Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка WEB додатку для порівняння типів інтерфейсу сайтів»
за освітньою програмою: «12 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1912»

	(підпис студента)	/Влас СЕРБІН/ (Ім'я ПРІЗВИЩЕ)
Керівник:	(підпис)	/доц. Олександра ГОРБОВА/ (посада, Ім'я ПРІЗВИЩЕ)
Нормоконтролер:	(підпис)	/доц. Світлана ВОЛКОВА/ (посада, Ім'я ПРІЗВИЩЕ)

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

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Development of a WEB application for comparison of WEBsite interface types»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ___: /Vlas SERBIN/

Scientific Supervisor: /Oleksandra GORBOVA/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Сербіну Власу Андрійовичу
1. Тема роботи: «Розробка WEB додатку для порівняння типів інтерфейсу сайтів»
Керівник роботи: Горбова Олександра Вікторівна, доцент
затверджені наказом № __ ст від __.__.202_
2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи:

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
Вступ
Збір та аналіз вимог до WEB додатку
Проектування WEB додатку
Розробка WEB додатку
Тестування WEB додатку
Загальні висновки
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
презентація,
відео роботи програми.

КАЛЕНДАРНИЙ ПЛАН

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	__.__.23 –__.__.23
Робочий проект	Програмування та відлагодження програми.	__.__.23 –__.__.23
	Тестування програми	__.__.23 –__.__.23
	Розробка, узгодження і затвердження програмної документації.	__.__.23 –__.__.23
	30 % виконаної роботи.	01.05.23 – 07.05.2023
	60 % виконаної роботи.	22.05.23 – 28.05.2023
	100 % виконаної роботи..	12.06.23 – 18.06.2023
	Подання кваліфікаційної роботи до кафедри	__.__.23
	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	__.__.23

Студент _____ Влас СЕРБІН
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олександра ГОРБОВА
(підпис) (Ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 7 розділів:

– вступ – у розділі описується сутність розробки WEB додатку, його актуальність існування та використання на даний момент. Складається з 2 сторінок;

– збір та аналіз вимог до WEB–додатку – у розділі описуються типи інтерфейсів сайтів, їх розбіжності та приклади застосування, визначення коефіцієнтів для оцінки та порівняння між собою. Складається з __ сторінок;

– проектування WEB додатку – у цьому розділі описується задача створення проекту, його функціональні вимоги, вхідні та вихідні дані, вибір мови програмування та підходу розробки, проектується інтерфейс та зовнішній вигляд сторінки, кількість та типи компонентів, їх зовнішній вигляд. Складається з __ сторінок;

– розробка WEB додатку – у цьому розділі описується розробка динаміки системи, сторінки WEB додатку разом з інтерфейсом, навігацією та дизайн. Складається з __ сторінок.

Тестування та налагодження WEB додатку – у цьому розділі виконується вибір методу тестування для розробленого WEB–додатку. Складається з __ сторінок.

– загальні висновки – у цьому розділі підводяться підсумки всієї роботи. Складається з 2 сторінки;

– список використаних джерел – у цьому розділі наводиться бібліографічний список використаної літератури. Складає 2 сторінки;

– додатки – у цьому розділі міститься технічне завдання, код робочого проекту.

Кількість таблиць: _ таблиць. Кількість рисунків: __ рисунків [1].

ЗМІСТ

ВСТУП

«Development of a WEB application for comparison of WEBsite interface types» – це самостійно спроектований та розроблений WEB–додаток з використанням компонентного підходу для підрахунку тематичних компонентів у коді сторінки, які мають відповідати напряму інформації на ній [1].

Поняття «WEB–додаток» являє собою програмне забезпечення, яке працює у будь якому браузері, що робить його універсальним для виконання поставленої задачі на будь якому пристрою, не потребує додаткової машинної потужності для виконання розрахунків.

Компонентноорієнтовианий підхід дозволяє суттєво зменшити об’єм програмного коду, шляхом використання раніше створених компонентів та їх задіяння у будь якому місці програми зі зміною вхідних параметрів, що дозволить значно зменшити кількість можливих помилок та прискорити розробку WEB–додатку для скорішого вирішення поставленої задачі.

WEB–додаток призначений для виконання аналізу та розпізнавання типу інтерфейсу сайтів шляхом зчитування його програмного коду з файлу типу TXT. Від кількості та типів задіяних компонентів при реалізації сторінки додаток буде аналізувати та розпізнавати який саме тип інтерфейсу був реалізований.

Використовуючи цей WEB–додаток, користувач має змогу проаналізувати вибрану ним сторінку та дізнатися кількість компонентів, їх тип, роль та задачу які вони виконують і вже на базі цих обчислень додаток виведе варіанти більш підходящих типів інтерфейсів.

Функціональне призначення – програмний продукт має виконувати підрахунок задіяних компонентів у програмному коді сайту та на базі цих обчислень продемонструвати варіанти типів інтерфейсу, які будуть мати наближені значення до отриманих.

Експлуатаційне призначення – за допомогою цього програмного продукту для розробника WEB сторінок полегшується визначення типу вибраного нами інтерфейсу сайту, виконавши аналіз внутрішніх його частин коду [3].

Актуальність роботи – на сьогоднішній день розробка сайтів включає у себе не тільки вміння писати програмний код, але й розумітися на таких поняттях як UX/UI. User experience та user interface забезпечать найкращий досвід користування програмними додатками, дозволяють проектувати взаємодію з інтерфейсом та кроки, які необхідно буде виконати для виконання задачі поставленою перед користувачем. Типів інтерфейсів користувача є певна кількість, усі вони націлені на певні напрямки та категорії, у одному сайті можуть бути використані різні типи та підтипи які різняться між собою візуальними деталями та практичним застосуванням [4].

«Development of a WEB application for comparison of WEBsite interface types» – додаток, який не зможе в повній мірі замінити практичні навички розпізнавання типів, але дозволить максимально швидко звузити коло варіантів, підраховуючи та аналізуючи кількість компонентів, їх стилі замість користувача, після чого виведе результати обчислень на екран у вигляді діаграми та списку більш задовольняючих за описом типів інтерфейсів, що буде економити певну кількість часу та спрощуючи процес аналізу [6].

РОЗДІЛ 1. ЗБІР ТА АНАЛІЗ ВИМОГ ДО WEB ДОДАТКУ

1.1 Різновиди сайтів

Для того щоб визначати тип інтерфейсу сайту необхідно розглянути загальні напрями їх діяльності та причини створення. На сьогоднішній день WEB–сторінки є невід'ємною частиною нашого повсякденного та робочого життя. Читання новин, купівля товарів, робота – для кожної з цих задач існують свої, цільові, типи інтерфейсів які повною мірою покривають запити користувача [5].

Види типів сайтів за сферою діяльності:

- сайти з продажу фізичних товарів чи послуг;
- сайти для оцифровування бізнесу;
- сайти для отримання комісій від угод;
- сайти для продажу платних онлайн–послуг;
- сайти для заробітку на рекламі;
- сайти для продажу підписок;
- сайти для взаємодії з громадянами.

Сайти з продажу фізичних товарів чи послуг – це тип WEB–сторінок, які створюються задля продажу та рекламування товарів чи послуг. Їх наповнення залежить від категорії товарів, але загалом складається з текстових блоків з контактною інформацією про компанію виробника.

Підтипи сайтів з продажу фізичних товарів та послуг:

- односторінковий сайт;
- сайт–візитка;
- корпоративний сайт;
- промо–сайт;
- інтернет–магазин;
- сайт–каталог.

Односторінкові сайти розробляються з ціллю спонукання відвідувачів до реєстрації та купівлі певних товарів та послуг. Проста будова дозволяє передбачити дії користувача та розмістити інформацію у необхідній послідовності, щоб цінність продукту для клієнта була найвища. Односторінокові сайти поділяють на цільові, рекламні та вірусні [7].

Особливості односторінкових сайтів:

- простота будови;
- вузька направленість;
- підвищують рівню відліку та купівлі товару користувачем.

Сайти–візитки розробляються з ціллю ознайомлення користувача із запропонованими послугами. Наповнені основною інформацією про товар, інформацією про його призначення та контактною інформацією про виробника [7].

Особливості сайту–візитки:

- простота будови;
- вузька направленість;
- надання детальної інформації про компанію та її контактні дані.

Корпоративний сайт розроблюється з ціллю інформування користувача про компанію, її товари та діяльність, виступають інформаційним центром компанії в інтернеті. Наповнені великим обсягом інформації про діяльність компанії та підтримують функціонал заведення особистого кабінету користувача для надання йому більш детальної інформації [7].

Особливості корпоративних сайтів:

- детальний опис компанії;
- наявність особистого кабінету з додатковою, внутрішньою інформацією;

– наявність корпоративного сайту надає користувачу більше впевненості щодо створення замовлення.

Промо–сайти розробляються з ціллю виділення певного товару та збільшення його продажів, шляхом проведення торгівельних акцій та заходів. Наповненість має вигляд рекламного буклету та є інтерактивною, з можливістю вписати промо–коди та зареєструватися для участі у заходах, привертає увагу та надає дані про організацію, її сферу діяльності, товари, послуги, контакти, акції та новини [7].

Особливості промо–сайтів:

– пряма реклама певної продукції;
– детальне представлення даних;
– інтерактивність.

Інтернет–магазини розробляються з ціллю організувати товаро оберт онлайн шляхом перенесення торгівельних полиць на екрани моніторів та можливістю замовити їх за одну мить. Інструмент який має набір функціоналу для забезпечення пошуку товарів, занесення товарів у кошик, здійснення купівлі з варіантами оплати та здійснення доставки. Для більшої зручності користування інтернет магазинами користувач має змогу створити свій особистий кабінет.

Особливості інтернет–магазинів:

– особистий кабінет користувача;
– автоматизація процесів поширення товарів;
– аналіз контрою показників прибутковості;
– системи накопичування знижок та бонусів.

Сайти–каталоги розробляються з ціллю надання інформації про товар, які потребують більш детальної консультації перед купівлею. Насичені зображеннями та детальним описом товарів з інструкціями користування,

технічними характеристиками, але при цьому не мають особистого кабінету та можливість придбання товару онлайн. Можуть виступати доповненнями до основного сайту чи інтернет магазину організації.

Особливості сайтів–каталогів:

- розгалужена структура;
- повна інформація про надання послуг;
- доповнення до основного сайту.

Сайти для цифровізації бізнесу – це тип WEB–сторінок, який вирізняється на фоні інших спрямованістю на автоматизацію бізнес–процесів всередині компанії.

Автоматизація допомагає покращити показники ведення бізнесу за рахунок використання та розробки модулів, які забезпечують комунікацію, ведення звітностей та виконання цільових завдань.

Види сайтів для цифровізації бізнесу:

- CRM–система;
- ERP – система;
- Б2Б портал.

CRM системи розробляються для авматоматизації оброблення та продажу заявок, які надходять у компанію. За допомогою цієї системи ведеться облік клієнтської бази, її швидке оновлення та використання [7].

Особливості CRM системи:

- багатозадачність;
- автоматизація обліку клієнтської бази.

ERP системи розробляються для комплексного управління ресурсами. За допомогою неї можна виконувати розподіл операцій підприємства та планувати їх використання, автоматизувати управління виробництвом, персоналом та обліком [7].

Особливості ERP системи:

- багатозадачність;
- автоматизація управління,
- можливість планування.

Б2Б порталів розробляється для закритої взаємодії між компанією та її дистриб’ютором. Насиченість порталу складається з відображення характеристик товару, його наявності, також є можливість аналізування та збору статистики [7].

Особливості Б2Б порталів:

- багатозадачність;
- автоматизація аналізу.

Сайти для отримання комісій від угод – це тип WEB–сторінок, який являє собою втілення окремого бізнесу на проведенні угод, або від надання послуг становлення гарантом.

Види сайтів для отримання комісій від угод:

- онлайн–обмінник;
- кешбек сайти;
- біржі фрілансу;
- дропшипінг платформа.

Онлайн обмінники розроблюються для обміну валют в онлайн режимі. Основний функціонал побудований на тому, що користувач обмінює валюту на іншу з невеликою комісією. Такі види наділені верифікацією користувачів через їх документи.

Особливості онлайн обмінників:

- багатозадачність;
- автоматизація купівлі–продажу.

Кешбек–сайти розробляються для залучає користувачів до певних магазинів з нарахуванням за це певної суми, яка після цього розприділяється мі власником та покупцем.

Особливості кешбек–сайтів:

- пряма реклама певної продукції;
- ефективна реклама.

Біржі фрілансу розробляються для організації укладання угоди між користувачами, задля їх безпечного проведення за що і береться комісія. Мають широке коло застосування.

Особливості бірж фрілансу:

- мобільність;
- автоматизація купівлі–продажу;
- безпека.

Дропшипінг–платформи розробляються для організації продажів товарів виробників. Цей процес дозволяє бути тільки посередником не витрачаючись на купівлю своїх товарів та оформлення складів, що дає можливість заробляти на встановленій націнці на товар.

Особливості дропшипінг–платформ:

- багатозадачність;
- зручність.

Сайти для продажу платних онлайн послуг – це тип WEB–сторінок, що розробляються для заробітку через виставлення вакансій та оголошень.

Види сайтів для продажу платних онлайн послуг:

- сайт з пошуку роботи,
- дошки оголошень,
- маркетплейси;

– онлайн школи.

Сайт з пошуку роботи розробляються для того, щоб компанії виставляли свої оголошення про найм працівників. Дохід йде за розміщення певної кількість оголошень, перегляд анкет.

Особливості сайтів з пошуку роботи:

– вузька направленість.

Дошки оголошень розробляються з метою поширення оголошень про продаж чи купівлю товарів. Дохід отримується за рахунок підняття оголошень у топ списку за додаткову комісію, але й є можливість безкоштовної публікації [7].

Особливості дошок оголошення:

– прибутковість;
– зручність.

Маркетплейси розроблюється для розміщення списків компаній та виробників, які пропонують товари та послуги. Користувач може вибрати товар порівнявши ціни від виробників, а розробник маркетплейсів заробляє на комісії не продаючи власні товари.

Особливості маркетплейсів:

– варіативність;
– зручність.

Онлайн школи розробляються для взаємодії між викладачами та учнями. За її допомогою спрощується контроль виконання роботи та покращується процес навчання.

Особливості онлайн шкіл:

– прибутковість;
– зручність.

Сайти для заробітку на рекламі – це тип WEB–сторінок, який являє собою інформаційні, або розважальні портали з додаванням до них рекламних оголошень.

Види сайтів для заробітку на рекламі:

- сайт новин;
- інформаційні портали;
- блог;
- агрегатори;
- відеохостинг.

Сайт новин розробляється для розповсюдження останніх новин певного регіону, або всього світу. Основною ціллю є залучення якомога більшої кількісті людей та отримувати кошти за розміщення банерної реклами.

Особливості сайтів новин:

- прибутковість;
- зручність.

Інформаційні портали розробляються для розповсюдження новин, які поділені за категоріями інформації та реклами.

Особливості інформаційних порталів:

- прибутковість;
- зручність.

Блоги розробляються для компаній або окремих людей, де поширюються пости про своє життя, або ж професійний напрямок, не потребує складної архітектури та легко задіюється. Заробіток за допомогою рекламних вставок.

Особливості блогів:

- прибутковість;

– зручність.

Агрегатори розробляються для сортування інформації та пропозицій користувачам за певними напрямками. По функціям нагадує каталог з великою кількістю товарів.

Особливості агрегаторів:

– прибутковість;
– зручність.

Відеохостинг розробляються для перегляду або завантаження мультимедії.

Особливості відеохостингів:

– прибутковість;
– зручність.

Сайти для продажу підписки – це тип WEB–сторінок, який орієнтується на продажі підписок користувачам на ресурси, які надають певні мультимедійні послуги.

Види сайтів для продажу підписки:

– онлайн–кінотеатри,
– конструктори сайтів;
– тематичні портали.

Онлайн кінотеатри розробляються для перегляду фільмів або серіалів за певну щомісячну плату.

Особливості онлайн кінотеатрів:

– доступ до мультимедії;
– зручність.

Конструктори сайтів розробляються для полегшення створення сайтів з їх подальшим використанням.

Особливості конструкторів сайтів:

– легкість налаштування.

Тематичні портали розробляються для того, щоб користувач міг детально вивчити конкретну тему з унікальною інформацію певного напряму.

Особливості тематичних порталів:

– надання актуальної інформація.

Сайти для взаємодії із громадянами – це тип WEB–сторінок для інформування населення про якісь внутрішні зміни та зручного доступу до використання державних послуг.

Види сайтів для взаємодії із громадянами:

– сайт міста;
– портал надання електронних послуг.

Сайти міста розроблюється для прямого інформування жителів цього міста про зміни керівного складу органів управління, організацію заходів.

Особливості сайтів міста:

– швидкий доступ до інформації;
– універсальність.

Портал надання електронних послуг розробляються для швидкого та зручного надання державних послуг онлайн, не перебуваючи у фізичних чергах. Організація таких можливостей як складання петицій, отримання довідок.

Особливості порталів надання електронних послуг:

– швидкий доступ до інформації;
– універсальність.

1.2 Аналіз особливостей та визначення коефіцієнтів

Основною задачею аналізування особливостей наповненості WEB–сторінок є визначення критеріїв, коефіцієнтів, за допомогою яких ми будемо визначати, які з частин програмного коду використовуються більше разів, мають більший вплив на донесення інформації та взаємодії з користувачем. Це дасть нам змогу, опираючись на розрахунки, припустити, до якого типу або типів відноситься вибрана нами WEB–сторінка.

Коефіцієнтами будемо називати компоненти WEB–сторінки. Їхня кількість напряму буде впливати на спосіб донесення інформації та на галузь використання сторінки. Наприклад, якщо на нашій WEB–сторінці більшу частину коду будуть займати компоненти текстового вигляду, то ми зможемо віднести цю сторінку до чотирьох видів типів сайтів за їхніми сферами діяльності. Це будуть: сайти з продажу фізичних товарів, чи послуг, сайти для продажу платних онлайн–послуг, сайти для заробітку на рекламі, сайти для продажу підписок та сайти для взаємодії із корисувачами. Більша частина типів сторінок будуть вказувати на переважаючу кількість текстових компонентів, бо для опису послуг, донесення новин та рекламування загалом і використовується текстова інформація, але враховуючи інші компоненти, які знаходяться на цій же сторінці, ми побачимо між ними вже більшу різницю, адже розбавляючи текстові компоненти – компонентами типу зображення, або типу кнопки ми вже можемо отримати не припущену статтю, а інтернет магазин [7].

Описуючи коефіцієнти, попередньо були вже озвучені деякі різновиди програмних частин за якими буде проводитися аналіз. Програмні частини WEB–сторінки будуть поділятися за своєю змістовою складовою та напрямом використання, визначати, в залежності від їхньої кількості на сторінці, їхнє типове надлежання.

Різновиди інформаційних блоків WEB–сторінки:

- компоненти типу текст;
- компоненти типу зображення;
- компоненти типу кнопки;
- компоненти типу списку;
- об’єднані компоненти.

До об’єднаних компонентів будуть належати компоненти, які об’єднують у собі певну кількість дочірніх компонентів.

1.3 Постановка задачі

WEB додаток «Development of a WEB application for comparison of WEBsite interface types» має проводити процедуру розпізнавання типів інтерфейсів сайту шляхом зчитування та його програмного коду.

В залежності від кількісті компонентів, їхнього типу та напряму донесення інформації визначається тип сайту, під який підходить проаналізований. Програмний додаток виводить пораховану кількість всіх програмних частин коду у вигляді секторної діаграми та встановлює тип сайту/сторінки за результатами розрахунків.

Користувач, використовуючи цей додаток, може на основі визначених компонентів розмітки провести розрахунки коефіцієнтів (кількості компонентів) та визначити більш підходящі варіанти видів типів інтерфейсів.

Просканувавши файл типу TXT, у якому буде знаходитися код сторінки, ми отримаємо кількість використання компонентів та зможемо побачити обсяг який вони займають на секторній діаграмі. Таким чином, розрахувавши обсяг всієї інформації на сторінці, поділивши їх на складові частини та порівнявши з очікуваними коефіцієнтами ми зможемо зробити припущення щодо виду типу WEB–сторінки.

Висновки до розділу 1:

Результатами роботи над першим розділом були проаналызовані різновиди WEB–сторінок, які поділяються за своїм напрямом, та за цими напрямами мають бути вибрані необхідні нам критерії оцінки інтерфейсу для подальшого аналізу та висунення припущення, щодо його типу. У результаті вивчення у нас вийшло відокремити їх тематики, їх призначення та поділити їх на окремі самодостатні частини. Таким чином ми будемо аналізувати вхідний код сторінки порівнюючи його з вже записаними коефіцієнтами у програмі, для кожного з типів та підтипів окремо.

Після розглянення типів та підтипів інтерфейсів, ми виділили шукані нами частини коду, які і будуть відповідати за проведення аналізу та порівняння:

Різновиди програмних частин WEB–сторінки.

- компоненти типу текст;
- компоненти типу зображення;
- компоненти типу кнопки;
- компоненти типу списку;
- об’єднані компоненти.

Під час проектування WEB–додатку, необхідно визначити:

- подальші задачі проекту;
- функціональні вимоги до проекту;
- вхідні та вихідні дані проекту, включаючи коефіцієнти, за якими ми будемо порівнювати вже записані у програму типи та підтипи інтерфейсів;
- вибрати мову програмування та підхід до розробки WEB–сторінки;
- спроектувати екран, інтерфейс на ньому та дизайн усіх складових
- описати структуру компонентів, з яких складається наша система.

РОЗДІЛ 2. ПРОЕКТУВАННЯ WEB–ДОДАТКУ

2.1 Задачі проекту

На основі проведеного аналізу різновидів сайтів та WEB–сторінок, метою якого було визначення можливих варіацій та напрямів інформаційного посилання цих ресурсів ми можемо сформувати подальші задачі проектування розроблювальної WEB–сторінки та проведення аналізу:

- проектування функціоналу зчитування вмісту файлу до програми;
- проектування екрану WEB–додатку, на якому буде відображений зміст текстового документу із програмним кодом;
- проектування внутрішніх списків програми з очікуваними коефіцієнтами оцінки різновидів;
- проектування алгоритму розпізнавання та підрахунку коефіцієнтів;
- проектування вікна виведення результатів підрахунків у вигляді секторної діаграма;
- проектування вікна виведення результатів підрахунків у вигляді списку можливих варіантів видів типів інтерфейсів WEB–сторінок.

2.2 Функціональні вимоги

Програмний продукт має забезпечити можливість вибору файлів, для зчитування їх вмісту, наступного типу – TXT.

Програмний продукт має виконувати розрахунки шляхом зчитування програмних частин та подальшого розрахунку коефіцієнтів та впорядкування.

Програмний продукт має забезпечити можливість демонстрації вмісту файлу, за яким відбувається аналіз з підсвічуванням певних компонентів наступними кольорами:

- компоненти типу текст – зелений колір;
- компоненти типу зображення – червоний колір;

– компоненти типу кнопки – блакитний колір;
– компоненти типу списку – жовтий колір;
– об’єднані компоненти – фіолетовий колір.

Програмний продукт має забезпечити можливість демонстрації результатів розрахунків шляхом відображення секторної діаграми.

Програмний продукт має забезпечити можливість демонстрації результатів розрахунків шляхом відображення списків варіантів.

Діаграма прецедентів зображена на рисунку 2.1

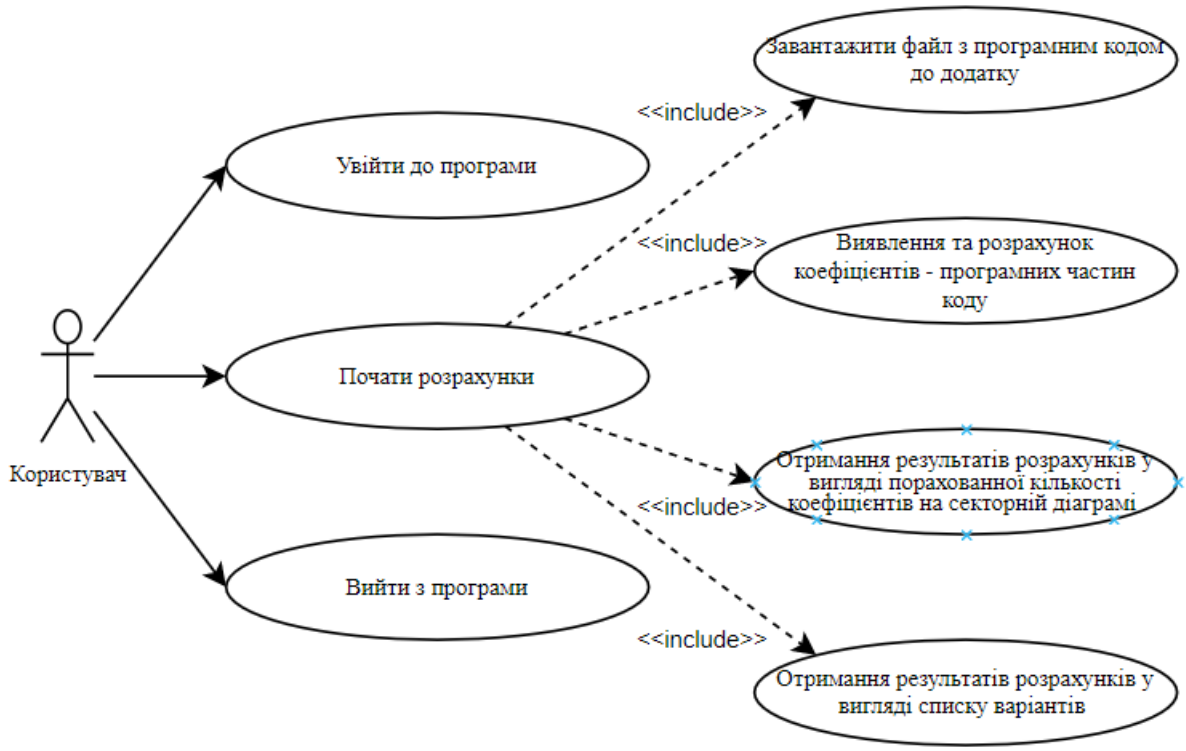


Рисунок 2.1 – Діаграма прецедентів

2.3 Вхідні та вихідні дані

На початку роботи WEB–додаток має наступні вхідні дані:

– файл з вмістом програмного коду над яким будуть проводитися розрахунки;

- набір однакових за складом списків очікуваних оціночних коефіцієнтів розрахунків;
- однаковий склад типів коефіцієнти, які вказують на кількість компонентів програмних частин коду, .

Коефіцієнти компонентів програмних частин коду:

- counter_text_component – компоненти типу текст;
- counter_image_component – компоненти типу зображення;
- counter_button_component – компоненти типу кнопки;
- counter_list_component – компоненти типу списку;
- counter_collection_component – об’єднані компоненти.

WEB–додаток має наступні вихідні дані:

- відображення результатів у вигляді секторної діаграми;
- відображення результатів у вигляді списку коефіцієнтів.

2.4 Вибір мови програмування

Розроблення WEB–додатку виконується за допомогою мови JavaScript з використанням бібліотеки React, який дозволяє розробляти WEB–сторінки шляхом створення налаштованих компонентів. Створення компоненту практично дозволяє створювати та використовувати власний шаблон інтерактивних частин сторінки, написавши будову та стилі для нього одноразово, але при цьому з можливістю використовувати стільки разів, скільки необхідно розробнику [12].

Робота з фреймворком цілком виконується мовою JSX, який з легкістю оброблює вирази на мові JS. У свою чергу це розширення мови JS, що дає змогу декларативно створювати власні компоненти.

Можливості JSX:

- проста декларативна розмітка;

- розміщення розмітки та коду одне;
- принцип поділу відповідальності;
- однакова реалізація для різних платформ.

При розробці проектів на бібліотеці React ми звертаємося до двох інноваційних підходів:

- задіяння архітектури Flux, що дозволяє реалізувати односпрямовану прив’язку даних;
- можливість реалізувати незмінність стану компонента.

Після створення наших компонентів їхній стан не змінюється в процесі виведення його на екран, не змінюється поки він рендериться, до цього додається можливість застосування ефектів для впливу на нього та роботи з подіями. Більшість компонентів мають вигляд чистих функцій з вхідними параметрами [13].

2.5 Опис компонентно-орієнтованого підходу

Компонентно орієнтований підхід організований на взаємодії різних типів компонентів, які знаходяться у певних полях – контейнерах. Створюючи компонент можна сказати, що він являє собою окрему функцію з описом її дій всередині. Компоненти відповідають за відображення та зміну наповненості сторінки, дозволяють модифікувати та вносити зміни у інтерфейс.

Даний підхід використовує успадкування з принесенням інновацій:

- здатність самоопису;
- модульність розробки;
- здатність зберігання та відновлення станів;
- багаторазове використання створених компонентів.

Використання даного підходу дозволяє припускати меншу кількість помилок, бо ми використовуємо одну структуру для реалізації компонентів, які мають один сценарій відпрацювання.

Проблеми, які вирішує компоненто–орієнтований підхід:

- проблема складності контрою ЖЦ об’єкта – зберігання та відновлення стану об’єкту;
- недостатній контроль даних – компоненти є складовою контейнерів у яких вони знаходяться, то при знищенні контейнера вся пам’ять звільняється.

2.6 Проектування екранів WEB–додатку

Екран сторінки має відображати:

- компонент який буде відображати вміст файлу, який ми аналізуємо;
- компонент, який буде відображати результати аналізу у вигляді секторної діаграми;
- компонент, який буде відображати результати аналізу у вигляді списку коефіцієнтів.

2.7 Проектування інтерфейсу WEB–додатку

Інтерфейс WEB–додатку має дозволяти користувачу взаємодіяти з текстом, який завантажується з файлу, демонструвати лічильники коефіцієнтів та виводити результати аналізу на екран [12].

Для відображення вмісту файлу будуть задіяні компоненти контейнеру та компоненти для прокрутки:

- компоненти View;
- компоненти Scrollview.

Для відображення лічильників будуть задіяні компоненти контейнеру та компоненти для тексту:

- компоненти View;
- компоненти Text.

Для відображення інструментів взаємодії будуть задіяні компоненти контейнеру, компоненти кнопки та компоненти для тексту:

- компоненти View;
- компоненти Button;
- компоненти Text.

Для відображення результатів аналізу будуть задіяні компоненти контейнеру, компоненти кнопки та компоненти для тексту:

- компоненти View;
- компоненти Text.

2.8 Проектування дизайну WEB–додатку

Для кожного елементу інтерфейсу були розроблені та представлені ескізи їх зовнішнього вигляду. Для відображення всієї сторінки нам будуть потрібні наступні компоненти:

- компоненти View;
- компоненти Button;
- компоненти Scrollview;
- компоненти Text.

Компонент View виконує роль контейнера та розділяє сторінку й інші компоненти на окремі блоки, що дозволяє редагувати та переміщувати їх під наші потреби. Будуть мати при собі варіацію зрізання кутів в 60 , 30 та 15 градусів для текстових полів, кнопок та поля для відображення тексту з файлу. Кожен контейнер буде описаний чорною лінією.

Кількість полів та їх параметри зображені на рисунках 2.2 – 2.5:

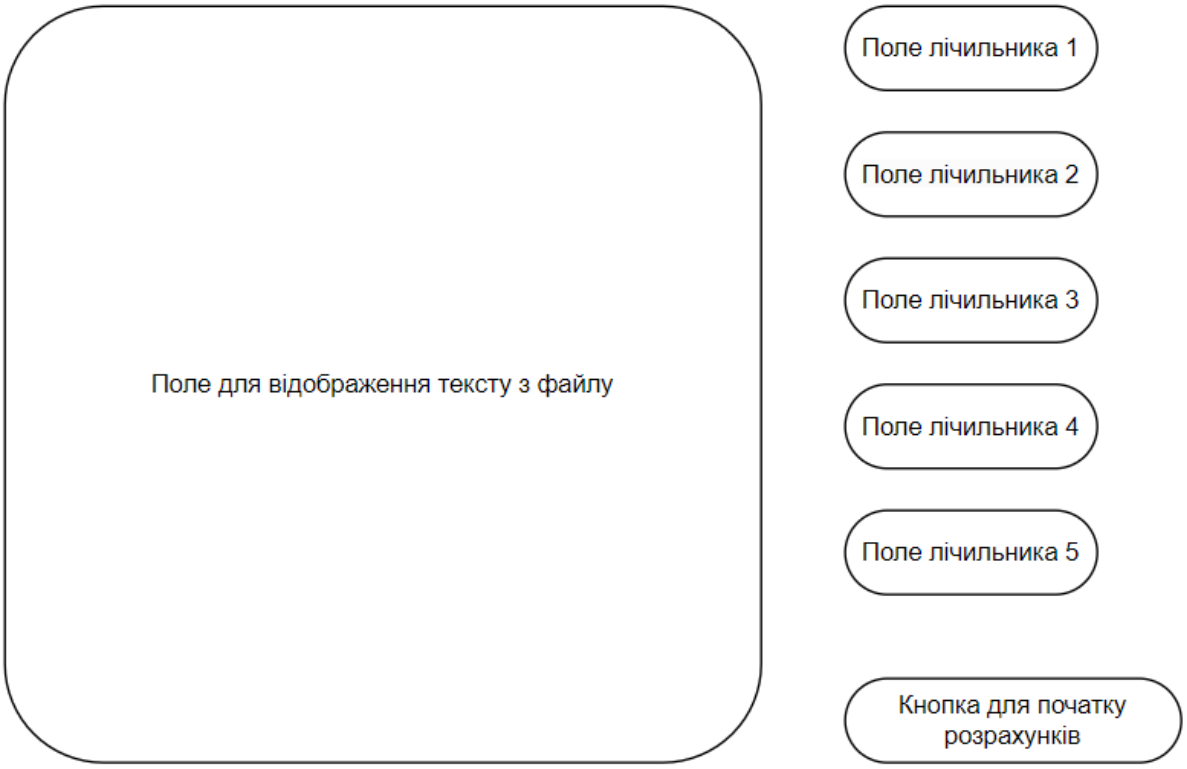


Рисунок 2.2 – поля та їх призначення на сторінці

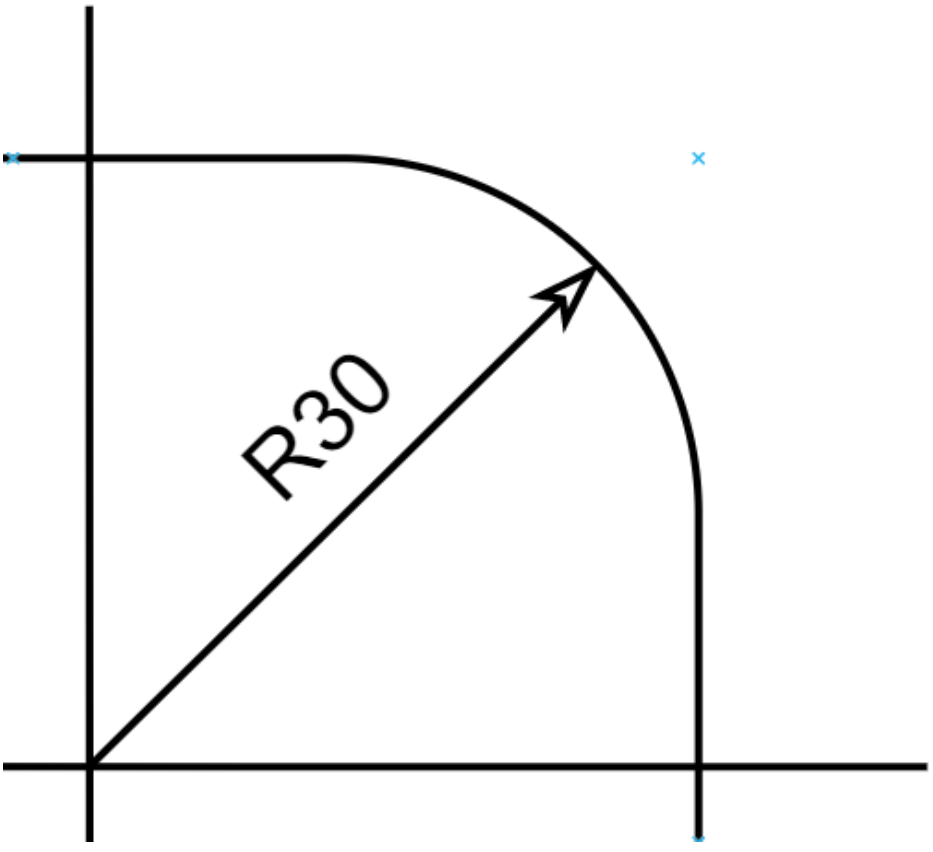


Рисунок 2.3 – радіус скручення текстового поля

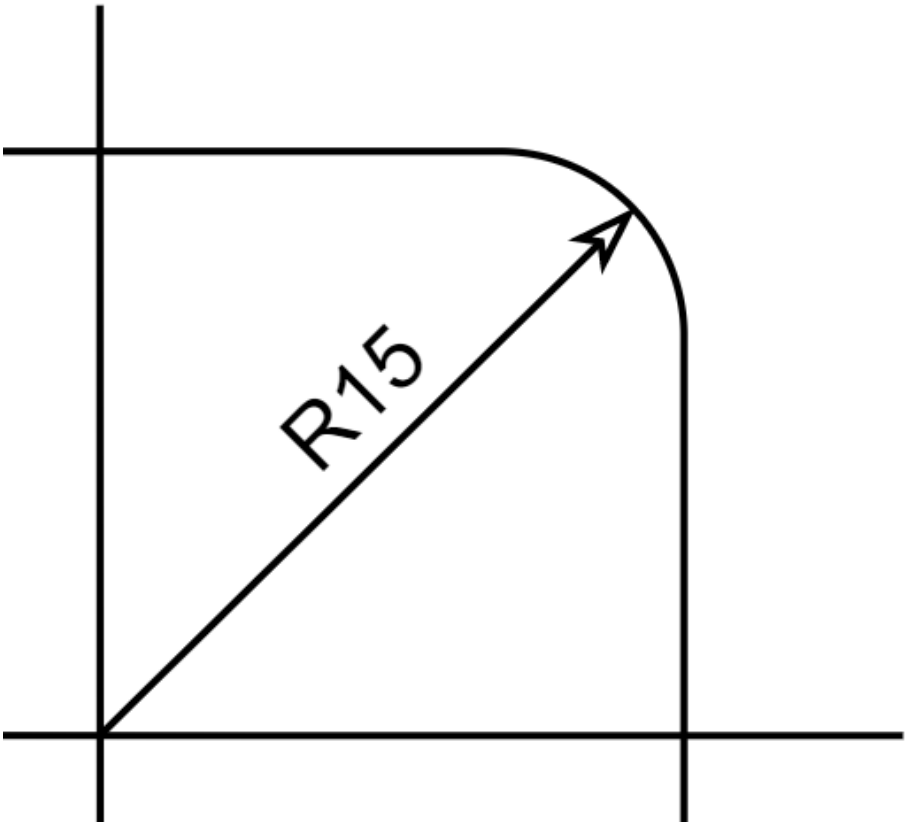


Рисунок 2.4 – радіус скручення прокручованого текстового поля

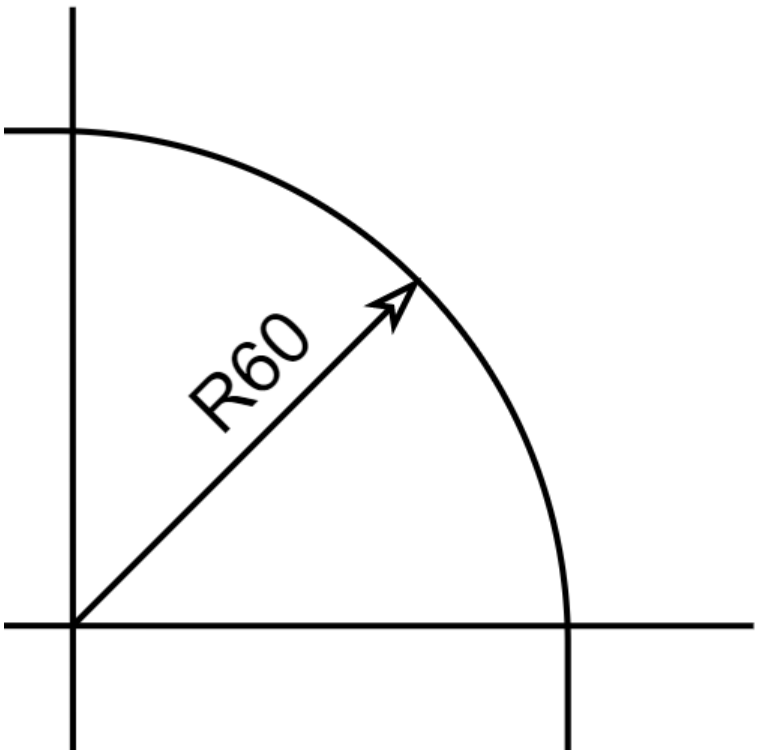


Рисунок 2.5 – радіус скручення поля для кнопок

Компонент Text виконує роль текстового поля. За його допомогою відображається інформація з лічильників, задача яких знаходити тематичні частини коду, напис на кнопках та текстовий вміст файлу:

- counter_text_component;
- counter_image_component;
- counter_button_component;
- counter_list_component;
- counter_collection_component.

Кожне текстове поле з лічильником буде мати свій окремий колір підсвічування тематичної частини коду. Матиме радіус скручення кутів 30 градусів. Кольори вибрані наступні:

- counter_text_component – зелений;
- counter_image_component – червоний;
- counter_button_component – синій;
- counter_list_component – жовтий;
- counter_collection_component – фіолетовий.

Визначання кольорів під лічильники зображені на рисунку 2.6:

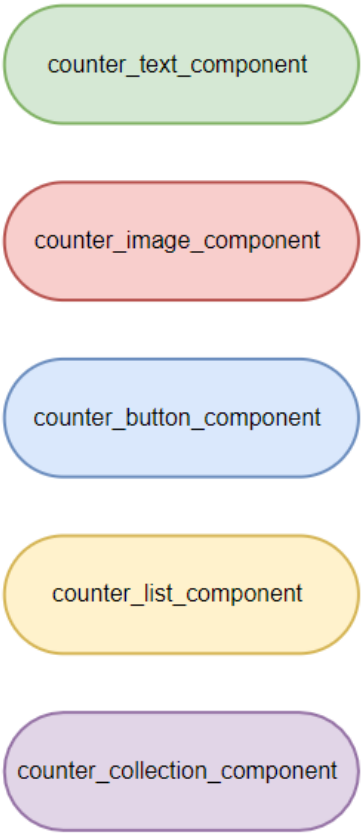


Рисунок 2.6 – лічильники та їх кольорові ідентифікатори

Компонент Scrollview виконує роль обгортки тексту, дозволяючи поміщати у неї багато текстової інформації та прокручувати за допомогою функції скролу до необхідного для нас місця. Матиме радіус скручення кутів 15 градусів для більш коректного відображення тексту, не закриваючи його більшим радіусом [12].

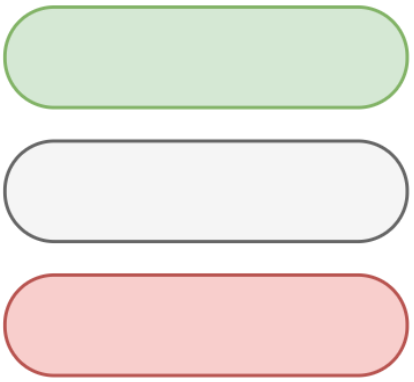
Компонент Button виконує роль взаємодії користувача та WEB–сторінки, за його допомогою користувач може почати аналіз тексту, визначити необхідні коефіцієнти та отримати результати аналізу. Матиме радіус скручення кутів 60 градусів та реакцію на взаємодії з користувачем.

Реакція взаємодії матиме три стани компоненту Button з кольоровим супроводом:

- стан ненатиснутий – зелений колір;
- стан натиснутий – червоний колір;

– стан під час натискання – сірий колір.

Реакції взаємодії зображені на рисунку 2.7:



Діаграма станів зображена на рисунку 2.7

Рисунок 2.7 – діаграма станів

2.9 Структура компонентів системи

Система компонентів зображена на рисунку 2.8

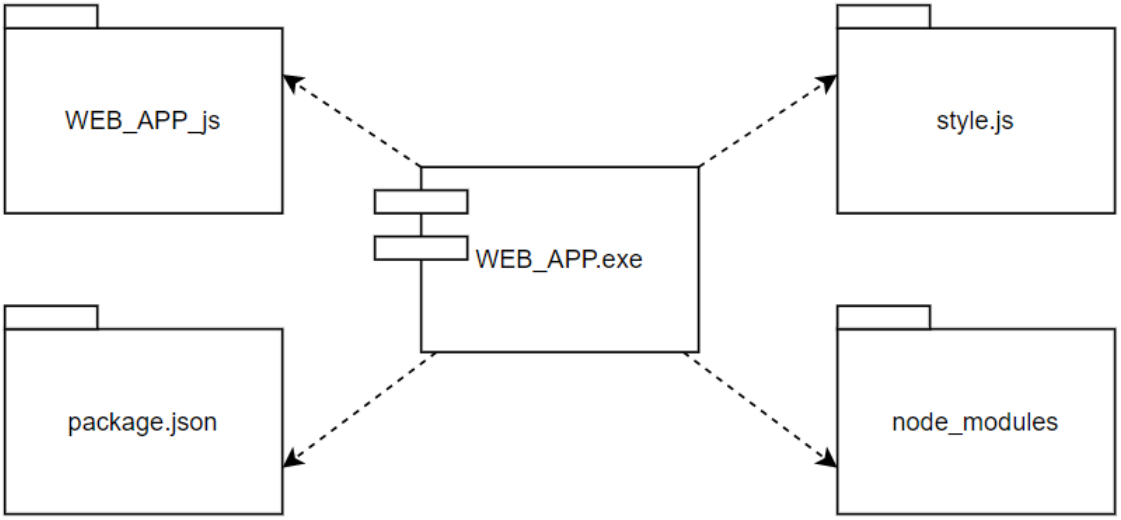


Рисунок 2.8 – система компонентів зображення

Висновки до розділу 2

Результатами роботи над другим розділом мають бути визначені та спроектовані задачі для подальшої розробки WEB–сторінки, сформовані функціональні вимоги, вхідні та вихідні дані проекту, що враховують в себе, створені нами, шаблони типів інтерфейсів, вибрана мова та підхід до програмування, описані екрани з інтерфейсом та дизайном.

Робота розпочалася з додаткового описання задач проектування проекту, вказано на моменти, які необхідно розглянути для подальшого проведення аналізу, визначення функціональних вимог та вхідних даних, що зазнали деяких змін та нововведень.

Були описані лічильники, за допомогою яких будуть обчислюватися наші коефіцієнти, визначені компоненти, з яких буде складатися наша WEB–сторінки, представлені їх ескізи та оформлення станів. Під час розробки ці

ескізи будуть зображені у вигляді відрендеренних компонентів на сторінці з демонстрацією виконання необхідних задач.

У процесі створення ескізів та прописання станів була визначена мова програмування, яка найкраще підійде для створення необхідних функціональних компонентів та підхід, з використанням якого написання коду буде лаконічним та більш безпечним, уникаючи більшу кількість можливих помилок та витрачаючи на створення набагато менше часу. Були зображені схеми:

- діаграма прецедентів;
- діаграма станів;
- система компонентів системи;
- схема станів інтерактивних компонентів;
- схема зовнішнього вигляду лічильників з їх кольоровим оформленням;
- схема скручення кутів блоків компонентів;
- схема полів та їх призначення на сторінці.

В кінці проектування був створений макет проекту з його визначеною структурою компонентів системи, за допомогою якого можна розпочати роботу з розробки зпроектованих нами компонентів та функціональних задач.

На основі проектування, при розробці необхідно реалізувати:

- динаміку системи;
- екран WEB–сторінки;
- інтерфейс WEB–сторінки;
- дизайн WEB–сторінки.

РОЗДІЛ 3. РОЗРОБКА WEB–ДОДАТКУ

3.1 Розробка динаміки системи

Діаграма діяльності зображена на рисунку 3.1:

Рисунок 3.1 – діаграма діяльності

Діаграма послідовності зображена на рисунку 3.2:

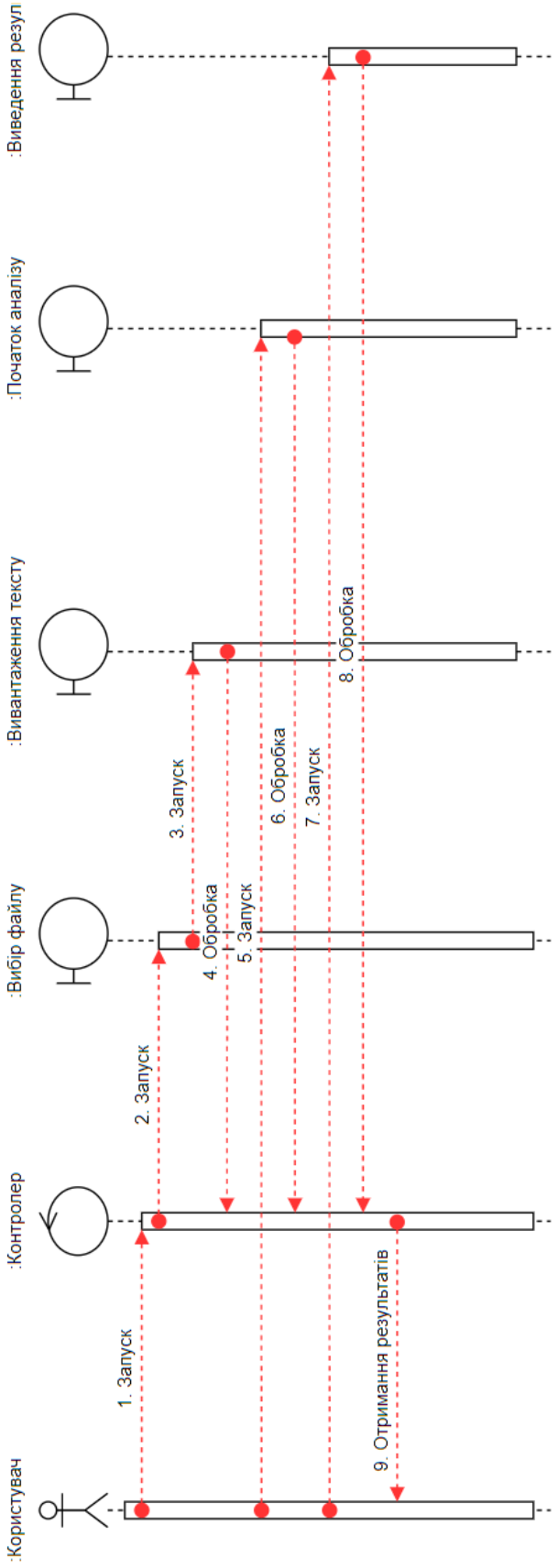


Рисунок 3.2 – діаграма послідовності

3.2 Розробка екранів WEB–додатку

WEB–сторінка буде налічувати один екран, якого цілком вистачить для відображення всього функціоналу проведення аналізування даних та відображення результатів.

Екран WEB–сторінки буде складатися з чотирьох рівнів, кожен з яких буде призначений для відображення інформації.

Перший рівень буде відображати вміст текстового файлу та налічуватиме 2 компоненти:

- компонент контейнеру – <View>;
- компонент області для прокручування – <Scrollview>.

Другий рівень буде відображати інформацію про стан лічильників та налічуватиме 6 наборів по 2 компоненти + 1 загальний контейнер:

- компонент контейнеру загальний – <View>;
- компонент контейнеру – <View>;
- компонент області для відображення тексту – <Text>;

Третій рівень буде відображати інформацію про результати аналізу у вигляду секторної діаграми, налічуватиме 6 наборів по 2 компоненти + 1 загальний контейнер:

- компонент контейнеру загальний – <View>;
- компонент контейнеру – <View>;
- компонент області для відображення тексту – <Text>;

Четвертий рівень буде відображати інформацію про результати аналізу у вигляду списку, налічуватиме 2 компоненти:

- компонент контейнеру – <View>;
- компонент області для прокручування – <Scrollview>.

Розміщення рівнів на сторінці зображено на рисунку 3.3:

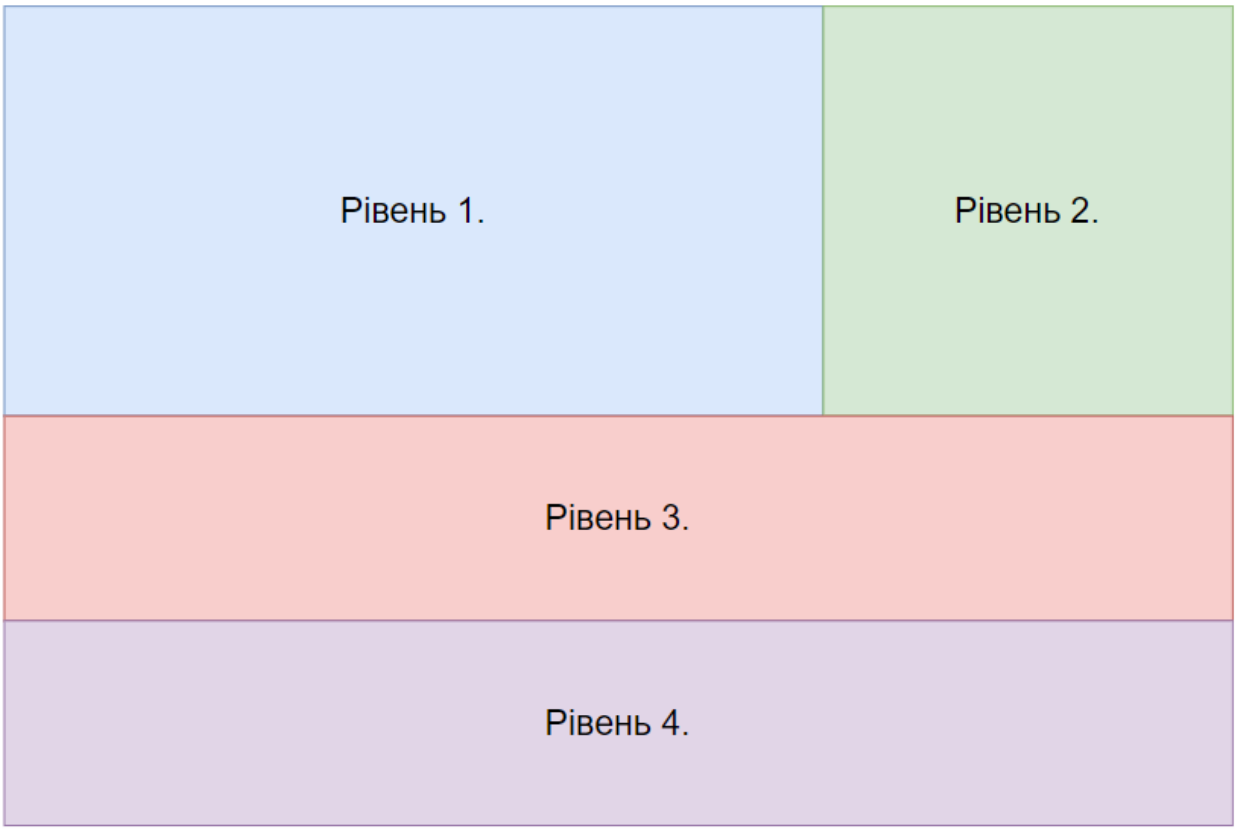


Рисунок 3.3 – розміщення рівнів на сторінці

3.3 Розробка інтерфейсу WEB–додатку

Ескіз зовнішнього вигляду інтерфейсу 1 рівня зображений на рисунку 3.4:

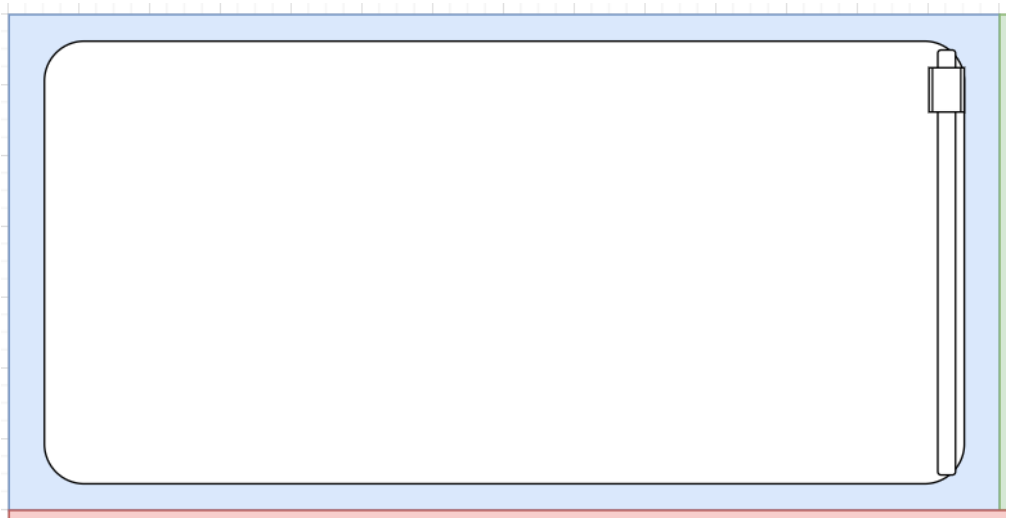


Рисунок 3.4 – ескіз зовнішнього вигляду інтерфейсу 1 рівня

Складова інтерфейсу 1 рівня зображений на рисунку 3.5:

```
<View>
  <View>
    <Scrollview>
      <Text>

      </Text>
    </Scrollview>
  </View>
</View>
```

Рисунок 3.5 – складова інтерфейсу 1 рівня

Ескіз зовнішнього вигляду інтерфейсу 2 рівня зображена на рисунку 3.6:

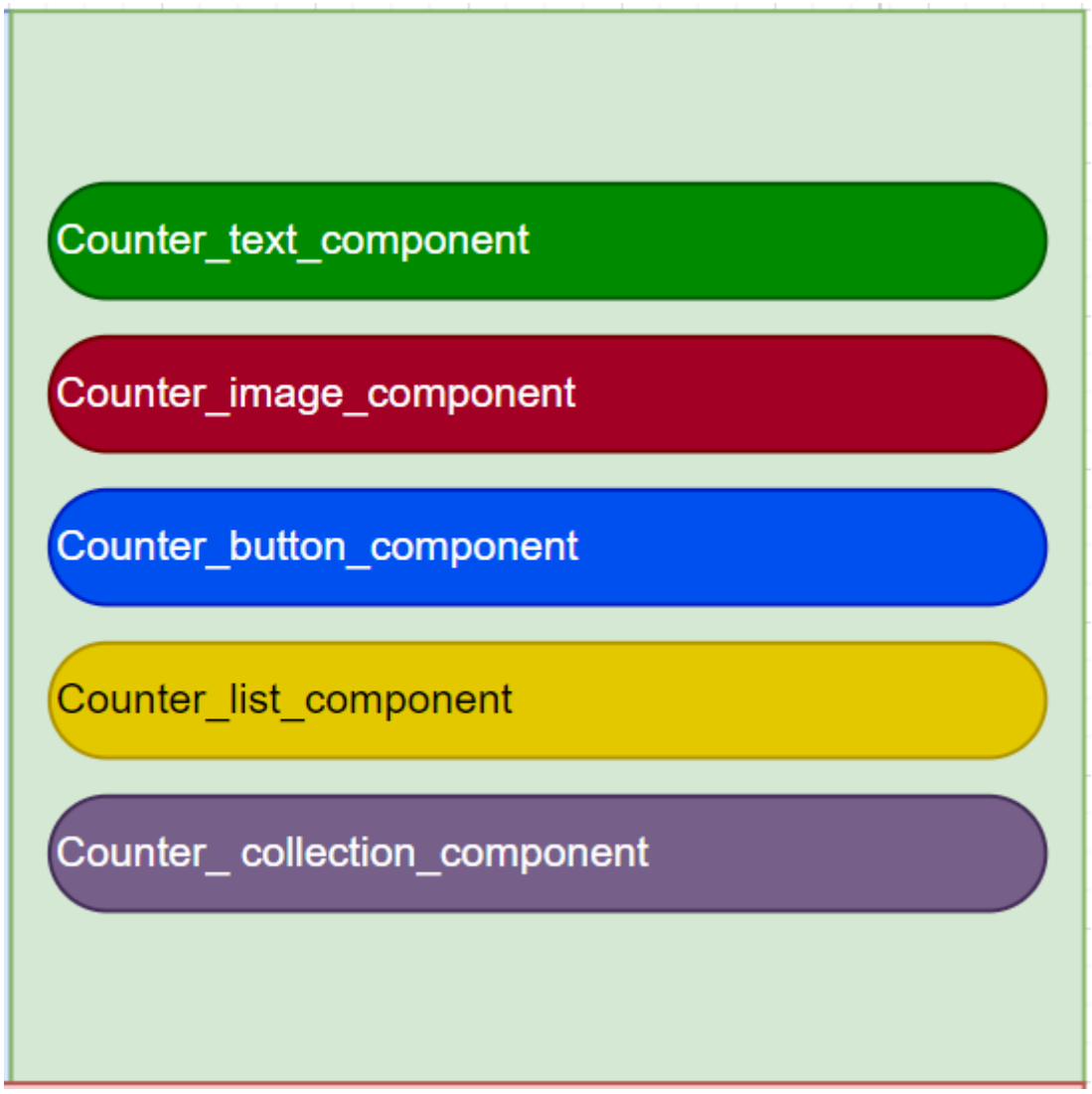


Рисунок 3.6 – складова інтерфейсу 1 рівня

Складова інтерфейсу 2 рівня зображена на рисунку 3.7:

```
<View>
  <View>
    <Text>Counter_text_component</Text>
  </View>
  <View>
    <Text>Counter_image_component</Text>
  </View>
  <View>
    <Text>Counter_button_component</Text>
  </View>
  <View>
    <Text>Counter_list_component</Text>
  </View>
  <View>
    <Text>Counter_collection_component</Text>
  </View>
</View>
```

Рисунок 3.7 – складова інтерфейсу 1 рівня

Ескіз зовнішнього вигляду інтерфейсу 3 рівня зображений на рисунку 3.8:

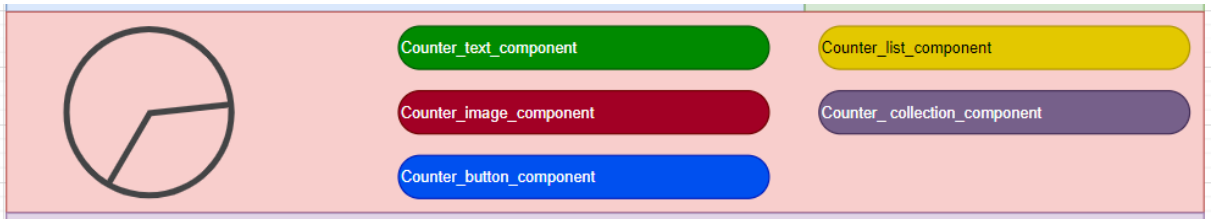


Рисунок 3.8 – складова інтерфейсу 3 рівня

Складова інтерфейсу 3 рівня зображена на рисунку 3.9:

```
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
  <PieChart style={styles.chart} data={pieData} />
</View>
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
</View>
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
</View>
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
</View>
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
</View>
<View style={styles.container}>
  <Text style={styles.title}>Pie Chart Example</Text>
</View>

</View>
```

Рисунок 3.9 – складова інтерфейсу 3 рівня

Ескіз зовнішнього вигляду інтерфейсу 4 рівня зображений на рисунку 3.10:

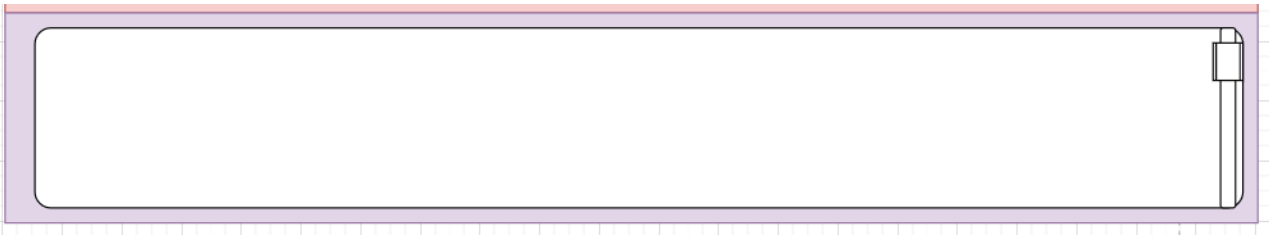


Рисунок 3.10 – складова інтерфейсу 3 рівня

Складова інтерфейсу 4 рівня зображена на рисунку 3.11:

```
<View>
  <View>
    <Scrollview>
      <Text>

      </Text>
    </Scrollview>
  </View>
</View>
```

Рисунок 3.11 – складова інтерфейсу 4 рівня

Ескіз зовнішнього вигляду інтерфейсу 1–4 рівнів зображений на рисунку 3.12:

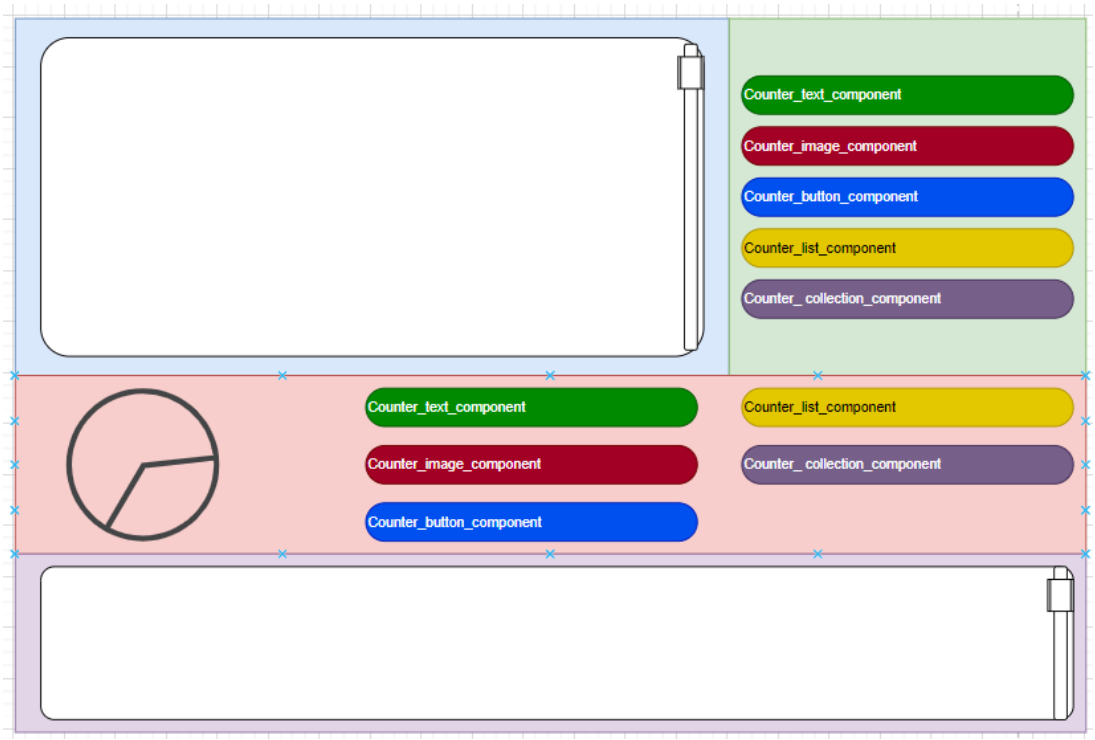


Рисунок 3.12 – ескіз зовнішнього вигляду інтерфейсу 1–4 рівнів

Складова інтерфейсу 1–4 рівнів зображена на рисунку 3.13:

```
<View>
     <View>
       <View>
         <ScrollView>
           <Text></Text>
         </ScrollView>
       </View>
     </View>

     <View>
       <View>
         <Text></Text>
       </View>
       <View>
         <Text></Text>
       </View>
       <View>
         <Text></Text>
       </View>
       <View>
         <Text></Text>
       </View>
       <View>
         <Text></Text>
       </View>
     </View>

     <View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
         <PieChart style={styles.chart} data={pieData} />
       </View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
       </View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
       </View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
       </View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
       </View>
       <View style={styles.container}>
         <Text style={styles.title}>Pie Chart Example</Text>
       </View>
     </View>

     <View>
       <View>
         <ScrollView>
           <Text></Text>
         </ScrollView>
       </View>
     </View>
   </View>
```

Рисунок 3.13 – складова інтерфейсу 1–4 рівнів

3.4 Розробка дизайну WEB–додатку

Зовнішній вигляд інтерфейсу 1 рівня зображений на рисунку 3.14:

```
const App = () => {
  const renderItem = ({ item }) => (
    <List title={item.title} image={item.image} />
  );

  return (
    <View style={styles.container}>
      <List
        data={data}
        renderItem={renderItem}
        keyExtractor={(item) => item.id}
      />
      <Button title="Click me" onPress={() => console.log('Button clicked!')} />
    </View>
  );
};
```

Рисунок 3.14 – Зовнішній вигляд інтерфейсу 1 рівня

Зовнішній вигляд інтерфейсу 2 рівня зображений на рисунку 3.15:

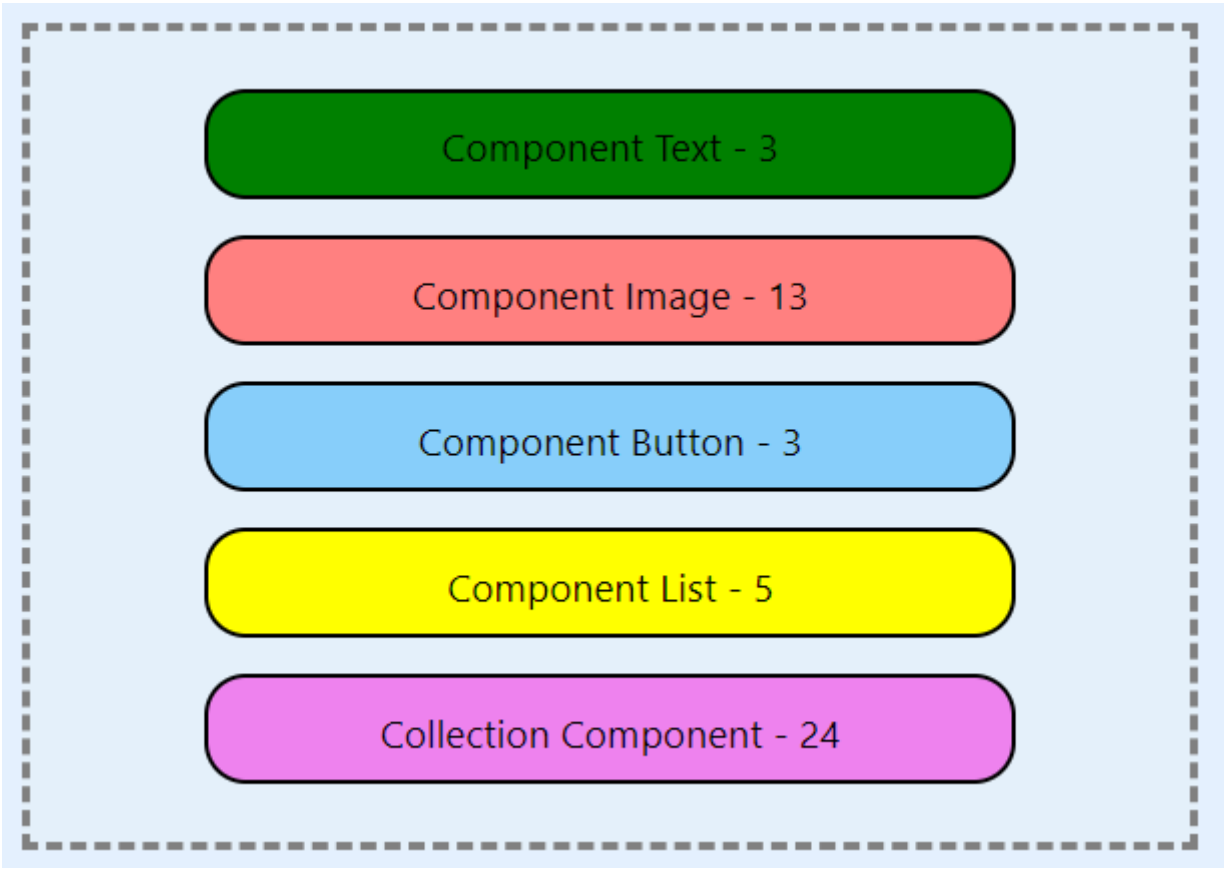


Рисунок 3.15 – Зовнішній вигляд інтерфейсу 2 рівня

Зовнішній вигляд інтерфейсу 3 рівня зображений на рисунку 3.16:

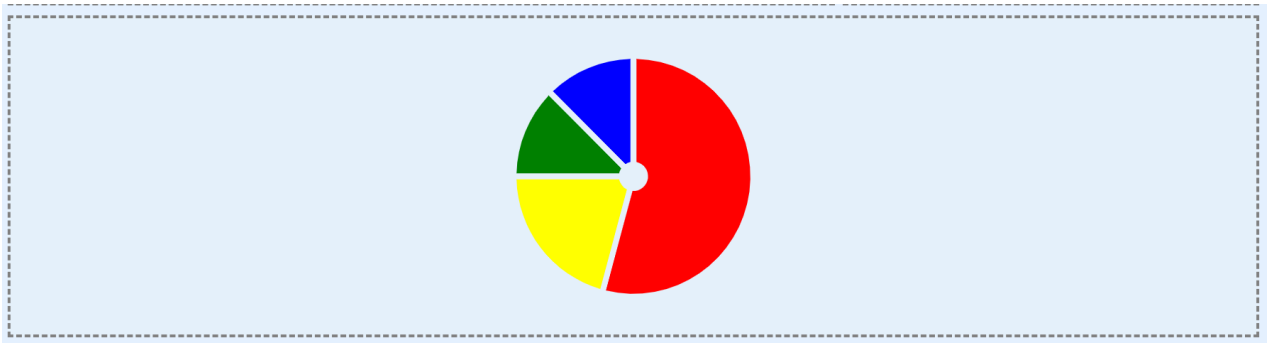


Рисунок 3.16 – Зовнішній вигляд інтерфейсу 3 рівня

Зовнішній вигляд інтерфейсу 4 рівня зображений на рисунку 3.17:

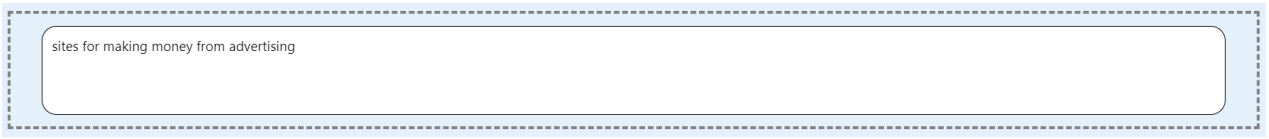


Рисунок 3.17 – Зовнішній вигляд інтерфейсу 4 рівня

Зовнішній вигляд додаткового інтерфейсу 0 рівня зображений на рисунку 3.18:

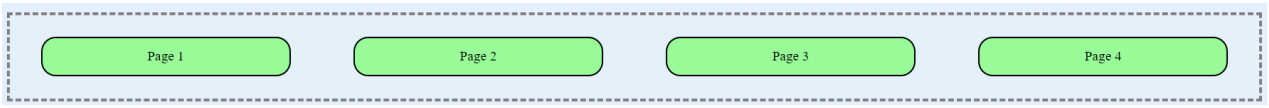


Рисунок 3.18 – Зовнішній вигляд додаткового інтерфейсу 0 рівня

Висновки до розділу 3

Результатами роботи над третім розділом мають бути визначені такі аспекти як, динаміка системи, розроблені екран, інтерфейс та дизайн WEB–сторінки.

Під час розробки WEB–сторінки, що відбувалося по приведеним вище у розділі ескізам, були додані виправлення щодо розміщення, наявності та зовнішнього вигляду компонентів, також були розроблені шаблони сторінок та приведені їхні числові коефіцієнти [12].

Розроблені функції демонструють наявний текстовий вміст реальних сторінок у призначеному для цього місці, рахують відповідні числові коефіцієнти, які відповідають заданим компонентам, та, за їх допомогою,

визнають, до яких саме шаблонів кожна з чотирьох підготовлених сторінок належить за їхнім типом інтерфейсу.

Реалізовані функціональні можливості:

– можливість вибору файлу з текстом програми під час виконання програмного коду WEB–сторінки;
– відображення секторної діаграми, зовнішній вигляд якої напряму залежить від числових коефіцієнтів налічуваних компонентів у тексті програми, який знаходиться у вибраному файлі;
– визначення типу інтерфейсу кожної сторінки за її коефіцієнтами та виведення результату у відповідному текстовому полі.

Діаграма діяльності зазнала змін, а саме видалення етапу натискання кнопки для запуску розрахунків.

Оновлена діаграма діяльності зображена на рисунку 3.19:

Діаграма діяльності до оновлення зображена на рисунку 3.20:

Рисунок 3.19 – Оновлена діаграма діяльності

Рисунок 3.20 –діаграма діяльності до оновлення

На основі розробки, при тестуванні необхідно визначити:

- коректність робото здібності компонентів;
- коректність виводу результатів аналізу;
- коректність взаємодії компонентів між собою.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

4.1 Тестування працездатності компонентів програми

Для проведення тестування робото здібності компонентів програми нам необхідно визначити інтерактивні компоненти, з якими користувач може взаємодіяти [17].

Інтерактивні компоненти:

- кнопка сторінки один;
- кнопка сторінки два;
- кнопка сторінки три;
- кнопка сторінки чотири.

При натисканні на кнопку вибору сторінки ми маємо побачити демонстрації вмісту файлу, за який і відповідає наша кнопка.

Вміст файлу 1 зображений на рисунку 4.1:

```
import React from 'react';
import { View, Text, StyleSheet, Button } from 'react-native';

const GamePage = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.heading}>Welcome to the Game Page!</Text>
      <Text>Get ready for some exciting gaming adventures!</Text>
      <View style={styles.section}>
        <Text style={styles.sectionHeading}>Popular Games</Text>
        <Text>Game 1</Text>
        <Text>Game 2</Text>
        <Text>Game 3</Text>
      </View>
      <View style={styles.section}>
        <Text style={styles.sectionHeading}>Featured Game</Text>
        <View style={styles.featuredGame}>
          <Text style={styles.featuredGameTitle}>Game Title</Text>
          <Text>Description of the featured game.</Text>
          <Button title="Play Now" onPress={() => console.log('Play Now button pressed')} />
        </View>
      </View>
    </View>
  );
};
});

export default GamePage;
```

Рисунок 4.1 – Вміст файлу 1

Відображення вмісту файлу 1 на сторінці зображене на рисунку 4.2:

```
import React from 'react';
import { View, Text, StyleSheet, Button } from 'react-native';

const GamePage = () => {
 return (
  <View style={styles.container}>
   <Text style={styles.heading}>Welcome to the Game Page!</Text>
   <Text>Get ready for some exciting gaming adventures!</Text>
   <View style={styles.section}>
    <Text style={styles.sectionHeading}>Popular Games</Text>
    <Text>Game 1</Text>
    <Text>Game 2</Text>
    <Text>Game 3</Text>
   </View>
   <View style={styles.section}>
    <Text style={styles.sectionHeading}>Featured Game</Text>
```

Рисунок 4.2 – Вміст файлу 1

Вміст файлу 2 зображений на рисунку 4.3:

```
import React from 'react';
import { View, Text, StyleSheet, Image, FlatList } from 'react-native';

const MyComponent = () => {
  const data = [
    {
      id: '1',
      title: 'Item 1',
      description: 'Description of item 1',
      imageUrl: 'https://example.com/image1.jpg',
    },
    {
      id: '2',
      title: 'Item 2',
      description: 'Description of item 2',
      imageUrl: 'https://example.com/image2.jpg',
    },
    {
      id: '3',
      title: 'Item 3',
      description: 'Description of item 3',
      imageUrl: 'https://example.com/image3.jpg',
    },
  ];

  const renderItem = ({ item }) => (
    <View style={styles.itemContainer}>
      <Image style={styles.itemImage} source={{ uri: item.imageUrl }} />
      <View style={styles.itemDetails}>
        <Text style={styles.itemTitle}>{item.title}</Text>
        <Text style={styles.itemDescription}>{item.description}</Text>
      </View>
    </View>
  );

  return (
    <View style={styles.container}>
      <Text style={styles.heading}>My Component</Text>
      <Image style={styles.mainImage} source={{ uri: 'https://example.com/main-image.jpg' }} />
      <List
```

Рисунок 4.3 – Вміст файлу 2

Відображення вмісту файлу 2 на сторінці зображене на рисунку 4.4:

```
import React from 'react';
import { View, Text, StyleSheet, Image, FlatList } from 'react-native';

const MyComponent = () => {
 const data = [
   {
    id: '1',
    title: 'Item 1',
    description: 'Description of item 1',
    imageUrl: 'https://example.com/image1.jpg',
   },
   {
    id: '2',
    title: 'Item 2',
    description: 'Description of item 2',
    imageUrl: 'https://example.com/image2.jpg',
```

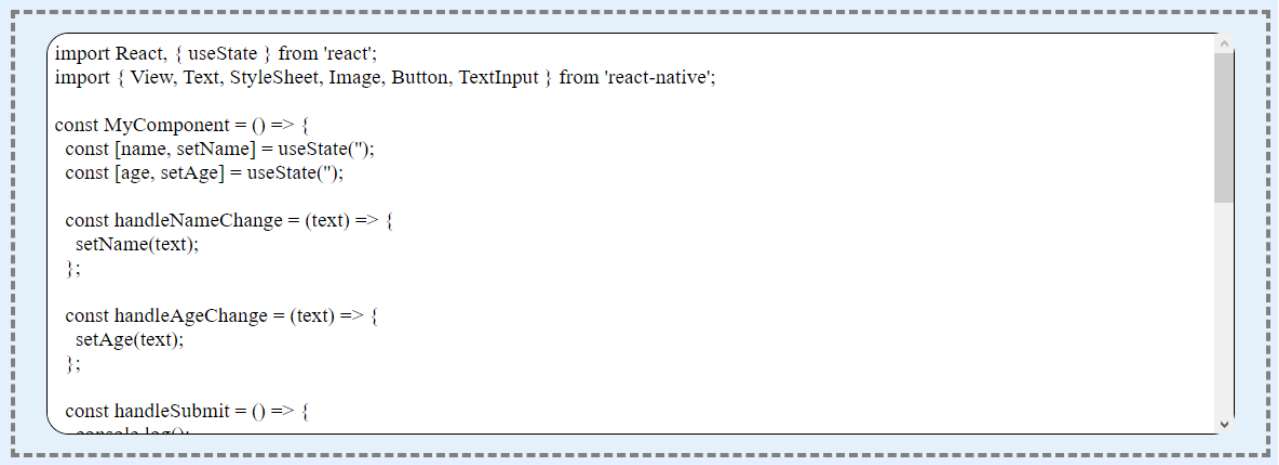
Рисунок 4.4 – Вміст файлу 2

Вміст файлу 3 зображений на рисунку 4.5:

```
import React, { useState } from 'react';
import { View, Text, StyleSheet, Image, Button, TextInput } from 'react-native';

const MyComponent = () => {
  const [name, setName] = useState('');
  const [age, setAge] = useState('');

  const handleNameChange = (text) => {
    setName(text);
  };

  const handleAgeChange = (text) => {
    setAge(text);
  };

  const handleSubmit = () => {
    console.log();
  };

  return (
    <View style={styles.container}>
      <Text style={styles.heading}>My Component</Text>
      <Image style={styles.image} source={require('./image.jpg')} />
      <TextInput
        style={styles.input}
        placeholder="Enter your name"
        value={name}
        onChangeText={handleNameChange}
      />
      <TextInput
        style={styles.input}
        placeholder="Enter your age"
        value={age}
        onChangeText={handleAgeChange}
      />
      <Button title="Submit" onPress={handleSubmit} />
    </View>
  );
};
export default MyComponent;
```

Рисунок 4.5 – Вміст файлу 3

Відображення вмісту файлу 3 на сторінці зображене на рисунку 4.6:



```
import React, { useState } from 'react';
import { View, Text, StyleSheet, Image, Button, TextInput } from 'react-native';

const MyComponent = () => {
  const [name, setName] = useState('');
  const [age, setAge] = useState('');

  const handleNameChange = (text) => {
    setName(text);
  };

  const handleAgeChange = (text) => {
    setAge(text);
  };

  const handleSubmit = () => {
```

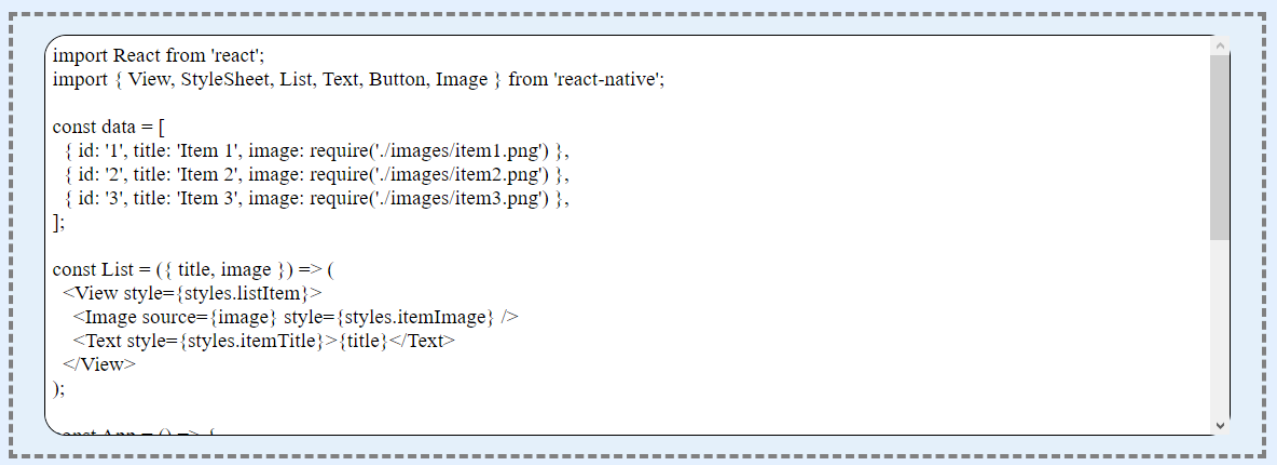
Рисунок 4.6 – Відображення вмісту файлу 3 на сторінці

Вміст файлу 4 зображений на рисунку 4.7:

```
import React from 'react';
import { View, StyleSheet, List, Text, Button, Image } from 'react-native';

const data = [
  { id: '1', title: 'Item 1', image: require('./images/item1.png') },
  { id: '2', title: 'Item 2', image: require('./images/item2.png') },
  { id: '3', title: 'Item 3', image: require('./images/item3.png') },
];

const List = ({ title, image }) => (
  <View style={styles.listItem}>
    <Image source={image} style={styles.itemImage} />
    <Text style={styles.itemTitle}>{title}</Text>
  </View>
);

const App = () => {
  const renderItem = ({ item }) => (
    <List title={item.title} image={item.image} />
  );

  return (
    <View style={styles.container}>
      <List
        data={data}
        renderItem={renderItem}
        keyExtractor={(item) => item.id}
      />
      <Button title="Click me" onPress={() => console.log('Button clicked!')} />
    </View>
  );
};
```

Рисунок 4.7 – Вміст файлу 4

Відображення вмісту файлу 4 на сторінці зображене на рисунку 4.8:



```
import React from 'react';
import { View, StyleSheet, List, Text, Button, Image } from 'react-native';

const data = [
  { id: '1', title: 'Item 1', image: require('./images/item1.png') },
  { id: '2', title: 'Item 2', image: require('./images/item2.png') },
  { id: '3', title: 'Item 3', image: require('./images/item3.png') },
];

const List = ({ title, image }) => (
  <View style={styles.listItem}>
    <Image source={image} style={styles.itemImage} />
    <Text style={styles.itemTitle}>{title}</Text>
  </View>
);
```

Рисунок 4.8 – Відображення вмісту файлу 4 на сторінці

Продивившись вміст чотирьох файлів та пройшовшись по всім чотирьом кнопкам ми можемо побачити, що текст успішно відображається у відповідних текстових полях [18].

4.2 Тестування коректності виводу результатів аналізу

Для перевірки коректності виводу результатів аналізу нам необхідно вручну перерахувати кількість шуканих нами компонентів та порівняти їх з порахованими коефіцієнтами.

Для перевірки виберемо вміст файлу під номером 1. Для пошуку відповідних програмних конструкцій будемо використовувати функціонал програми Word – навігація по документу [19].

Пошук компоненту з назвою Text зображено на рисунку 4.9:

Пошук компоненту з назвою Image зображено на рисунку 4.10:

Пошук компоненту з назвою Button зображено на рисунку 4.11:

Пошук компоненту з назвою List зображено на рисунку 4.12:

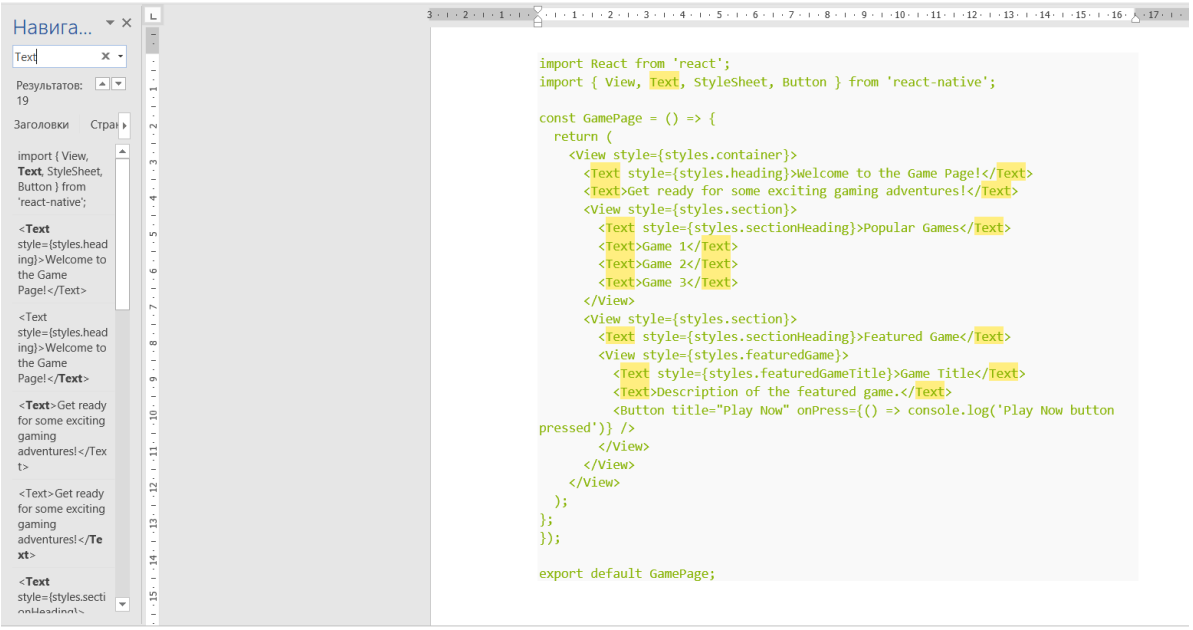


Рисунок 4.9 – Пошук компоненту з назвою Text

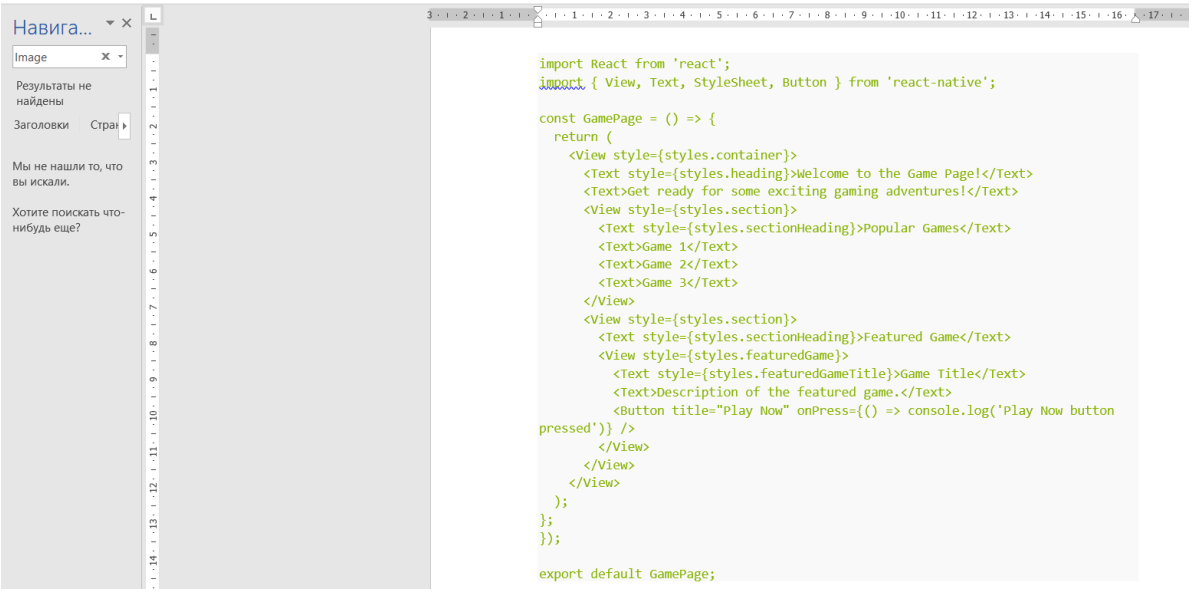


Рисунок 4.10 – Пошук компоненту з назвою Image

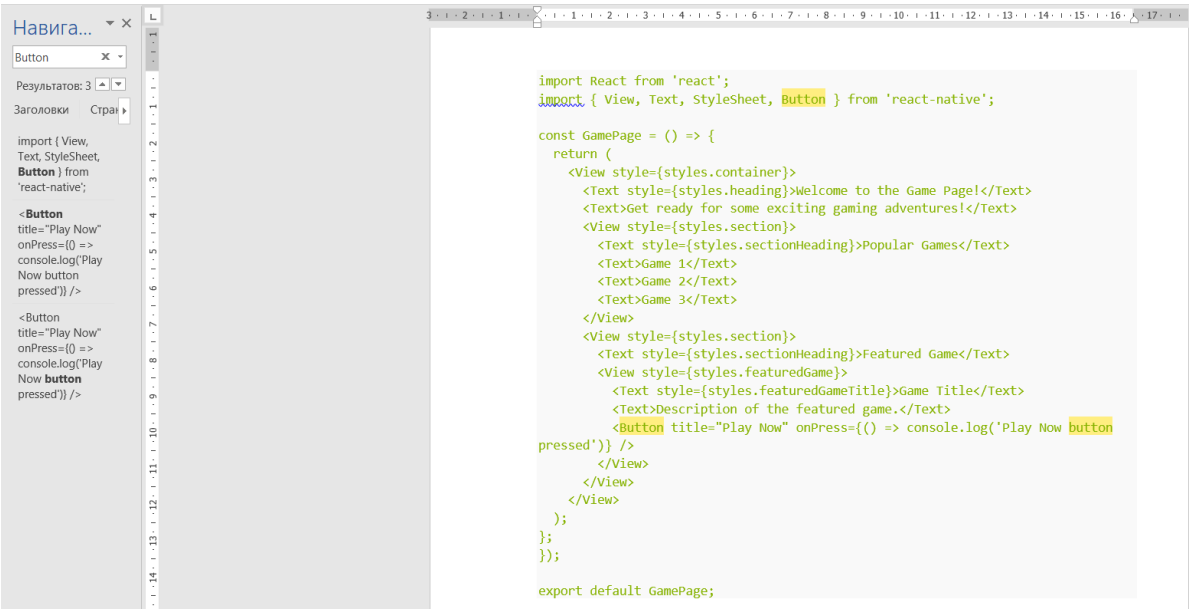


Рисунок 4.11 – Пошук компоненту з назвою Button

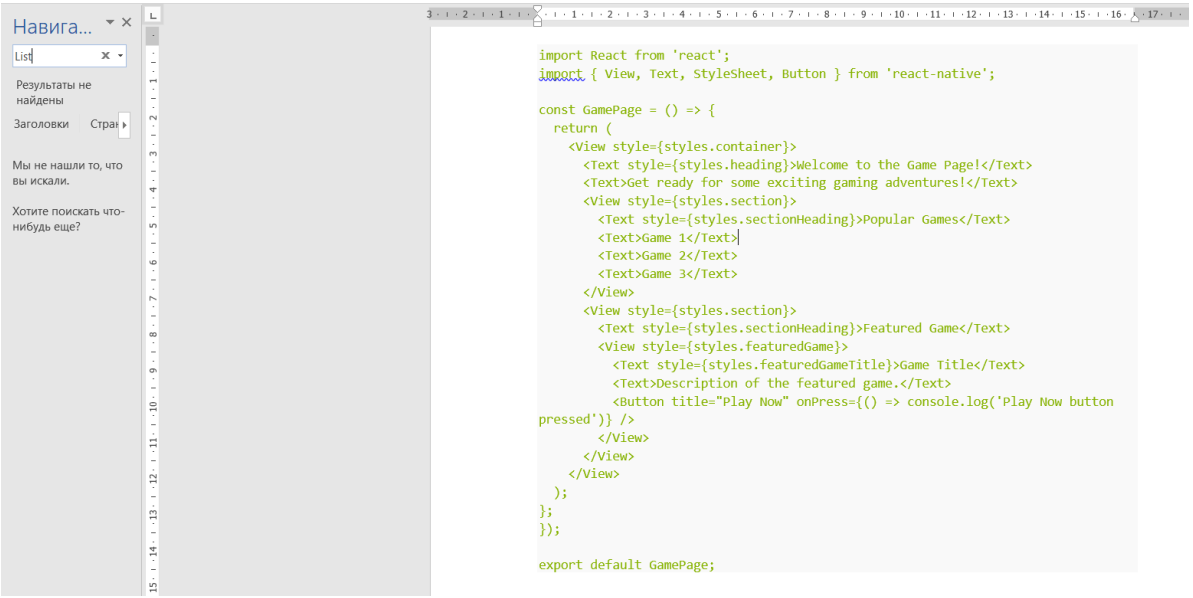


Рисунок 4.12 – Пошук компоненту з назвою List

Пошук компонентів програмою зображено на рисунку 4.13:

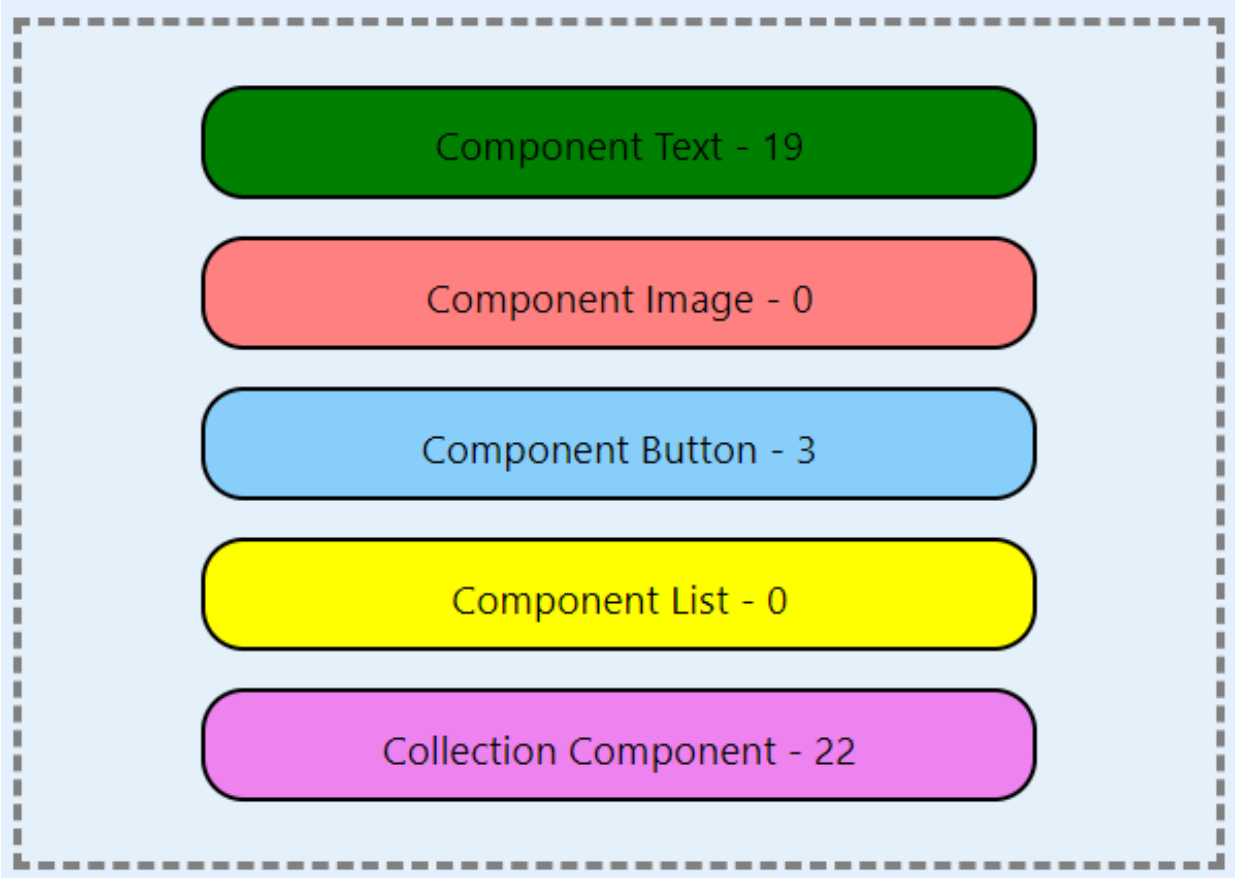


Рисунок 4.13 – Пошук компоненту з назвою List

По рисунках 4.9–4.13 видно, що програма рахує кількість компонентів і коефіцієнти вірно [20].

4.3 Тестування коректності взаємодії компонентів між собою

Для перевірки коректності взаємодії компонентів між собою слід звернути увагу на їхню діаграму діяльності. Взаємодія компонентів складається таким чином:

- вивантаження вмісту файлу;
- розрахунок;
- отримання результатів.

Слід зауважити, що отримання результатів у нас відбувається двома способами:

- виведення типу інтерфейсу сайту;

– виведення діаграми коефіцієнтів.

Відображення коректності взаємодії між собою зображено на рисунках 4.14–4.15:

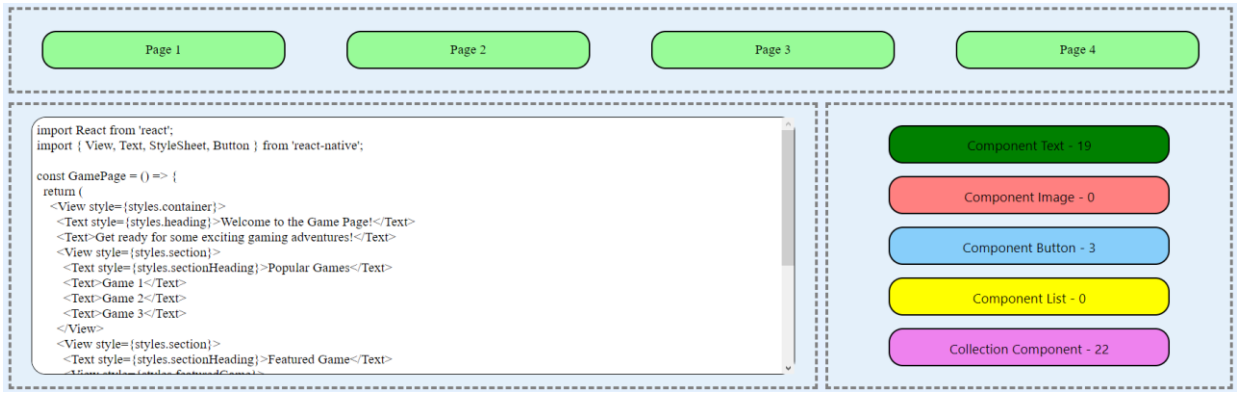


Рисунок 4.14 – Відображення коректності взаємодії між собою

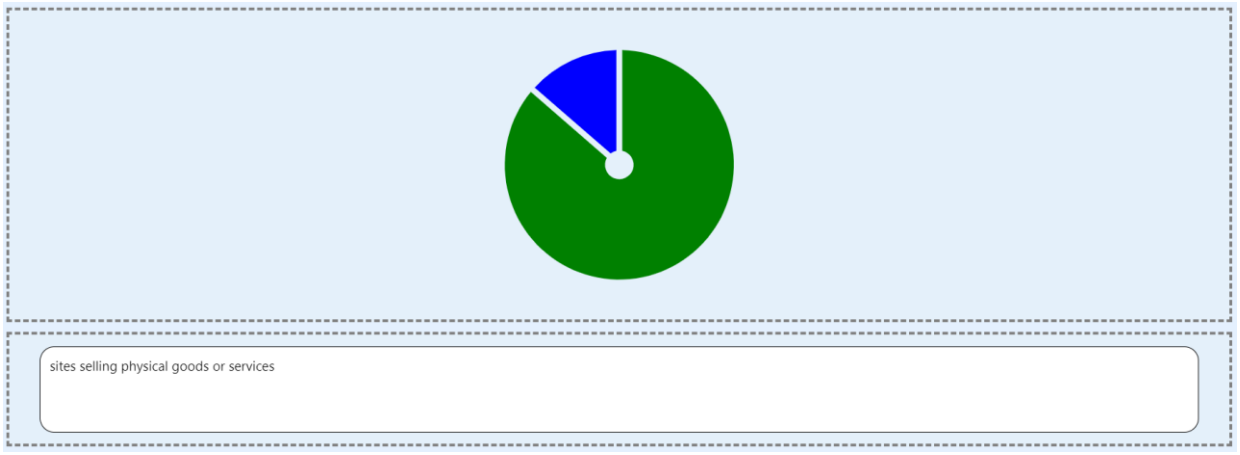


Рисунок 4.15 – Відображення коректності взаємодії між собою

Пройшовши весь шлях діаграми діяльності, ми впевнилися в тому, що компоненти взаємодіяють між собою та демонструють інформацію по вмісту одного файлу.

Висновки до розділу 4

Результатами роботи над четвертим розділом визначені такі аспекти як коректність роботоздібності компонентів, коректність виводу результатів аналізу, коректність взаємодії компонентів між собою. Всі компоненти та результати працюють коректно та видають необхідні результати.

Загальні висновки

На першому розділі визначені різновиди WEB–сторінок, які поділяються за своїм напрямом, та за цими напрямами вибрані необхідні нам критерії оцінки інтерфейсу для подальшого аналізу та висунення припущення, щодо його типу.

Під час опису різновидів сайтів проаналізовані різновиди типів інтерфейсів сайтів, їхні підтипи та особливості, які допоможуть нам у визначенні коефіцієнтів та їх підрахунках задля визначення типів інтерфейсів. Були виділені необхідні критерії оцінки, опираючись на які ми зможемо продовжувати проектування нашого WEB–додатку поповнюючи список необхідних функціональних вимог для покращення аналізу та виконання розрахунків.

На другому розділі визначені та спроектовані задачі для подальшої розробки WEB–сторінки, сформовані функціональні вимоги, вхідні та вихідні дані проекту, що враховують в себе, створені нами, шаблони типів інтерфейсів, вибрана мова та підхід до програмування, описані екрани з інтерфейсом та дизайном. Описані лічильники, за допомогою яких будуть обчислюватися наші коефіцієнти, визначені компоненти, з яких буде складатися наша WEB–сторінки, представлені їх ескізи та оформлення станів.

Результатами роботи над третім розділом розроблені такі аспекти, як динаміка системи, розроблені екран, інтерфейс та дизайн WEB–сторінки.

Під час розробки WEB–сторінки, що відбувалося по приведеним вище у розділі ескізам, були додані виправлення щодо розміщення, наявності та зовнішнього вигляду компонентів, також були розроблені шаблони сторінок та приведені їхні числові коефіцієнти. Змінили діаграму діяльності.

Діаграма діяльності зазнала змін, а саме видалення етапу натискання кнопки для запуску розрахунків.

На четвертому розділі визначені такі аспекти як коректність роботоздібності компонентів, коректність виводу результатів аналізу, коректність взаємодії компонентів між собою. Всі компоненти та результати працюють коректно та видають необхідні результати.

Список використаних джерел

1. Горячкін, В. М. Навчальний посібник для підготовки кваліфікаційної роботи на здобуття ОС Бакалавр [Текст] / В. М. Горячкін, О. В. Горбова, О. С. Куроп’ятник. – Дніпро: Український державний університет науки і технологій, 2022. – 137с.

2. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп’ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с.

3. Майерс, Г., Баджетт, Т., Сандлер К. Искусство тестирования программ, 3-е изд.: пер. с англ. – М.: ООО “И.Д. Вильямс”, 2012. –272 с.

4. Інженерія програмного забезпечення [Текст]: навчальний посібник / В. І. Шинкаренко, О. В. Горбова, О. П. Іванов, В. О. Андрющенко, В. Я. Нечай; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Дніпро, 2019. – 140 с.

5. Програмне забезпечення систем [Текст]: методичні вказівки до виконання дипломного проекту / уклад. В. В. Скалозуб, В. І. Шинкаренко, В. О. Андрющенко, Ю. М. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2013. – 41 с.

6. Web-design [Електронний ресурс] / Режим доступу: URL: https://en.wikipedia.org/wiki/Web_design/. – Дата звернення: 02 травня 2023.

7. How To Analyze Your Website's Content [Електронний ресурс] / Режим доступу: URL: https://www.demandjump.com/blog/how-to-analyze-your-websites-content-a-guide-for-content-audits/. – Дата звернення: 05 травня 2023.

8. Види веб сайтів, класифікація, типи та їх призначення [Електронний ресурс] / Режим доступу: URL: https://webtune.com.ua/statti/web-rozrobka/vydy-sajtiv-ta-yih-funkczional/. – Дата звернення: 06 травня 2023.

9. What is Web Design? The Ultimate Guide To Website Design [Електронний ресурс] / Режим доступу: URL: https://www.pagecloud.com/blog/web-design-guide/. – Дата звернення: 06 травня 2023.

10. React Native · Learn once, write anywhere [Електронний ресурс] / Режим доступу: URL: https://reactnative.dev/. – Дата звернення: 13 травня 2023.

11. Айзенман, Б. Learning React Native [Текст] / Айзенман Бонні. – O'Reilly Media, 2017. – 240 с.

12. Лебенсолд, Д. React Native Cookbook: Bringing the Web to Native Platforms [Текст] / Лебенсолд Джонатан. –O'Reilly Media, 2018. – 176 с.

13. JavaScript [Електронний ресурс] / Режим доступу: URL: https://en.wikipedia.org/wiki/JavaScript/. – Дата звернення: 13 травня 2023.

14. WebStorm [Електронний ресурс] / Режим доступу: URL: https://uk.wikipedia.org/wiki/WebStorm/. – Дата звернення: 14 травня 2023.

15. React Native | WebStorm Documentation - JetBrains [Електронний ресурс] / Режим доступу: URL: https://www.jetbrains.com/help/webstorm/react-native.html/. – Дата звернення: 14 травня 2023.

16. React Native Tutorial [Електронний ресурс] / Режим доступу: URL: https://www.tutorialspoint.com/react_native/index.htm/. – Дата звернення: 15 травня 2023.

17. Як тестувати веб-сайт: основні етапи і поради [Електронний ресурс] / Режим доступу: URL: https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi/. – Дата звернення: 20 травня 2023.

18. Повне керівництво по тестуванню веб-додатків [Електронний ресурс] / Режим доступу: URL: https://uk.myservername.com/web-application-testing-complete-guide/. – Дата звернення: 20 травня 2023.

19. Тестування веб-проєктів: основні етапи та поради [Електронний ресурс] / Режим доступу: URL: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/. – Дата звернення: 20 травня 2023.

20. Testing React Native Apps [Електронний ресурс] / Режим доступу: URL: https://jestjs.io/docs/tutorial-react-native. – Дата звернення: 20 травня 2023.

Додатки

Додаток А

Розробка WEB додатку для порівняння типів інтерфейсу сайтів

Текст програми

_____

Листів __

2023

АНОТАЦІЯ

Документ _____ «Розробка WEB додатку для порівняння типів інтерфейсу сайтів. Текст програми» входить до складу програмної документації на програму, що виконує роль додатку для аналізування коефіцієнтів, які формуються на основі кількості комонентів.

У даному документі представлений текст програм. Програми написані на мові JS з використанням фреймворку Reacty програмному середовищі WebStorm 2023.

ЗМІСТ

1. СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛІВ

Схема взаємодії програмних подулів зображена на рисунку 1.1

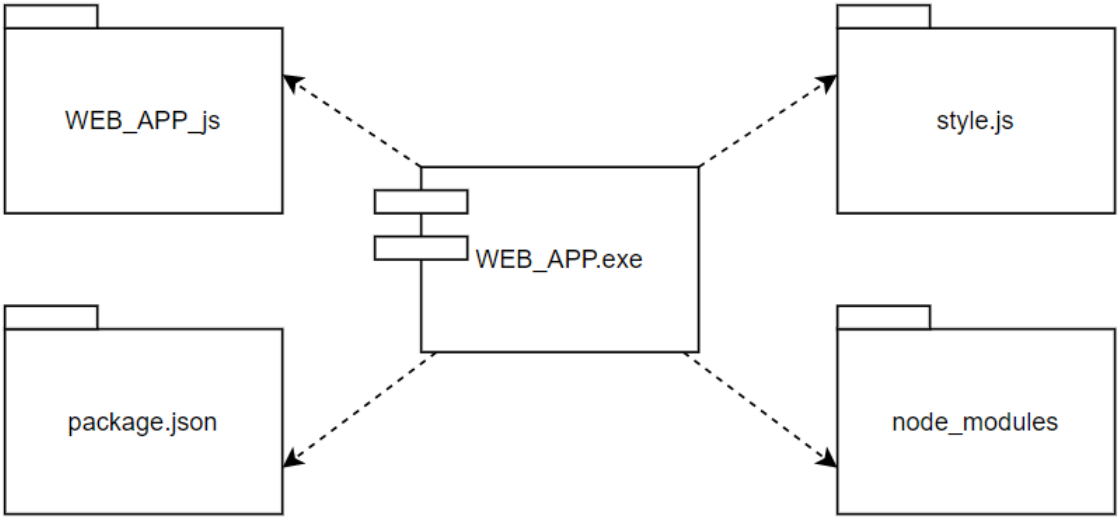


Рисунок 1.1 – Схема взаємодії модулів

2. ТЕКСТ ПРОГРАМИ

2.1. Текст програми файлу WEB_APP.js

```
import * as React from 'react';
import { useState } from 'react';
import MyCustomFont from './1.ttf';
import {
  View,
  Text,
  StyleSheet,
  ScrollView,
  TouchableHighlight,
} from 'react-native';
import { PieChart } from 'react-
native-svg-charts';

const countOccurrences = (text,
searchString) => {
  const regex = new
RegExp(searchString, 'gi');
  const matches = text.match(regex);
  return matches ? matches.length : 0;
};

const PieChartExample = () => {
  const [count, setCount] =
useState('');
  const text1 =
    `import React from 'react';
import { View, Text, StyleSheet,
Button } from 'react-native';

const GamePage = () => {
  return (
    <View style={styles.container}>
      <Text
style={styles.heading}>Welcome to the
Game Page!</Text>
      <Text>Get ready for some
exciting gaming adventures!</Text>
      <View style={styles.section}>
        <Text
style={styles.sectionHeading}>Popular
Games</Text>
        <Text>Game 1</Text>
        <Text>Game 2</Text>
        <Text>Game 3</Text>
      </View>
      <View style={styles.section}>
        <Text
style={styles.sectionHeading}>Featured
Game</Text>
        <View
style={styles.featuredGame}>
          <Text
style={styles.featuredGameTitle}>Game
Title</Text>
          <Text>Description of the
featured game.</Text>
          <Button title="Play Now"
onPress={() => console.log('Play Now
button pressed')} />
        </View>
      </View>
    </View>
  );
};
});

export default GamePage;`;
  const searchString1 = 'Text';
  const count1 =
countOccurrences(count,
searchString1);

  const text2 =
    `import React from 'react';
import { View, Text, StyleSheet,
Image, FlatList } from 'react-native';

const MyComponent = () => {
  const data = [
    {
      id: '1',
      title: 'Item 1',
      description: 'Description of
item 1',
      imageUrl:
'https://example.com/image1.jpg',
    },
    {
      id: '2',
      title: 'Item 2',
      description: 'Description of
item 2',
```

```
      imageUrl:
'https://example.com/image2.jpg',
    },
    {
      id: '3',
      title: 'Item 3',
      description: 'Description of
item 3',
      imageUrl:
'https://example.com/image3.jpg',
    },
  ];

  const renderItem = ({ item }) => (
    <View
style={styles.itemContainer}>
      <Image style={styles.itemImage}
source={{ uri: item.imageUrl }} />
      <View
style={styles.itemDetails}>
        <Text
style={styles.itemTitle}>{item.title}<
/Text>
        <Text
style={styles.itemDescription}>{item.d
escription}</Text>
      </View>
    </View>
  );

  return (
    <View style={styles.container}>
      <Text style={styles.heading}>My
Component</Text>
      <Image style={styles.mainImage}
source={{ uri:
'https://example.com/main-image.jpg'
}} />
      <List
        data={data}
        renderItem={renderItem}
        keyExtractor={(item) =>
item.id}
      />
    </View>
  );
};
export default MyComponent;
`;
  const searchString2 = 'Image';
  const count2 =
countOccurrences(count,
searchString2);

  const text3 =
    `import React, { useState } from
'react';
import { View, Text, StyleSheet,
Image, Button, TextInput } from
'react-native';

const MyComponent = () => {
  const [name, setName] =
useState('');
  const [age, setAge] = useState('');

  const handleNameChange = (text) => {
    setName(text);
  };

  const handleAgeChange = (text) => {
    setAge(text);
  };

  const handleSubmit = () => {
    console.log();
  };

  return (
    <View style={styles.container}>
      <Text style={styles.heading}>My
Component</Text>
      <Image style={styles.image}
source={require('./image.jpg')} />
      <TextInput
        style={styles.input}
        placeholder="Enter your name"
        value={name}
        onChangeText={handleNameChange
}
      />
      <TextInput
        style={styles.input}
        placeholder="Enter your age"
        value={age}
        onChangeText={handleAgeChange}
      />
      <Button title="Submit"
onPress={handleSubmit} />
    </View>
```

```
  );
};
export default MyComponent;`;
  const searchString3 = 'Button';
  const count3 =
countOccurrences(count,
searchString3);

  const text4 =
    `import React from 'react';
import { View, StyleSheet, List, Text,
Button, Image } from 'react-native';

const data = [
  { id: '1', title: 'Item 1', image:
require('./images/item1.png') },
  { id: '2', title: 'Item 2', image:
require('./images/item2.png') },
  { id: '3', title: 'Item 3', image:
require('./images/item3.png') },
];

const List = ({ title, image }) => (
  <View style={styles.listItem}>
    <Image source={image}
style={styles.itemImage} />
    <Text
style={styles.itemTitle}>{title}</Text
>
  </View>
);

const App = () => {
  const renderItem = ({ item }) => (
    <List title={item.title}
image={item.image} />
  );

  return (
    <View style={styles.container}>
      <List
        data={data}
        renderItem={renderItem}
        keyExtractor={(item) =>
item.id}
      />
      <Button title="Click me"
onPress={() => console.log('Button
clicked!')} />
    </View>
```

```
  );
};`;
  const searchString4 = 'List';
  const count4 =
countOccurrences(count,
searchString4);
  const count5 = count1 + count2 +
count3 + count4;
  const data = [
    {
      key: 1,
      amount: count1,
      svg: { fill: 'Green' },
      label: 'Green',
    },
    {
      key: 2,
      amount: count2,
      svg: { fill: 'Red' },
      label: 'Red',
    },
    {
      key: 3,
      amount: count3,
      svg: { fill: 'Blue' },
      label: 'Blue',
    },
    {
      key: 4,
      amount: count4,
      svg: { fill: 'Yellow' },
      label: 'Yellow',
    },
  ];

  const pieData = data.map((item) =>
({
    value: item.amount,
    svg: item.svg,
    key: item.key,

    arc: { outerRadius: '80%',
innerRadius: '10%' }, // Зовнішній та
внутрішній радіуси
  }));

  const data1 = [
    {
      key: 'sites selling physical
goods or services',
```

```
      key_mas: [19, 0, 3, 0],
    },
    {
      key: 'sites for business
digitization',
      key_mas: [7, 13, 0, 2],
    },
    {
      key: 'sites for receiving
commissions from transactions',
      key_mas: [12, 4, 2, 0],
    },
    {
      key: 'sites for selling paid
online services',
      key_mas: [13, 4, 2, 0],
    },
    {
      key: 'sites for making money
from advertising',
      key_mas: [3, 13, 3, 5],
    },
    {
      key: '',
      key_mas: [count1, count2,
count3, count4],
    },
  ];



  function result() {
    for (var i = 0; i < 6; i++) {
      if (
        data1[i].key_mas.every(
          (element, index) => element
== data1[5].key_mas[index]
        )
      ) {
        return data1[i].key;
      }
    }
  }


  return (
    <View style={styles.container}>
      <ScrollView
style={styles.container_scroll}>
```

```
        <View
style={styles.container_third_2_box}>
          <View
style={styles.text_container_2}>
            <TouchableHighlight
              style={{ width: '100%',
height: '100%' }}
              activeOpacity={0.6}
              underlayColor="#DDDDDD"
              onPress={() =>
setCount(text1)}>
              <Text
style={styles.buttonText}>Page
1</Text>
            </TouchableHighlight>
          </View>
          <View
style={styles.text_container_2}>
            <TouchableHighlight
              style={{ width: '100%',
height: '100%' }}
              activeOpacity={0.6}
              underlayColor="#DDDDDD"
              onPress={() =>
setCount(text2)}>
              <Text
style={styles.buttonText}>Page
2</Text>
            </TouchableHighlight>
          </View>
          <View
style={styles.text_container_2}>
            <TouchableHighlight
              style={{ width: '100%',
height: '100%' }}
              activeOpacity={0.6}
              underlayColor="#DDDDDD"
              onPress={() =>
setCount(text3)}>
              <Text
style={styles.buttonText}>Page
3</Text>
            </TouchableHighlight>
          </View>
          <View
style={styles.text_container_2}>
            <TouchableHighlight
              style={{ width: '100%',
height: '100%' }}
              activeOpacity={0.6}
```

```
              underlayColor="#DDDDDD"
              onPress={() =>
setCount(text4)}>
              <Text
style={styles.buttonText}>Page
4</Text>
            </TouchableHighlight>
          </View>
        </View>
        <View
style={styles.container_bt_fisrt_secon
d_box}>
          <View
style={styles.container_fisrt_box}>
            <ScrollView
style={styles.container_scroll_s}>
              <Text style={{ padding:
5, fontFamily: 'MontserratBold',
}}>{count}</Text>
            </ScrollView>
          </View>
          <View
style={styles.container_bw_box}>
            <Text></Text>
          </View>
          <View
style={styles.container_second_box}>
          <View
style={styles.container_bw_box}>
            <Text></Text>
          </View>
            <View
style={styles.text_container_green}>
              <Text>Component Text -
{count1}</Text>
            </View>
            <View
style={styles.text_container_red}>
              <Text>Component Image -
{count2}</Text>
            </View>
            <View
style={styles.text_container_blue}>
              <Text>Component Button -
{count3}</Text>
            </View>
            <View
style={styles.text_container_yellow}>
              <Text>Component List -
{count4}</Text>
            </View>
            <View
style={styles.text_container_violet}>
              <Text>Collection
Component - {count5}</Text>
            </View>
            <View
style={styles.container_bw_box}>
            <Text></Text>
          </View>
          </View>
        </View>
        <View
style={styles.container_bw_box}>
          <Text></Text>
        </View>
        <View
style={styles.container_third_box}>

          <View
style={styles.container_sec}>
            <PieChart
style={styles.chart} data={pieData} />
          </View>
        </View>
        <View
style={styles.container_fourth_box}>
          <ScrollView
style={styles.container_scroll_s}>
            <Text style={{ padding: 10
}}>{result()}</Text>
          </ScrollView>
        </View>
      </ScrollView>
    </View>
  );
};
```

2.2. Текст програми файлу style.js

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    width: '100%',
    height: 1000,
    alignItems: 'center',
    justifyContent: 'center',
```

```
},
buttonText: {
  fontFamily: 'MontserratBold',
  paddingTop: 10,
  textAlign: 'center',
},
container_sec: {
  flex: 1,
  height: '100%',
  width: '100%',
  alignItems: 'center',
  justifyContent: 'center',
  borderStyle: 'dashed',
  borderWidth: 3,
  borderColor: 'gray',
},
container_scroll: {
  width: '100%',
  height: '100%',
  padding: 50,
  backgroundColor: '#E4F0FE',
  justifyContent: 'space-around',
  flexDirection: 'column',
},

container_scroll_s: {
  borderWidth: 1,
  borderRadius: 15,
  marginVertical: 12,
  width: '95%',
  height: '100%',
  backgroundColor: 'white',
},

container_bw_box: {
  flex: 0.01,
},

text_container: {
  overflow: 'hidden',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: '#98FB98',
},
text_container_2: {
  overflow: 'hidden',
  borderWidth: 2,
  borderRadius: 15,
  width: '20%',
  height: 40,
  backgroundColor: '#98FB98',
},
text_container_green: {
  paddingTop: 10,
  textAlign: 'center',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: 'green',
},
text_container_red: {
  paddingTop: 10,
  textAlign: 'center',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: '#FF8080',
},
text_container_blue: {
  paddingTop: 10,
  textAlign: 'center',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: '#87CEFA',
},
text_container_yellow: {
  paddingTop: 10,
  textAlign: 'center',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: 'yellow',
},
text_container_violet: {
  paddingTop: 10,
  textAlign: 'center',
  borderWidth: 2,
  borderRadius: 15,
  width: '70%',
  height: 40,
  backgroundColor: 'violet',
},
container_fisrt_box: {
```

```
    alignItems: 'center',
    justifyContent: 'center',
    borderStyle: 'dashed',
    backgroundColor: '#E4F0FA',
    flex: 1,
    height: '100%',
    borderWidth: 3,
    borderColor: 'gray',
  },
  container_third_1_box: {
    alignItems: 'center',
    backgroundColor: '#E4F0FA',
    flex: 1,
    height: '100%',
    justifyContent: 'space-around',
    flexDirection: 'column',
    borderStyle: 'dashed',
    borderWidth: 3,
    borderColor: 'gray',
  },

  container_third_2_box: {
    alignItems: 'center',
    backgroundColor: '#E4F0FA',
    height: 90,
    justifyContent: 'space-around',
    flexDirection: 'row',
    borderStyle: 'dashed',
    borderWidth: 3,
    borderColor: 'gray',
  },

  container_second_box: {
    alignItems: 'center',
    borderStyle: 'dashed',
    backgroundColor: '#E4F0FA',
    flex: 0.5,
    height: '100%',
    borderWidth: 3,
    borderColor: 'gray',
    justifyContent: 'space-around',
    flexDirection: 'column',
  },
  container_bt_fisrt_second_box: {
    width: '100%',
    height: 300,
    marginVertical: 10,
    flexDirection: 'row',
    justifyContent: 'space-around',
  },
  container_third_box: {
    alignItems: 'center',
    width: '100%',
    height: 330,
    justifyContent: 'space-around',
    flexDirection: 'row',
    backgroundColor: '#E4F0FA',
  },

  container_fourth_box: {
    marginVertical: 10,
    alignItems: 'center',
    justifyContent: 'center',
    width: '100%',
    height: 120,
    borderStyle: 'dashed',
    borderWidth: 3,
    borderColor: 'gray',
    backgroundColor: '#E4F0FA',
  },
  chart: {
    width: 300,
    height: 300,
  },
});
```

2.3. Текст програми файлу package.json

```
{
  "dependencies": {
    "react-native-fs": "*",
    "expo-file-system": "~15.1.1",
    "react-native-svg": "13.4.0",
    "react-native-svg-charts": "*",
    "expo-font": "~11.0.1",
    "expo-app-loading": "~2.1.1"
```