурс розсилає своїм передплатникам на комп'ютери і мобільні пристрої. Такі повідомлення повертають користувача на сайт, який він уже відвідував (причому роблять це набагато ефективніше, ніж контекстна реклама або будь-який інший вид масових розсилок). Технічні можливості push-розсилок дозволяють розробляти більш продуктивну і «чуйну» до поведінки користувача маркетингову стратегію [23].

2.2 Розробка моделі роботи допоміжного інструменту для розробки мобільних додатків

Для випускної роботи в рамках досліджуваної теми було прийнято рішення створення допоміжного інструменту для розробки, а саме інструменту категорії QA (quality assurance / забезпечення якості) - програму, що зчитує та аналізує скільки часу користувач персонального комп’ютеру з запущеною

програмою витрачав, використовуючи програми, встановлені на комп'ютері. Інструмент допомагає у веденні детальної звітності, що до процесу розробки, що, в свою чергу, має значну вагу в оцінюванні ефективності та подальшому плануванні майбутніх задач, проектів, тощо. Програма є важливою не тільки в командній розробці, де необхідно дотримуватись рівномірного навантаження між членами команди й ефективно налаштовувати процес, а й також для самостійних проектів, що виконуються однією людиною. За відсутності команди й управління дуже важко стає керувати часом, що витрачається на виконання, як окремих задач, так й замовлень в цілому, через що важко планувати власний дохід на одиницю часу. За допомогою інструмента можна побачити скільки часу було витрачено на використання як робочих програм, так й програм особистого використання.

За мову програмування був обраний Python. Для написання програми використовувався редактор коду Microsoft Visual Code, у зв'язку зі швидкодією та зручністю редактору, широкими можливостями для налаштувань, та великим вибором допоміжних розширень для роботи. Робота виконувалась та тестувалась програма на операційній системі Windows 10, останньої, на момент написання, версії – версія 2004, збірка 19041,985.

Загальний принцип роботи програми WinTime (надалі просто WinTime) можна розділити на декілька етапів. Перший етап наведений на схемі (рис. 2.1.), де необхідним результатом є отримання набору даних відслідкованого додатку.

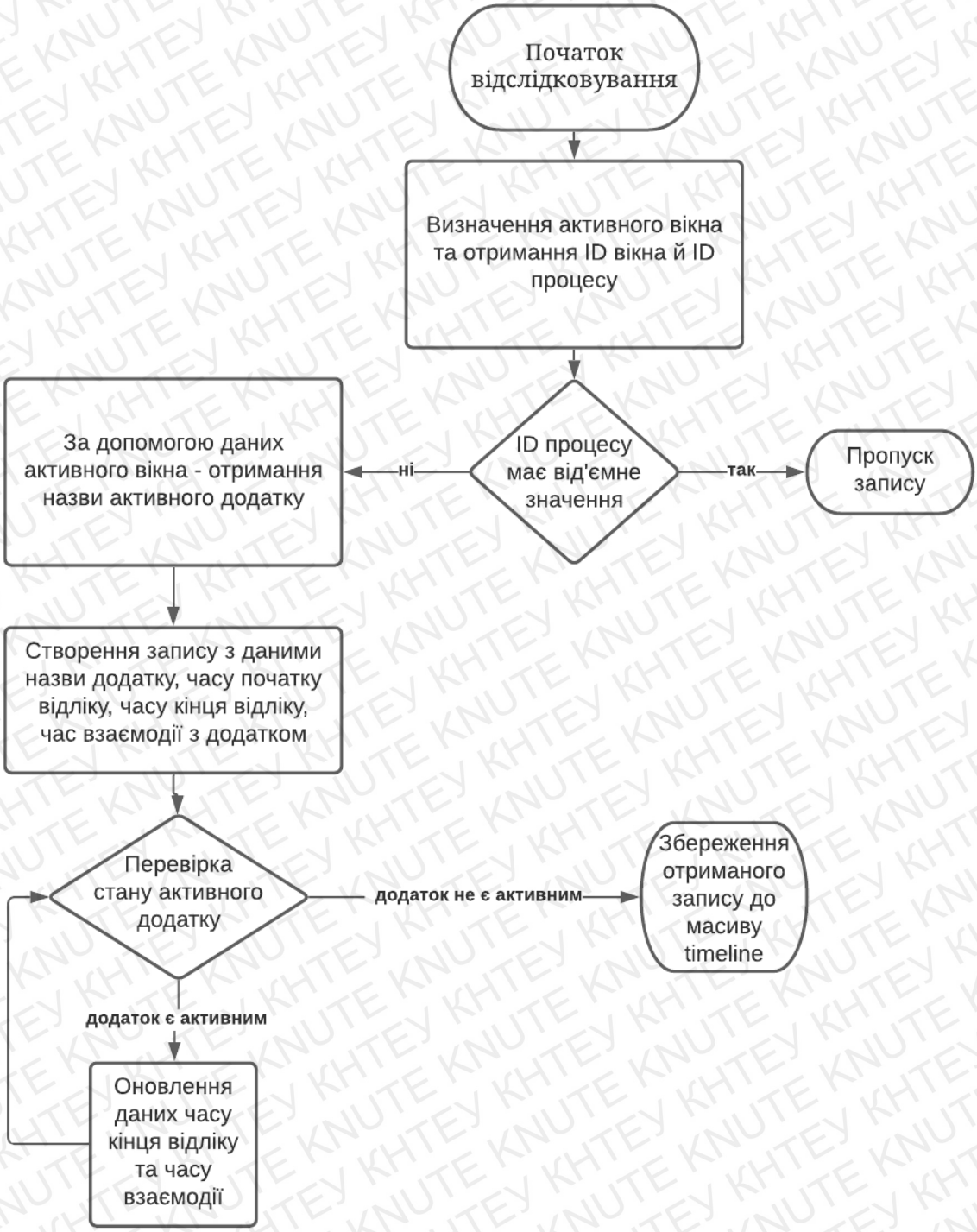

Рис. 2.1. Схема отримання запису даних активного додатку

WinTime визначає активне вікно операційної системи Windows, й, за допомогою даних активного вікна отримує назву обраної програми, час, коли ця програма стала активною (було обрана користувачем як активне вікно), коли перестала бути активною, та загальний час взаємодії. В першу чергу відбувається аналіз ID отриманого процесу. Стандартно ID процесів мають додатне значення, однак в Windows визначається також таких процес, як системні переривання, що мають від'ємне значення, або «фантомні» процеси з нульовим ID. Подібні процеси можуть бути знайдені WinTime при швидкому перемиканні вікон й, у випадку їх визначення – записи таких процесів не зберігаються, а процеси ігноруються.

Якщо ID процесу є додатним, то спочатку відбувається створення форми запису, куди додається назва додатку, час початку відліку, час кінця відліку та час роботи програми. Перші два значення є сталими й не змінюються, в той час коли час кінця відліку й роботи програми є тимчасовими й дорівнюють значенню початку відліку й нулю відповідно. Одразу після створення запису відбувається перевірка, чи додаток є все ще активним. Оновлення записів й перевірка відбуваються з затримкою в 1 секунду. Встановлення запису й перевірки без затримки може становити сильне навантаження на систему. До поки додаток є активним відбувається прорахунок часу кінця відліку та часу взаємодії. Коли додаток перестає бути активним – отриманий набір даних зберігається як окремий запис. Кожен отриманий запис – це кожна окрема ітерації відкриття та закриття вікна програм користувачем.

Наступним етапом є отримання просумованих даних всіх ітерацій відкриття програм в час роботи WinTime, наведений на схемі (Рис. 2.2.).

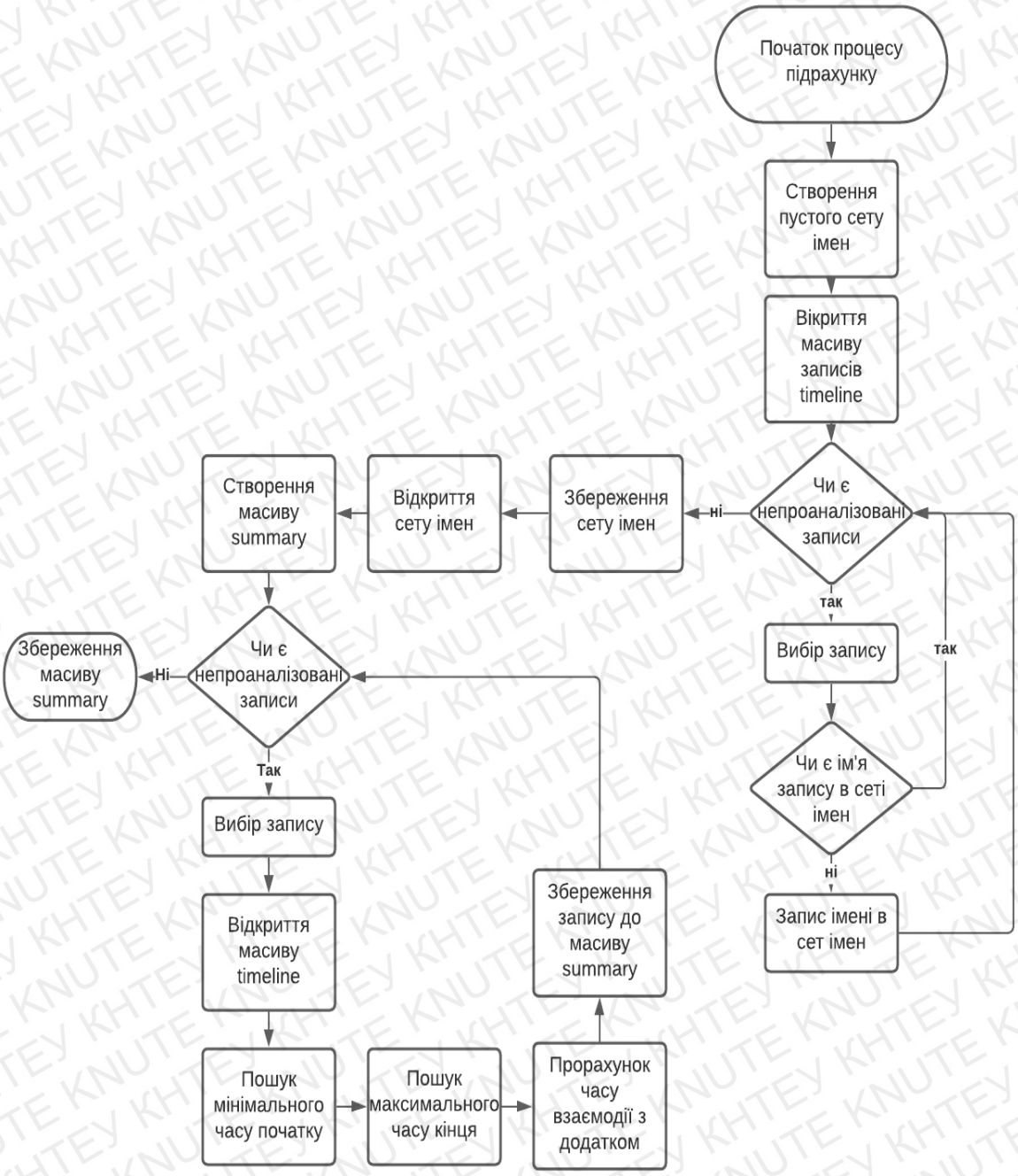

Рис. 2.2. Схема отримання просумованих записів роботи додатків

На цьому етапі першочерговим є створення сету імен для створення списку проаналізованих WinTime додатків. Є необхідним серед усіх записів в масиві timeline створити список програм без дублікатів. В мові програмування Python зручним інструментом для цього є один з різновидів масивів, а саме – сет. Його особливість в тому, що дії пов’язані з аналізом існуючих записів в сеті та інформації, що надається до запису відбувається автоматично й в результаті дублікати в сет не записуються, а пропускаються.

Після отримання списку імен для кожного імені створюється повний запис з часом першого відкриття додатку, останнього закриття та прорахованого часу роботи додатку. Отримані результати записуються в масив summary. Час відкриття та час закриття знаходиться серед усіх записів в масиві timeline, що відповідають обраному імені.

WinTime запускається за допомогою розробленого користувацього інтерфейсу де, зокрема запуску й зупиненню програми, знаходиться інтерактивний календар та область відображення даних за обраний в календарі період часу. Дані для відображення беруться зі збереженого CSV файлу, який заповнюється по закінченню роботи WinTime. Поки WinTime працює – відображуються й оновлюються в реальному часу дані саме за цей запуск.

РОЗДІЛ 3. Створення допоміжного інструменту для розробки мобільних додатків

3.1 Аналіз використаних в процесі роботи зовнішніх бібліотек Python

win32gui

win32gui націлена на роботу з інтерфейсом операційної системи. Для розробленої програми WinTime є необхідним ідентифікація вікон відкритих в операційній системі програм, які, в свою чергу є частиною інтерфейсу операційної системи. За допомогою бібліотеки отримується ID обраного вікна.

win32process

Отриманий в процесі роботи програми ID вікна не є можливим для ідентифікації програми, бо можуть бути запущені як декілька вікон з різними ID, так й сама програма на протязі проаналізованого періоду може відкриватись неодноразова з різними ID. У зв'язку з цим постало питання отримання стабільного ідентифікатору для аналізованих програми. За допомогою win32process для кожного вікна можливо отримати масив з ID процесу та ID потоку.

Psutil

Однак проблема залишається, бо ID процесу також може змінюватись в процесі роботи операційної системи. Бібліотека psutil є необхідною, бо саме за допомогою неї є можливим отримати назву запущеної програми за допомогою ID відповідного процесу. Назва є сталим параметром програми, що й використовується в розробці як ідентифікатор програми.

CSV

В процесі розробки постало питання як зберігати отримані в процесі роботи програми дані для можливого подальшого використання та для можливості аналізу даних за декілька запусків програми. Для вирішення цього питання було обрано бібліотеку CSV, що відповідає за взаємодію з файлами одноймненного розширення. Окрім збереження фінальних результатів роботи програми за всі її ітерації після зупинки у відповідні CSV файли також зберігалися тимчасові дані, що є необхідними саме для цього запуску програми, а саме: масиви timeline та summary. Це допомагає для відлагодження та тестування розробленої програми.

OS

Також необхідною для збереження результатів було необхідним використання бібліотеки OS, що дозволяє створювати нові та читати вже існуючі файли.

SYS

Бібліотека SYS пропонує методи, які дозволяють працювати з різними елементами середовища виконання Python. З його допомогою можна взаємодіяти з інтерпретатором, використовуючи різні функції. У розробці є необхідним для збірки програми.

Datetime, timedelta

Datetime та timedelta відповідають за роботу з даними, пов’язаними з часом за датами. Використовується для визначення теперішнього часу, в який відбувається оновлення даних о роботі програми та для математичного обрахунку часу роботи програми. Стандартно дані часу вивантажуються в форматі, який неможливо використовувати для математичних операцій і, відповідно, потребую конвертації в інші формати, що також виконується за допомогою вказаних бібліотек.

PyQt5

Використовується для побудування інтерактивного інтерфейсу програми. PyQt - набір розширень графічного фреймворку Qt, , який також функціонує як крос-платформна середовище розробки додатків, для мови програмування Python, виконаний у вигляді розширення Python. Qt - це популярне середовище C ++ для створення програмного забезпечення за допомогою графічного інтерфейсу для всіх основних настільних, мобільних і вбудованих платформ (підтримує Linux, Windows, MacOS, Android, iOS, Raspberry Pi і багато інших).

3.2 Програмна реалізація допоміжного інструменту для розробки мобільних додатків

Надалі – детальніше про розробку програми.

```
import csv
import os
import sys
from datetime import datetime, timedelta

import psutil
import win32gui
import win32process
from PyQt5 import QtCore
from PyQt5.QtCore import QObject, Qt, QThread, QTimer, pyqtSignal
from PyQt5.QtWidgets import (QApplication, QCalendarWidget, QGridLayout,
                             QLabel, QListWidget, QListWidgetItem, QMessageBox,
                             QPushButton, QScrollBar, QWidget)
```

Рис.3.1 Імпорт бібліотек

Спочатку імпортуються бібліотеки для подальшої роботи (Рис.3.1).

```
class Process:
    def __init__(self, window_id):
        self.window_id = window_id
        thread_process_id = win32process.GetWindowThreadProcessId(
            window_id)[-1]
        process_id = psutil.Process(thread_process_id)
        self.name = process_id.name()
        self.start = datetime.now()
        self.end = datetime.now()
```

Рис.3.2 Отримання даних процеса

За допомогою класу Process ми при визові отримаємо дані обраного процесу за допомогою ID його вікна (Рис.3.2).

В цьому блоці, використовуючи команду бібліотеки win32process GetWindowThreadProcessId(window_id)[-1] отримуємо ID процесу, а не вікна. Й вже за допомогою вікна через іншу бібліотеку отримуємо ім’я процесу. Важливим є те, що при використанні наведеної команди ми отримаємо масив з 2-ух чисел, де перше це код потоку, а другий – код процесу вікна, тому обираємо останній елемент в масиві. Конструкція є необхідною, бо код вікна й код потоку можуть змінюватись при повторному запуску програми, тому ідентифікація програм виконується за допомогою імені. В цьому ж блоці задаємо початкове значення часу, коли додаток був відкритий й закритий – початково це єдина точка в часі.

```
def __init__(self, parent=None):
        QObject.__init__(self, parent=parent)
        if not os.path.isfile("./overall_summary.csv"):
            with open("overall_summary.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        if not os.path.isfile("./summary.csv"):
            with open("summary.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        if not os.path.isfile("./timeline.csv"):
            with open("timeline.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        self.continue_run = True  # provide a bool run condition for the class
```

Рис.3.3 Створення необхідний файлів при запуску

Цей блок відповідає за створення необхідних файлів для подальшого збереження результатів роботи WinTime. Створення відбувається лише за умовою, що необхідні файли ще не знаходяться у теці, де знаходиться програма (Рис.3.3).

```
def do_work(self):
        temp_window = None
        self.is_working = True
        while self.continue_run:  # give the loop a stoppable condition
            active_window = win32gui.GetForegroundWindow()
            if temp_window != active_window:
                try:
                    temp_window = active_window

                    window_object = Process(active_window)
                    duration = window_object.end - window_object.start
                    duration = datetime.strptime(
                        str(duration), '%H:%M:%S').time()
                    self.overall_timeline.append([
                        window_object.name,
                        window_object.start,
                        window_object.end,
                        timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])

                except ValueError:
                    pass
                except psutil.NoSuchProcess:
                    pass
            else:
                self.overall_timeline[-1][2] = datetime.now()
                start = self.overall_timeline[-1][1]
                end = self.overall_timeline[-1][2]
                d = end - start
                self.overall_timeline[-1][3] = d

            self.generate_timeline()
            self.get_summary()

            QThread.sleep(1)

        self.finished.emit()  # emit the finished signal when the loop is done
```

Рис.3.4 Основна робота програми

В методі do_work за допомогою win32gui.GetForegroundWindow отримується код вікна, який, в свою чергу, передається класу Process. За допомогою класу Process ми створюємо масив, де кожен запис має значення імені, часу початку, часу кінця та часу проведеного у вікні програми. В блоці if дія виконується по одному разу для кожного нового відкриття вікна при першій ітерації циклу. В інших ітераціях виконуються операції блоку else, а саме – за допомогою вже даних у вже створеному масиві оновлюємо дані кінця та час проведений у вікні. Також при кожній ітерації виконуються методи generate_timeline та get_summary, що будуть описані далі. За допомогою команди QThread.sleep(1) встановлюється затримка потоку – вона необхідна, щоб обмежити виконання програми й, відповідно, оновленням даних одним

виконанням на секунду. Це знижує навантаження на систему, у порівнянні з роботою програми без затримки. Результати обчислень зберігаються в масив overall_timeline

Винятки у циклі потрібні, у випадках системних преривань та інших системних процесів, що можуть відбуватись при швидкій зміні між вікнами. Програма ігнорує ці процеси й не дозволяє їм впливати на роботу програми й процес збору даних (Рис.3.4).

```
def stop(self):
    with open("overall_summary.csv", "a", newline="") as file:
        writer = csv.writer(file)
        writer.writerows(summary)
        file.close()
    self.continue_run = False
    self.is_working = False  # set the run condition to false on stop
```

Рис.3.5 Зупинка програми

Метод stop відповідає не тільки за припинення роботи do_work, а й також за оновлення даних в файлі overall_summary.csv за допомогою масиву summary, що попередньо заповнюється методом get_summary (Рис.3.5).

```
def generate_timeline(self):
        try:
            with open("timeline.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(self.overall_timeline)
                file.close()
                self.names_existed.add(self.overall_timeline[-1][0])
        except IndexError:
            pass
```

Рис.3.6 Запис файлу timeline.csv

У цьому блоці у файл timeline.csv записуються дані з масиву overall_timeline й також заповнюється множина (або інакше – сет) імен

names_existed. Особливість сетів – при записі значення в сет, що в сеті вже є – воно пропускається, а не дублюється. Це допомагає в сеті names_existed створити список імен програм, що були, в цілому, відкриті за час роботи програми.

Пропуск необхідний для ситуацій, коли масив подається пустим, ще без записів. В інакшому випадку WinTime аварійно завершав би роботу (Рис.3.6).

```
def get_summary(self):
        summary.clear()
        self.raw_data.clear()
        with open("timeline.csv", 'r', newline="") as file:
            reader = csv.reader(file)
            for p in reader:
                try:
                    duration = datetime.strptime(p[3], "%H:%M:%S")
                except ValueError:
                    duration = datetime.strptime(p[3], "%H:%M:%S.%f")
                self.raw_data.append([p[0],
                                      datetime.strptime(
                                          p[1], "%Y-%m-%d %H:%M:%S.%f"),
                                      datetime.strptime(
                                          p[2], "%Y-%m-%d %H:%M:%S.%f"),
                                      timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])
        start = None
        end = None
        # raw_data = []
        for p in self.names_existed:
            sum_time = timedelta(hours=0, minutes=0, seconds=0)
            for i in self.raw_data:
                if i[0] == p:
                    if start == None:
                        start = i[1]
                    end = i[2]
                    sum_time = sum_time + i[3]
            summary.append([p, start, end, sum_time])

        with open("summary.csv", 'w', newline="") as file:
            file.flush()
            writer = csv.writer(file)
            writer.writerows(summary)
            file.close()
```

Рис.3.7 Прорахунок та запис основних даних знайдених програм

В методі get_summary ми читаємо записані дані в timeline.csv для кожного запису та записуємо ці дані, конвертуючи їх у необхідні формати, до масиву raw_data. Надалі цей масив ми використовуємо для того, щоб вирахувати загальні дані по використаних програмах, бо раніше ми знайшли та записали в файл timeline.csv дані щодо кожного окремого відкриття вікна програм.

Для цього ми проходимось по сету імен names_existed, й опрацьовуємо дані з масиву raw_data для кожного імені програми. Отримані дані записуємо у файл summary.csv (Рис.3.7).

```
def get_overall_summary(self):
    self.selected_summary.clear()
    self.all_names.clear()
    self.overall_summary.clear()
    with open("overall_summary.csv", 'r', newline="") as file:
        reader = csv.reader(file)
        for p in reader:
            self.all_names.add(p[0])
            try:
                duration = datetime.strptime(p[3], "%H:%M:%S")
            except ValueError:
                duration = datetime.strptime(p[3], "%H:%M:%S.%f")
            self.overall_summary.append([p[0], datetime.strptime(
                p[1], "%Y-%m-%d %H:%M:%S.%f"),
                datetime.strptime(
                p[2], "%Y-%m-%d %H:%M:%S.%f"),
                timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])
    start = None
    end = None

    for p in self.all_names:
        sum_time = timedelta(hours=0, minutes=0, seconds=0)
        for i in self.overall_summary:
            if i[0] == p:
                if start == None:
                    start = i[1]
                end = i[2]
                sum_time = sum_time + i[3]
        self.selected_summary.append([p, start, end, sum_time])
```

Рис.3.8 Запис масиву на основі даних всіх запусків програми

Оброблені дані по завершенню роботи програми зберігаються до файлу overall_summary.csv. У вказаному блоці очищуються масиви, що були записані раніше й дані з вже записаного файлу overall_summary.csv читаються та додаються до масиву selected_summary. Це необхідно для відображення даних за минулі запуски програми (Рис.3.8).

```
def initUI(self):
    self.is_working = False
    self.timer = QTimer()
    self.timer.setInterval(1000)
    self.timer.timeout.connect(self.show_summary)

    self.timer.start()
    # Buttons:
    self.btn_start = QPushButton('Start')
    self.btn_stop = QPushButton('Stop')

    # GUI title, size, etc...
    # self.setGeometry(300, 300, 300, 220)
    self.setMinimumSize(700, 300)
    self.setWindowTitle('WinTime')

    self.layout = QGridLayout()
    self.cal = QCalendarWidget()
    self.cal.clicked[QtCore.QDate].connect(self.select_timeframe)
    self.info = QListWidget()
    self.info.setStyleSheet("QListWidget"
                            "{"
                            "border : 2px solid black;"
                            "background : lightgrey;"
                            "}"
                            "QListWidget QScrollBar"
                            "{"
                            "background : lightblue;"
                            "}"
                            "QListView::item"
                            "{"
                            "border : 2px solid black;"
                            "background : white;"
                            "}"
                            )
    self.info.setSpacing(5)

    self.info_scroll = QScrollBar()
    self.info.setVerticalScrollBar(self.info_scroll)
    self.info.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

    self.start_date = QLabel("Start Date: ")
    self.end_date = QLabel("End Date: ")
    self.info_popup = QMessageBox()
    self.info_popup.setFixedSize(200, 200)
    self.info_popup.setWindowTitle("Info Section")
    self.info_popup.setText("1. Press 'Start' Button in order to run the app.\n
1.1. Select Date in the calendar in order to see summary for exact date timeframe\n
2. Press 'Stop' button in order to stop monitoring and save data into the file.\n
3. You can get summary csv file in the app directory.")
    self.info_popup_btn = QPushButton("Info")
    self.info_popup_btn.clicked.connect(self.show_info)

    self.layout.addWidget(self.info_popup_btn, 1, 4)
    self.layout.addWidget(self.cal, 0, 0, 1, 2)
    self.layout.addWidget(self.info, 0, 2, 1, 3)

    self.layout.addWidget(self.btn_start, 1, 0)
    self.layout.addWidget(self.start_date, 1, 1)

    self.layout.addWidget(self.end_date, 1, 2)
    self.layout.addWidget(self.btn_stop, 1, 3)

    self.start_date.setText(self.cal.selectedDate().toString())
    self.end_date.setText(self.cal.selectedDate().toString())

    self.setLayout(self.layout)
```

Рис.3.9 Інтерфейс програми

Тут задається інтерфейс програми. Як вже було вказано раніше – для нього використовується бібліотека PyQt5. Задаються такі дані, як мінімальний розмір вікна, назва вікна, стилі, додається інтерактивний календар. Елементи

позиціонуються за допомогою типу відображення сіткою – задається за допомогою строки self.layout = QGridLayout(). Також при налаштуванні інтерфейсу додається допоміжне вікно з інформацію для користування програмою, що відкривається за допомогою кнопки Info. З тої причини, що календар є інтерактивним й ми за його допомогою обираємо дати для відображення даних - додаємо до календаря поля з текстом початкової обраної дати та кінцевої. За замовчуванням задаємо дату запуску програми на обидві позиції (Рис.3.9).

```
# Thread:
self.thread = QThread()
self.worker = Worker()
# connect stop signal to worker stop method
self.stop_signal.connect(self.worker.stop)
self.worker.moveToThread(self.thread)

# connect the workers finished signal to stop thread
self.worker.finished.connect(self.thread.quit)
# connect the workers finished signal to clean up worker
self.worker.finished.connect(self.worker.deleteLater)
# connect threads finished signal to clean up thread
self.thread.finished.connect(self.thread.deleteLater)

self.thread.started.connect(self.worker.do_work)
self.thread.finished.connect(self.worker.stop)

# Start Button action:

self.btn_start.clicked.connect(self.thread.start)
self.btn_stop.clicked.connect(self.stop_thread)

self.first_date = self.cal.selectedDate()
self.last_date = self.cal.selectedDate()

self.show()
```

Рис.3.10 Налаштування потоків

Стандартно програми використовують лише один потік, тобто виконують одночасно лише одну операцію. В нашому випадку – всі операції, що стосуються основної роботи програми виконуються по черзі й не потребують декількох потоків. Однак за умовами програма буде виконуватись поки не отримає сигнал для зупинки, а для опрацювання цього сигналу потрібно виконати другу операцію – операцію зупинки програми паралельно з її роботою.

Для цього у вказаному блоці відокремлюємо основний процес програми й процес зупинки програми на різні потоки та налаштовуємо їх на кнопки, що знаходяться в інтерфейсі програми (Рис.3.10).

```
def show_summary(self):
    gui.info.clear()

    if self.is_choosing:
        if self.timeframe_selected:
            if self.is_blinking:
                self.start_date.setText("")
                self.is_blinking = False
            else:
                self.start_date.setText(
                    "Start: " + self.first_date.toString())
                self.is_blinking = True
        else:
            if self.is_blinking:
                self.end_date.setText("")
                self.is_blinking = False
            else:
                self.end_date.setText("End: " + self.last_date.toString())
                self.is_blinking = True

    if self.worker.is_working:
        for i in summary:
            if (i[1] >= self.first_date and i[2] <= self.last_date) or (i[1] == self.first_date and i[2] == self.last_date):
                gui.info.addItem(
                    QListWidgetItem(
                        "Name: " + i[0] +
                        "\nFirst Start:\t\t" + datetime.strftime(i[1], "%H:%M:%S") +
                        "\nLast Exit:\t\t" + datetime.strftime(i[2], "%H:%M:%S") +
                        "\nOverall duration:\t" + str(i[3])))
    else:
        print(self.worker.selected_summary)
        for i in self.worker.selected_summary:
            if (i[1] >= self.first_date and i[2] <= self.last_date) or (i[1] == self.first_date and i[2] == self.last_date):
                gui.info.addItem(
                    QListWidgetItem(
                        "Name: " + i[0] +
                        "\nFirst Start:\t\t" + datetime.strftime(i[1], "%H:%M:%S") +
                        "\nLast Exit:\t\t" + datetime.strftime(i[2], "%H:%M:%S") +
                        "\nOverall duration:\t" + str(i[3])))
```

Рис.3.11 Оновлення виводу даних в програмі

За допомогою цього методу налаштовуємо те, як користувач буде отримувати дані роботи програми. Поки програма виконує операцію збору й обробки – в реальному часі відображаються й оновлюються дані саме за цей запуск WinTime. Коли процес вже був зупинений, або ще не був запущений – йде відображення усіх даних, що були зібрані за обраний в календарі проміжок часу.

Щоб обрати конкретний проміжок було більш зрозуміло – коли обирається початкова дата – блимає текст поля Start, коли обирається кінцева дата – поля End. Після вибору обох дат відображаються в полі звітності роботи

програми її результати. Для вибору нових дат потрібно натиснути один раз по будь-якій даті календаря, після цього увімкнеться режим повторного вибору дати (Рис.3.11).

```
def select_timeframe(self):
        if self.is_choosing:
            if self.timeframe_selected:
                # if self.cal.selectedDate() <= self.last_date:
                self.first_date = self.cal.selectedDate()
                self.start_date.setText(
                    "Start: " + self.first_date.toString())
                self.timeframe_selected = False
                if self.last_date < self.first_date:
                    self.last_date = self.first_date
                    self.end_date.setText("End: " + self.last_date.toString())
                    self.is_choosing = False
            else:
                if self.cal.selectedDate() >= self.first_date:
                    self.last_date = self.cal.selectedDate()
                    self.end_date.setText("End: " + self.last_date.toString())
                    self.timeframe_selected = True
                    self.is_choosing = False
        else:
            self.is_choosing = True

        self.worker.get_overall_summary()
```

Рис.3.12 Вибір

На вказаному скріншоті реалізований сам вибір дат, а також відбувається оновлення масиву для відображення звітності роботи (Рис.3.12).

```
def stop_thread(self):
    self.stop_signal.emit()  # emit the finished signal on stop
```

Рис.3.13 Зупинка потоку

Команда self.stop_signal.emit() у методі stop_thread відправляє сигнал для зупинення потоку (Рис.3.13).

```
def show_info(self):
    self.info_popup.show()
```

Рис.3.14 Вікно Info

Команда self.stop_signal.emit() дає сигнал для відображення вікна Info з інструкцією (Рис.3.14).

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    gui = Gui()
    sys.exit(app.exec_())
```

Рис.3.15 збірка додатку

На цьому скріншоті відображена зборка програми з її моделей (Рис.3.15).

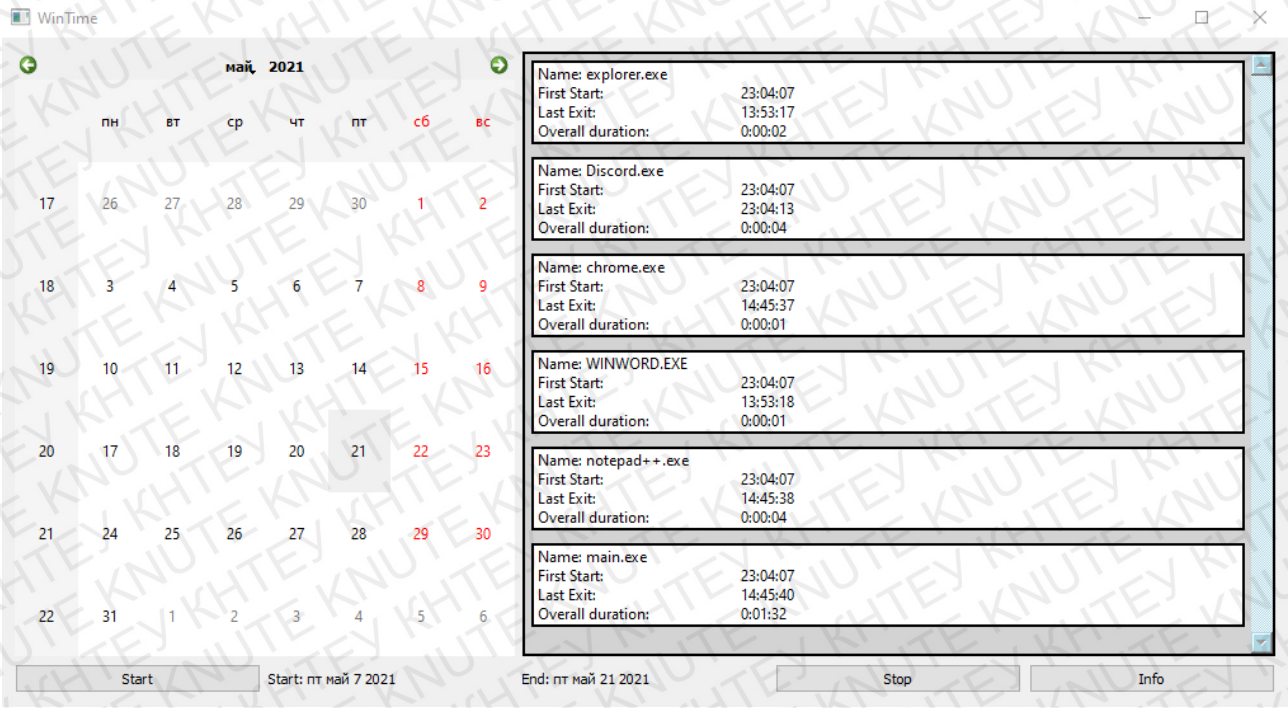

Рис.3.16 Програма WinTime в завершеному вигляді.

Так виглядає програма в її завершеному варіанті. Кнопки Start та Stop надають сигнали для запуску й, відповідно, зупинки програми. Кнопка Info відкриває окреме вікно з короткою інструкцією по роботі з програмою. Інтерактивний календар зліва підтягує календар з встановленої операційної системи Windows з його підписами та датою. В полі справа відображені записи за обраний в календарі час. Також непросумовані дані кожного окремого запуску програми, для випадків необхідних детальних розрахунків через Excel, зберігається у overall_summary.csv (Рис.3.16).

ВИСНОВКИ

1. Ринок мобільних додатків – дуже динамічний ринок, що знаходиться в стані активного розвитку та зросту. Йому характерні такі особливості – широкий спектр категорій інструментів та послуг, що, надаються для користувачів, глибоке залучення інших сфер, таких як фінансова сфера послуг, електронна комерція.

2. Проаналізовані існуючі інструментами для розробки мобільних додатків за категоріями наданих послуг - це є як додатки, в яких відбувається процес розробки програми так й комплекти готових рішень різних категорій (таких як штучний інтелект, системи платежів) та допоміжні інструменти (аналітичні та маркетингові інструменти). Доцільним є використання одразу декількох інструментів у відповідності до потреб розробки.

3. Розроблено та обґрунтовано модель роботи допоміжного інструменту для розробки мобільних додатків - програму для збору інформації про витрачений час користування тим чи іншим додатком.

4. Розроблено програму для збору інформації про витрачений час користування тим чи іншим додатком. Ці дані є важливими для налаштування ефективного процесу розробки додатку, командної взаємодії, фінансової ефективності процесу, особливо в умовах сфери розробки мобільних додатках через відносно малий термін створення додатків, великий рівень конкуренції та швидкозмінність трендів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IOS — что это такое: подробно и простыми словами [Електронний ресурс]. - Режим доступу: https://fans-android.com/ios-chto-eto-takoe/#IOS__8212

2. Костелло С. The History of iOS, from Version 1.0 to 14.0 [Електронний ресурс]. - Режим доступу: https://www.lifewire.com/ios-versions-4147730

3. App store [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/App_store

4. Android (operating system) [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Android_(operating_system)

5. Сутягин А. История бренда: Android [Електронний ресурс]. - Режим доступу: https://androidinsider.ru/eto-interesno/istoriya-brenda-android.html

6. Маджид Куреші А. Top 10 smartphone brands in the world [Електронний ресурс]. - https://www.techsaa.com/top-10-smartphone-brands-in-the-world-2020/

7. Google Play [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Google_Play

8. Aptoide [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Aptoide

9. F-Droid / [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/F-Droid

10. Huawei AppGallery [Електронний ресурс]. - Режим доступу: https://en.wikipedia.org/wiki/Huawei_AppGallery

11. Android vs. iOS - Smartphone OS sales market share evolution [Електронний ресурс]. - https://www.kantarworldpanel.com/global/smartphone-os-market-share/

12. A Billion iPhone Users [Електронний ресурс]. - Режим доступу: https://www.aboveavalon.com/notes/2020/10/26/a-billion-iphone-users

13. The State of Mobile 2021 [Електронний ресурс]. - Режим доступу: https://www.appannie.com/en/go/state-of-mobile-2021

14. Десять трендов маркетинга мобильных приложений 2021 года по версии экспертов [Електронний ресурс]. - Режим доступу: https://vc.ru/marketing/195488-desyat-trendov-marketinga-mobilnyh-prilozheniy-2021-goda-po-versii-ekspertov

15. Хінді Д. 15 Mobile App Development Trends of 2021 [Електронний ресурс]. - Режим доступу: https://buildfire.com/mobile-app-development-trends/

16. Аналитика мобильных приложений. Выбираем рабочие инструменты [Електронний ресурс]. - Режим доступу: https://blog.advertmobile.net/analitika-mobilnyx-prilozhenij-vybiraem-rabochie-instrumenty/

17. Гуменюк Е. 10 лучших фреймворков для разработки мобильных приложений [Електронний ресурс]. - Режим доступу: https://freelance.today/poleznoe/10-luchshih-freymvorkov-dlya-razrabotki-mobilnyh-prilozheniy.html

18. Разработка кроссплатформенных мобильных приложений [Електронний ресурс]. - Режим доступу: https://wellsoft.pro/blog/shkola-zakazchika-krossplatformennye-prilozheniya

19. КОНСТРУКТОР МОБИЛЬНЫХ ПРИЛОЖЕНИЙ [Електронний ресурс]. - Режим доступу: https://kitapp.pro/konstruktor-mobilnyh-prilozhenij/

20. Что нужно для разработки вашего первого приложения с дополненной реальностью [Електронний ресурс]. - Режим доступу: https://axmor.ru/articles/chto-nuzhno-dlia-razrabotki-vashiegho-piervogho-prilozhieniia-s-dopolniennoi-riealnostiu/

21. Приложения для бизнеса с искусственным интеллектом [Електронний ресурс]. - Режим доступу: https://polygant.net/ru/ai/prilozheniya-dlya-biznesa-s-iskusstvennym-intellektom/#2

22. In-App Advertising for Publishers [Електронний ресурс]. - Режим доступу: https://www.publift.com/blog/in-app-advertising-for-publishers

23. Что такое push-уведомления: как они работают и как выглядят [Електронний ресурс]. - Режим доступу: https://gravitec.net/ru/blog/chto-takoe-push-uvedomleniya-kak-oni-rabotayut-i-kak-vy-glyadyat/

24. Руководство по языку программирования Python [Електронний ресурс]. - Режим доступу: https://metanit.com/python/tutorial/

25. Python for Windows (pywin32) Extensions [Електронний ресурс]. - Режим доступу: https://github.com/mhammond/pywin32

26. PyWin32 Documentation [Електронний ресурс]. - Режим доступу: http://timgolden.me.uk/pywin32-docs/contents.html

27. psutil documentation [Електронний ресурс]. - Режим доступу: https://psutil.readthedocs.io/en/latest/

Додаток 1

Споживчі витрати, завантаження та витрачені години в додатках по країнам на 2020 рік

	Consumer Spend (B USD) IOS & Android		Downloads (B) IOS and Android		Hours Spent (B) Android	
Країна	Позиція	Дані	Позиція	Дані	Позиція	Дані
Китай	1	48,5300	1	96,2200	1	1153,5100
Штати	2	32,6300	3	13,3900	4	176,3800
Японія	3	20,2200	10	2,6000	15	42,6300
Південна Корея	4	5,6400	17	2,0500	13	45,4700
Великобританія	5	3,3500	12	2,4100	24	25,7300
Германія	6	3,1200	14	2,2000	19	33,5900
Канада	7	2,2200	26	1,1400	29	13,8600
Франція	8	2,0600	15	2,1500	17	35,1600
Австралія	9	1,9100	31	0,8300	31	9,0400
Росія	10	1,3300	6	5,5800	8	94,6100
Італія	12	0,9600	20	1,7400	20	31,6200
Іспанія	15	0,7000	23	1,4600	22	26,5500
Турція	18	0,5500	8	3,4200	14	44,2900
Мексика	19	0,5200	7	4,5000	7	98,8300
Індія	22	0,5000	2	24,2700	2	650,6600
Польща	26	0,3900	29	0,8700	27	18,7200
Бельгія	27	0,3700	37	0,3700	35	4,6100
Україна	34	0,2200	24	1,2300	21	29,3200
Румінія	39	0,1700	36	0,5300	30	11,7600
Греція	42	0,1500	43	0,2500	34	4,6800
Венгрія	43	0,1400	44	0,2500	36	4,3400
Єгипет	44	0,1200	13	2,3000	12	58,4100
Болгарія	52	0,0700	50	0,1700	41	2,7500

Додаток 2

Рейтинг додатків соціальних мереж по країнам за 2020 рік

Позиція	1	2	3	4	5
Аргентина	TikTok	Instagram	Telegram	Discord	Facebook
Бразилія	TikTok	Telegram	Instagram	Pinterest	Twitter
Канада	TikTok	WhatsApp	Twitter	Discord	Reddit
Мексика	TikTok	Instagram	Pinterest	Telegram	Twitter
США	TikTok	Discord	Facebook	Instagram	WhatsApp
Австралія	TikTok	WhatsApp	Houseparty	Twitter	Discord
Китай	TikTok	WeiSho	WeChat	Red - Shop the World	Zhihu
Індія	Telegram	WhatsApp	Instagram	MX TakaTak	Moj
Індонезія	Instagram	Telegram	TikTok	Facebook	WhatsApp
Японія	Instagram	Twitter	Line	Pinterest	Plus Message
Південна Корея	Danggeun Market	Instagram	Discord	Pinterest	TikTok
Франція	WhatsApp	TikTok	Instagram	Pinterest	Discord
Німеччина	TikTok	Telegram	Instagram	Pinterest	Discord
Росія	Telegram	Instagram	TikTok	WhatsApp	Facebook
Турція	EBA	Telegram	TikTok	Twitter	Pinterest
Великобританія	TikTok	Houseparty	WhatsApp	Discord	Pinterest

Додаток 3

Рейтинг додатків та окремо мобільних ігор за 2020 рік за кількістю користувачів

	За завантаженнями	
Позиція	Додатки	Ігри
1	TikTok	Free Fire
2	Facebook	Among Us!
3	WhatsApp Messenger	Subway Surfers
4	ZOOM Cloud Meetings	PUBG MOBILE
5	Instagram	Gardenscapes - New Acres
6	Facebook Messenger	Hunter Assassin
7	Google Meet	Brain Out
8	Snapchat	My Talking Tom Friends
9	Telegram	Tiles Hop: EDM Rush
10	Netflix	Ludo King

Додаток 4

Рейтинг додатків, мобільних ігор та їх видавників за 2020 рік за завантаженнями

	За завантаженнями					
Позиція	**Додатки**	**Ігри**	**Видавники додатків**	**Країна Видавника**	**Видавники ігор**	**Країна Видавника**
1	TikTok	Free Fire	Google	США	Voodoo	Франція
2	Facebook	Among Us!	Facebook	США	AppLovin	США
3	WhatsApp Messenger	Subway Surfers	ByteDance	Китай	Crazy Labs	Ізраїль
4	ZOOM Cloud Meetings	PUBG MOBILE	Microsoft	США	Jinke Culture - Outfit7	Китай
5	Instagram	Gardenscapes - New Acres	InShot Inc	Китай	SayGames	Беларусь
6	Facebook Messenger	Hunter Assassin	Alibaba Group	Китай	Playgendary	Німеччина
7	Google Meet	Brain Out	Amazon	США	Azur Interactive Games	Росія
8	Snapchat	My Talking Tom Friends	Tencent	Китай	Miniclip	Швейцарія
9	Telegram	Tiles Hop: EDM Rush	ABISHKKING	Гонконг	BabyBus	Китай
10	Netflix	Ludo King	Zoom Video Commun	США	Playrix	Ірландія

Додаток 5

Рейтинг додатків, мобільних ігор та їх видавників за 2020 рік за користувацькими витратами

	За користувацькими витратами					
Позиція	**Додатки**	**Ігри**	**Видавники додатків**	**Країна Видавника**	**Видавники ігор**	**Країна Видавника**
1	Tinder	Honour of Kings	Google	США	Tencent	Китай
2	TikTok	Pokémon GO	Tencent	Китай	Playrix	Ірландія
3	YouTube	ROBLOX	Disney	США	NetEase	Китай
4	Disney+	Monster Strike	ByteDance	Китай	Activision Blizzard	США
5	Tencent Video	Coin Master	Match Group	США	Zynga	США
6	Netflix	Game For Peace	InterActiveCorp (IAC)	США	BANDAI NAMCO	Японія
7	Google One	PUBG MOBILE	LINE	Японія	Supercell	Фінляндія
8	iQIYI	Fate/Grand Order	Baidu	Китай	Netmarble	Південна Корея
9	BIGO LIVE	Candy Crush Saga	Amazon	США	Playtika	Ізраїль
10	Pandora Music	Gardenscapes - New Acres	JOYY Inc.	Китай	Lilith	Китай

Додаток 6

Код розробленої програми

```
import csv
import os
import sys
from datetime import datetime, timedelta

import psutil
import win32gui
import win32process

from PyQt5 import QtCore
from PyQt5.QtCore import QObject, Qt, QThread, QTimer, pyqtSignal
from PyQt5.QtWidgets import (QApplication, QCalendarWidget, QGridLayout,
                             QLabel, QListWidget, QListWidgetItem, QMessageBox,
                             QPushButton, QScrollBar, QWidget)

summary = []


class Process:
    def __init__(self, window_id):
        self.window_id = window_id
        thread_process_id = win32process.GetWindowThreadProcessId(
            window_id)[-1]
        process_id = psutil.Process(thread_process_id)
        self.name = process_id.name()
        self.start = datetime.now()
        self.end = datetime.now()
```

```
class Worker(QObject):
    is_working = False
    overall_summary = []
    all_names = set()
    selected_summary = []
    finished = pyqtSignal()  # give worker class a finished signal
    overall_timeline = []
    raw_data = []
    names_existed = set()

    def __init__(self, parent=None):
        QObject.__init__(self, parent=parent)
        if not os.path.isfile("./overall_summary.csv"):
            with open("overall_summary.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        if not os.path.isfile("./summary.csv"):
            with open("summary.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        if not os.path.isfile("./timeline.csv"):
            with open("timeline.csv", "w", newline="") as file:
                writer = csv.writer(file)
                writer.writerows(summary)
                file.close()
        self.continue_run = True  # provide a bool run condition for the class

    def do_work(self):
        temp_window = None
        self.is_working = True
```

```
while self.continue_run:  # give the loop a stoppable condition
    active_window = win32gui.GetForegroundWindow()
    if temp_window != active_window:
        try:
            temp_window = active_window

            window_object = Process(active_window)
            duration = window_object.end - window_object.start
            duration = datetime.strptime(
                str(duration), '%H:%M:%S').time()
            self.overall_timeline.append([
                window_object.name,
                window_object.start,
                window_object.end,
                timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])

        except ValueError:
            pass
        except psutil.NoSuchProcess:
            pass
    else:
        self.overall_timeline[-1][2] = datetime.now()
        start = self.overall_timeline[-1][1]
        end = self.overall_timeline[-1][2]
        d = end - start
        self.overall_timeline[-1][3] = d

    self.generate_timeline()
    self.get_summary()

    QThread.sleep(1)

self.finished.emit()  # emit the finished signal when the loop is done
```

```
def stop(self):
    with open("overall_summary.csv", "a", newline="") as file:
        writer = csv.writer(file)
        writer.writerows(summary)
        file.close()
    self.continue_run = False
    self.is_working = False  # set the run condition to false on stop

def generate_timeline(self):
    try:
        with open("timeline.csv", "w", newline="") as file:
            writer = csv.writer(file)
            writer.writerows(self.overall_timeline)
            file.close()
            self.names_existed.add(self.overall_timeline[-1][0])
    except IndexError:
        pass

def get_summary(self):
    summary.clear()
    self.raw_data.clear()
    with open("timeline.csv", 'r', newline="") as file:
        reader = csv.reader(file)
        for p in reader:
            try:
                duration = datetime.strptime(p[3], "%H:%M:%S")
            except ValueError:
                duration = datetime.strptime(p[3], "%H:%M:%S.%f")
            self.raw_data.append([p[0],
                                  datetime.strptime(
                                      p[1], "%Y-%m-%d %H:%M:%S.%f"),
                                  datetime.strptime(
```

```
                    p[2], "%Y-%m-%d %H:%M:%S.%f"),
                timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])
    start = None
    end = None
    # raw_data = []
    for p in self.names_existed:
        sum_time = timedelta(hours=0, minutes=0, seconds=0)
        for i in self.raw_data:
            if i[0] == p:
                if start == None:
                    start = i[1]
                end = i[2]
                sum_time = sum_time + i[3]
        summary.append([p, start, end, sum_time])

    with open("summary.csv", 'w', newline="") as file:
        file.flush()
        writer = csv.writer(file)
        writer.writerows(summary)
        file.close()

def get_overall_summary(self):
    self.selected_summary.clear()
    self.all_names.clear()
    self.overall_summary.clear()
    with open("overall_summary.csv", 'r', newline="") as file:
        reader = csv.reader(file)
        for p in reader:
            self.all_names.add(p[0])
            try:
                duration = datetime.strptime(p[3], "%H:%M:%S")
            except ValueError:
                duration = datetime.strptime(p[3], "%H:%M:%S.%f")
```

```
            self.overall_summary.append([p[0], datetime.strptime(
                p[1], "%Y-%m-%d %H:%M:%S.%f"),
                datetime.strptime(
                p[2], "%Y-%m-%d %H:%M:%S.%f"),
                timedelta(hours=duration.hour, minutes=duration.minute, seconds=duration.second)])
        start = None
        end = None

        for p in self.all_names:
            sum_time = timedelta(hours=0, minutes=0, seconds=0)
            for i in self.overall_summary:
                if i[0] == p:
                    if start == None:
                        start = i[1]
                    end = i[2]
                    sum_time = sum_time + i[3]
            self.selected_summary.append([p, start, end, sum_time])


class Gui(QWidget):
    is_blinking = False
    is_choosing = True
    # make a stop signal to communicate with the worker in another thread
    stop_signal = pyqtSignal()
    timeframe_selected = True

    def __init__(self):
        super().__init__()
        self.initUI()
        # self.select_timeframe()

    def initUI(self):
        self.is_working = False
```

```
self.timer = QTimer()
self.timer.setInterval(1000)
self.timer.timeout.connect(self.show_summary)

self.timer.start()
# Buttons:
self.btn_start = QPushButton('Start')
self.btn_stop = QPushButton('Stop')

# GUI title, size, etc...
# self.setGeometry(300, 300, 300, 220)
self.setMinimumSize(700, 300)
self.setWindowTitle('WinTime')

self.layout = QGridLayout()
self.cal = QCalendarWidget()
self.cal.clicked[QtCore.QDate].connect(self.select_timeframe)
self.info = QListWidget()
self.info.setStyleSheet("QListWidget"
                        "{"
                        "border : 2px solid black;"
                        "background : lightgrey;"
                        "}"
                        "QListWidget QScrollBar"
                        "{"
                        "background : lightblue;"
                        "}"
                        "QListView::item"
                        "{"
                        "border : 2px solid black;"
                        "background : white;"
                        "}"
                        )
```

```
        self.info.setSpacing(5)

        self.info_scroll = QScrollBar()
        self.info.setVerticalScrollBar(self.info_scroll)
        self.info.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

        self.start_date = QLabel("Start Date: ")
        self.end_date = QLabel("End Date: ")
        self.info_popup = QMessageBox()
        self.info_popup.setFixedSize(200, 200)
        self.info_popup.setWindowTitle("Info Section")
        self.info_popup.setText("1. Press 'Start' Button in order to run the app.\n1.1. Select Date in the calendar in order to see summary for exact date timeframe\n2. Press 'Stop' button in order to stop monitoring and save data into the file.\n3. You can get summary csv file in the app directory.")
        self.info_popup_btn = QPushButton("Info")
        self.info_popup_btn.clicked.connect(self.show_info)

        self.layout.addWidget(self.info_popup_btn, 1, 4)
        self.layout.addWidget(self.cal, 0, 0, 1, 2)
        self.layout.addWidget(self.info, 0, 2, 1, 3)

        self.layout.addWidget(self.btn_start, 1, 0)
        self.layout.addWidget(self.start_date, 1, 1)

        self.layout.addWidget(self.end_date, 1, 2)
        self.layout.addWidget(self.btn_stop, 1, 3)

        self.start_date.setText(self.cal.selectedDate().toString())
        self.end_date.setText(self.cal.selectedDate().toString())

        self.setLayout(self.layout)

        # Thread:
        self.thread = QThread()
```

```
        self.worker = Worker()
        # connect stop signal to worker stop method
        self.stop_signal.connect(self.worker.stop)
        self.worker.moveToThread(self.thread)

        # connect the workers finished signal to stop thread
        self.worker.finished.connect(self.thread.quit)
        # connect the workers finished signal to clean up worker
        self.worker.finished.connect(self.worker.deleteLater)
        # connect threads finished signal to clean up thread
        self.thread.finished.connect(self.thread.deleteLater)

        self.thread.started.connect(self.worker.do_work)
        self.thread.finished.connect(self.worker.stop)

        # Start Button action:

        self.btn_start.clicked.connect(self.thread.start)
        self.btn_stop.clicked.connect(self.stop_thread)

        self.first_date = self.cal.selectedDate()
        self.last_date = self.cal.selectedDate()

        self.show()

    def show_summary(self):
        gui.info.clear()

        if self.is_choosing:
            if self.timeframe_selected:
                if self.is_blinking:
                    self.start_date.setText("")
                    self.is_blinking = False
```

```
            else:
                self.start_date.setText(
                    "Start: " + self.first_date.toString())
                self.is_blinking = True
            else:
                if self.is_blinking:
                    self.end_date.setText("")
                    self.is_blinking = False
                else:
                    self.end_date.setText("End: " + self.last_date.toString())
                    self.is_blinking = True

        if self.worker.is_working:
            for i in summary:
                if (i[1] >= self.first_date and i[2] <= self.last_date) or (i[1] == self.first_date and i[2] == self.last_date):
                    gui.info.addItem(
                        QListWidgetItem(
                            "Name: " + i[0] +
                            "\nFirst Start:\t\t" + datetime.strftime(i[1], "%H:%M:%S") +
                            "\nLast Exit:\t\t" + datetime.strftime(i[2], "%H:%M:%S") +
                            "\nOverall duration:\t" + str(i[3])))
        else:
            for i in self.worker.selected_summary:
                if (i[1] >= self.first_date and i[2] <= self.last_date) or (i[1] == self.first_date and i[2] == self.last_date):
                    gui.info.addItem(
                        QListWidgetItem(
                            "Name: " + i[0] +
                            "\nFirst Start:\t\t" + datetime.strftime(i[1], "%H:%M:%S") +
                            "\nLast Exit:\t\t" + datetime.strftime(i[2], "%H:%M:%S") +
                            "\nOverall duration:\t" + str(i[3])))

    def select_timeframe(self):
        if self.is_choosing:
```

```
            if self.timeframe_selected:
                self.first_date = self.cal.selectedDate()
                self.start_date.setText(
                    "Start: " + self.first_date.toString())
                self.timeframe_selected = False
                if self.last_date < self.first_date:
                    self.last_date = self.first_date
                    self.end_date.setText("End: " + self.last_date.toString())
                    self.is_choosing = False
            else:
                if self.cal.selectedDate() >= self.first_date:
                    self.last_date = self.cal.selectedDate()
                    self.end_date.setText("End: " + self.last_date.toString())
                    self.timeframe_selected = True
                    self.is_choosing = False
        else:
            self.is_choosing = True

        self.worker.get_overall_summary()

    def stop_thread(self):
        self.stop_signal.emit()  # emit the finished signal on stop

    def show_info(self):
        self.info_popup.show()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    gui = Gui()
    sys.exit(app.exec_())
```