НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних систем, мереж та кібербезпеки

Касаткін Д.Ю., к.пед.н., доц.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розроблення комп'ютерної системи підтримки прийняття рішень в умовах невизначеності»

Спеціальність 123 «Комп’ютерна інженерія»

Гарант освітньої програми

к.фіз.-мат.н., доц. _____ / Нікітенко Є.В. /
(підпис) (ПІБ)

Керівник дипломного проекту: _____ / Шкарупило В.В. /
(підпис) (ПІБ)

Виконав: _____ / Петренко Б.В. /
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
_____ / Касаткін Д.Ю., к.пед.н., доц. /
(підпис) (ПІБ, вчене звання і ступінь)
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ СТУДЕНТУ

Петренко Богдану Валерійовичу
(прізвище, ім'я, по батькові)

Спеціальність (напрям підготовки): комп'ютерна інженерія

Тема бакалаврської кваліфікаційної роботи: «Розроблення комп'ютерної системи підтримки прийняття рішень в умовах невизначеності»

Термін подання завершеної роботи на кафедру _____

Вихідні дані до бакалаврської кваліфікаційної роботи вимоги до апаратного і програмного забезпечення: Android 7.0+, 4-ядерний процесор, 2ГБ оперативної пам'яті.

Перелік питань, що підлягають розробці:
1. Аналіз існуючих методів прийняття рішень за невизначеності
2. Вибір середовища розробки та технічних засобів
3. Розробка програмного забезпечення

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "_____" _____ 2024 р.

Керівник кваліфікаційної роботи _____ Шкарупило В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Петренко Б.В.
(підпис) (прізвище та ініціали студента)

Н У Б і П

У к

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	07.02.2025 р. – 18.02.2025 р.	Виконано
2	Проектування комп'ютерної системи	05.03.2025 р. – 25.03.2025 р.	Виконано
3	Реалізація комп'ютерної системи	15.04.2025 р. – 13.05.2025 р.	Виконано
4	Тестування комп'ютерної системи	10.05.2025 р. – 15.05.2025 р.	Виконано
5	Оформлення пояснювальної записки	16.05.2025 р. – 20.05.2025 р.	Виконано

Студент _____ **Б.В. Петренко**

(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **В.В. Шкарупило**

(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 79 сторінок, 27 рисунки, 2 таблиці, 6 лістингів, 20 джерел.

ПІДТРИМКА ПРИЙНЯТТЯ РІШЕНЬ, ANDROID, JAVA, МОБІЛЬНИЙ ДОДАТОК, ЗВАЖЕНА СУМА, ІНТЕРФЕЙС, КРИТЕРІЇ, ВАРІАНТИ, АЛГОРИТМ.

Об’єктом дослідження є процес прийняття рішень в умовах невизначеності.

Метою роботи є розроблення комп’ютерної системи підтримки прийняття рішень в умовах невизначеності у вигляді Android-додатку, що реалізовує метод зваженої суми для вибору оптимального варіанту.

Проект складається з 4 розділів.

У першому розділі проведено аналіз процесу прийняття рішень в умовах невизначеності, розглянуто існуючі методи та обґрунтовано доцільність використання методу зважених оцінок.

У другому розділі здійснено проєктування комп’ютерної системи: обрано середовище розробки, мову програмування, структуру даних та інтерфейс користувача.

У третьому розділі описано процес реалізації системи: створення проєкту, розробка інтерфейсу, логіки введення, обробки та виведення даних.

У четвертому розділі проведено тестування системи, наведено результати та виконано їх аналіз.

У результаті виконання роботи створено функціональний мобільний додаток для прийняття рішень у багатокритеріальному середовищі.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

СППР — Система підтримки прийняття рішень

UI — Інтерфейс користувача (User Interface)

SDK — Набір інструментів для розробки програмного забезпечення (Software Development Kit)

AVD — Віртуальний пристрій Android (Android Virtual Device)

API — Програмний інтерфейс додатків (Application Programming Interface)

XML — Розширювана мова розмітки (eXtensible Markup Language)

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій проблема ухвалення ефективних рішень в умовах невизначеності набуває особливої актуальності. У багатьох сферах діяльності людина змушена приймати рішення, маючи обмежену або неточну інформацію, що вимагає використання спеціалізованих інструментів для підтримки процесу вибору.

Комп'ютерні системи підтримки прийняття рішень дозволяють підвищити обґрунтованість вибору, мінімізувати ризики та підвищити ефективність діяльності в складних ситуаціях. Розробка таких систем є актуальним завданням для інформаційних технологій, оскільки вони сприяють автоматизації процесів аналізу альтернатив і прийняття обґрунтованих рішень.

Дана бакалаврська кваліфікаційна робота присвячена розробленню комп'ютерної системи підтримки прийняття рішень в умовах невизначеності, що функціонує на базі мобільного застосунку. У рамках роботи буде проведено аналіз існуючих підходів до підтримки прийняття рішень, спроектовано архітектуру системи, реалізовано програмний продукт та виконано його тестування.

Актуальність теми зумовлена потребою в доступних, мобільних і простих у використанні інструментах для підтримки ухвалення рішень в реальному часі.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Поняття підтримки прийняття рішень в умовах невизначеності

Прийняття рішень — це одна з найважливіших когнітивних функцій людини, яка супроводжує її у повсякденному житті, професійній діяльності, соціальних комунікаціях та стратегічному плануванні. Незалежно від галузі — чи то бізнес, медицина, освіта, військова справа або особисте життя — людина постійно стикається з необхідністю обирати між кількома можливими варіантами дій. Прийняття рішення передбачає не лише вибір, але й оцінку альтернатив, прогнозування наслідків, співставлення критеріїв і готовність нести відповідальність за обраний варіант [1].

У низці випадків вибір буває інтуїтивно очевидним або ґрунтується на детермінованих даних. Наприклад, якщо людині потрібно обрати між двома маршрутами на роботу і один з них завжди швидший, вибір не викликає труднощів. Однак на практиці такі ситуації зустрічаються нечасто. У реальному світі інформація, на основі якої приймаються рішення, часто є неповною, суперечливою, неточною або змінюється з плином часу. Такі умови називають умовами невизначеності.

Невизначеність може мати різну природу. Наприклад:

Інформаційна невизначеність — коли недостатньо даних про об'єкт, процес або зовнішнє середовище.

Структурна невизначеність — коли не зовсім зрозуміло, як взаємодіють різні елементи системи.

Стохастична (ймовірнісна) невизначеність — коли існують випадкові фактори, які можуть впливати на результат.

Суб'єктивна невизначеність — пов’язана з людськими уподобаннями, експертними оцінками, психологією та емоціями.

Невизначеність може також бути наслідком часового обмеження — коли рішення потрібно ухвалити швидко, а час на повний аналіз відсутній. Часто це стосується таких критичних сфер, як медицина (діагноз та лікування),

військові операції або кризове управління (ліквідація наслідків стихійного лиха).

Щоб зменшити рівень невизначеності та підвищити обґрунтованість вибору, були створені системи підтримки прийняття рішень (СППР; англ. DSS — Decision Support Systems) [2]. Це спеціалізовані комп'ютерні системи, які допомагають людині або організації формалізувати процес прийняття рішення, оцінити всі доступні альтернативи, зіставити їх з критеріями, проаналізувати потенційні ризики та зробити логічно обґрунтований вибір.

Згідно з встановленими підходами, рішення можна класифікувати за ознаками, наведеними у таблиці 1.1

Основна мета СППР полягає не у тому, щоб замінити людину, а у тому, щоб підсилити її аналітичні можливості. Системи не дають прямої відповіді, що саме слід робити, а натомість пропонують варіанти, будують моделі, візуалізують наслідки і дозволяють користувачу краще зрозуміти ситуацію. СППР є інструментом аналітичної підтримки, який покращує якість управлінських рішень, особливо у складних і нестандартних ситуаціях.

Таблиця 1.1 - Типологія рішень

Критерій	Характеристика рішення
Ступінь формалізації проблеми	Добре структуровані / Слабко структуровані / Неструктуровані
Кількість етапів прийняття рішення	Один етап (статичне) / Багатоетапне (динамічне)
Інформованість про проблему	Визначеність / Ризик / Невизначеність
Кількість осіб, що приймають рішення	Одна особа / Група осіб
Тип рішення за змістом	Стратегічне / Тактичне

Типова структура СППР включає кілька основних компонентів [3]:

База моделей — набір математичних, логічних або емпіричних моделей, які дозволяють описати ситуацію, провести розрахунки, змоделювати поведінку системи.

База знань або база даних — зберігає необхідну інформацію для аналізу (вхідні параметри, історичні дані, правила, експертні оцінки).

Модуль обробки та аналізу даних — виконує логічні, статистичні, економіко-математичні обчислення.

Інтерфейс користувача — забезпечує взаємодію з користувачем, представлення результатів у зручній для сприйняття формі.

У ситуаціях, де рішення мають прийматися в умовах невизначеності, СППР використовують спеціальні методи і підходи:

Метод зважених оцінок — дозволяє порівнювати альтернативи, присвоюючи кожному критерію вагу відповідно до його важливості [4].

Метод аналізу ієрархій (AHP) — дозволяє декомпонувати складну проблему на підзадачі і визначити пріоритети шляхом парних порівнянь.

Методи нечіткої логіки — дозволяють працювати з нечіткими поняттями, які важко формалізувати (наприклад, "високий ризик", "достатній рівень").

Ймовірнісні моделі — базуються на теорії ймовірностей та очікуваних значеннях.

Сценарний аналіз — моделює декілька можливих сценаріїв майбутнього та оцінює вплив кожного з них [6].

Ці методи дозволяють значно підвищити точність оцінок та забезпечити гнучкість у прийнятті рішень.

У сучасному світі СППР використовуються у широкому спектрі сфер. Зокрема:

Бізнес та фінанси: стратегічне планування, управління ризиками, оптимізація логістики.

Медицина: підтримка лікарських рішень на основі симптомів, діагностика, вибір методу лікування.

Транспорт: вибір оптимальних маршрутів, планування навантаження.

Екологія: моделювання впливу на довкілля, прийняття рішень щодо природоохоронних заходів.

Управління підприємствами: автоматизація вибору інвестицій, проектів, закупівель.

Військова справа: оперативне управління, симуляції бойових сценаріїв, аналіз ризиків.

Створення та впровадження СППР має важливе практичне значення, адже сприяє прийняттю більш обґрунтованих рішень, знижує ризики, підвищує ефективність і зменшує втрати, зумовлені неправильними діями в умовах невизначеності.

1.2 Огляд існуючих методів прийняття рішень за невизначеності

Ухвалення рішень в умовах повної визначеності передбачає, що особа або система має вичерпну інформацію про всі можливі альтернативи, наслідки вибору кожної з них та значення відповідних критеріїв. Проте в реальному житті такі ідеальні умови трапляються рідко. Найчастіше рішення доводиться приймати у ситуації, коли частина інформації відсутня, неточна або суперечлива. У таких випадках ми говоримо про прийняття рішень в умовах невизначеності. Саме тому упродовж десятиліть вченими й інженерами були розроблені численні методи, які дозволяють формалізувати, аналізувати та обґрунтовувати рішення навіть за відсутності повної картини.

Одним з основоположних принципів у цьому контексті є уявлення про багатокритеріальність рішень [20], де кожна альтернатива оцінюється не за одним, а за кількома критеріями одночасно. Різні методи по-різному підходять до цього завдання, але головна мета залишається спільною — надати людині або автоматизованій системі можливість зробити зважений вибір у складних умовах.

Метод зважених оцінок

Одним із найпоширеніших та інтуїтивно зрозумілих підходів до прийняття рішень є метод зважених оцінок (англ. Weighted Sum Model). Його застосування полягає у кількісній оцінці альтернатив на основі заздалегідь визначених критеріїв [2]. Кожному критерію призначається вага — числове значення, що відображає його відносну важливість у контексті завдання. Далі всі альтернативи оцінюються за кожним критерієм (зазвичай за шкалою від 1 до 10), після чого підсумковий бал обчислюється як сума добутків ваг на відповідні оцінки.

Перевагами цього методу є:

простота реалізації;

прозорість для користувача;

гнучкість у налаштуванні критеріїв;

можливість застосування у різних галузях — від логістики до інвестицій.

Недоліком є чутливість до суб’єктивного вибору ваг та оцінок. Однак у поєднанні з іншими методами він дає хороші результати навіть за умов часткової невизначеності.

Метод аналізу ієрархій

Більш формалізованим і точним методом є метод аналізу ієрархій, розроблений американським науковцем Томасом Сааті [5]. Його ключова ідея полягає у побудові ієрархічної моделі задачі: на верхньому рівні розташовується глобальна ціль, далі — критерії та підкритерії, і лише на нижньому рівні — альтернативи вибору (рис.1.1).

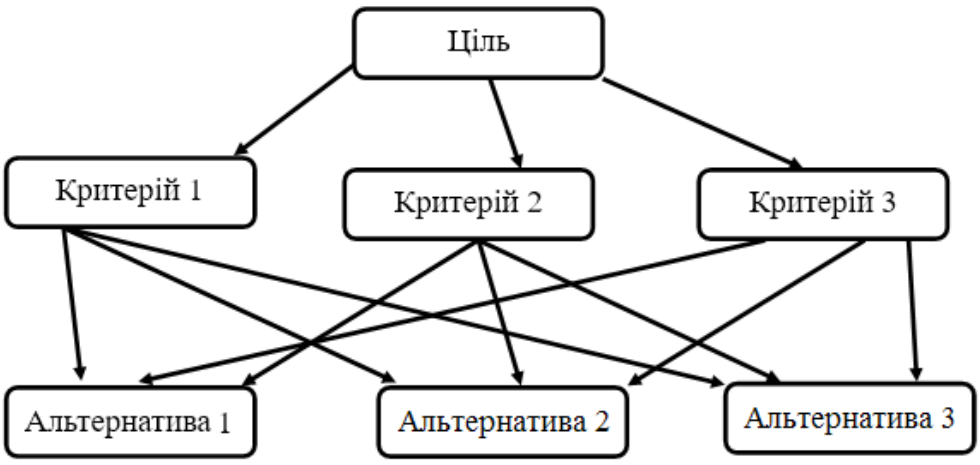

Рисунок 1.1 - Метод аналізу ієрархій

Процес оцінки полягає у парному порівнянні елементів кожного рівня ієрархії за шкалою від 1 до 9, після чого за допомогою спеціальних матричних перетворень система обчислює вектор пріоритетів, тобто вагові коефіцієнти для кожного критерію та альтернативи.

Цей метод має наступні переваги:

дозволяє враховувати як кількісні, так і якісні критерії;

забезпечує високу точність при великій кількості альтернатив;

добре адаптований до групових рішень (через агрегування оцінок декількох експертів).

Серед недоліків — необхідність значного часу на побудову ієрархії та виконання великої кількості парних порівнянь, особливо при великій кількості варіантів.

Метод сценаріїв [3]

Сценарний підхід ґрунтується на ідеї, що майбутнє може розвиватися за різними сценаріями, і жоден з них не має абсолютної гарантії реалізації. Тому замість того, щоб шукати «одне правильне рішення», аналізуються кілька можливих напрямків подій. Для кожного сценарію оцінюються можливі ризики, вигоди та наслідки, після чого порівнюється стійкість кожної альтернативи до змін.

Метод є особливо ефективним у стратегічному плануванні, де важливо врахувати вплив політичних, економічних, соціальних та технологічних факторів. Його використовують у банківській сфері, енергетиці, екології, обороні тощо.

Нечітка логіка

Коли інформація, на якій базується вибір, є не лише неповною, а й нечіткою (наприклад, «помірний ризик», «висока ймовірність», «досить вигідно»), традиційні логічні моделі не підходять. У таких випадках застосовується нечітка логіка (fuzzy logic), запропонована Лотфі Заде у 1965 році [7]. Вона дозволяє використовувати не бінарні значення істинності, а значення на проміжку [0;1], тобто оперувати «ступенем істинності» (рис. 1.2).

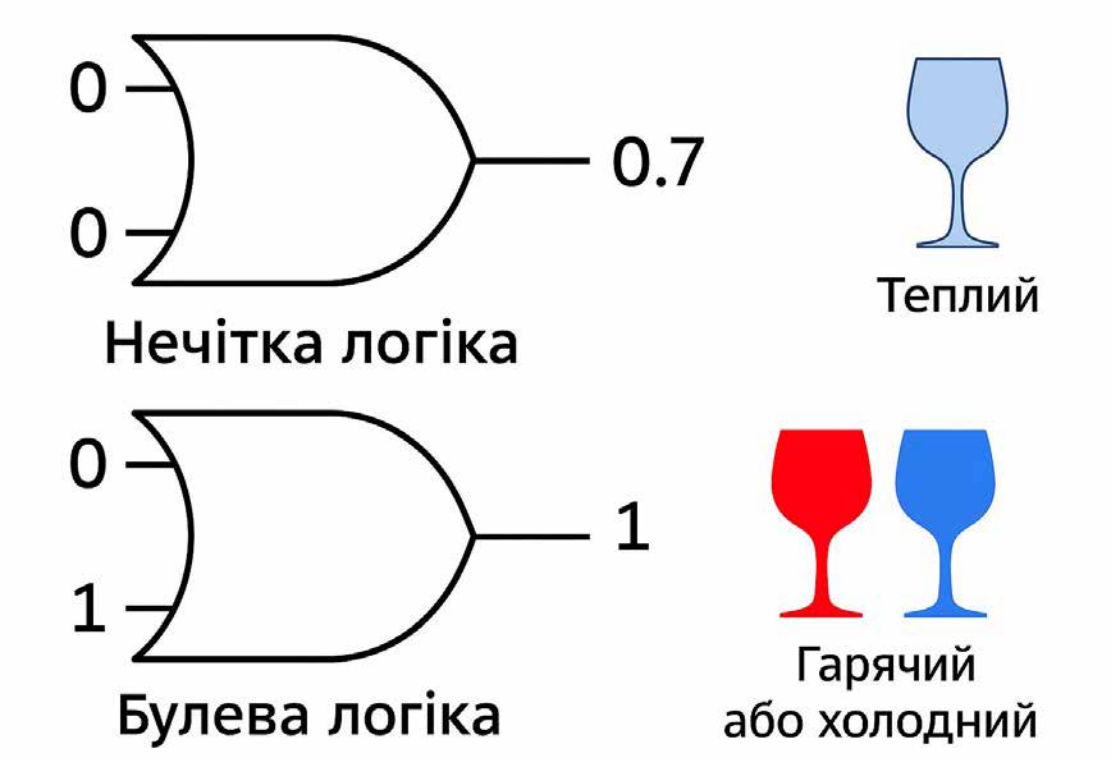

Рисунок 1.2 - Нечітка логіка в порівнянні з булевою

У рамках СППР нечітка логіка дозволяє:

моделювати експертні оцінки, виражені у лінгвістичній формі;

формалізувати судження, які інакше було б складно оцінити кількісно;

забезпечити гнучкість у формуванні правил прийняття рішень.

Застосування нечіткої логіки є поширеним у системах керування технікою, медичних діагностичних системах, інтелектуальних агентних системах, а також в адаптивному програмному забезпеченні.

Ймовірнісні методи: теорія корисності та байєсівський підхід

Якщо відомі ймовірності певних подій, доречним є використання теорії очікуваної корисності або байєсівського аналізу. У першому випадку кожна альтернатива оцінюється за формулою [8]:

Очікувана корисність = Σ (ймовірність події × корисність результату)

Байєсівський підхід, своєю чергою, дозволяє оновлювати оцінки ймовірностей у процесі надходження нової інформації. Він широко використовується у фінансовому аналізі, ризик-менеджменті, страхуванні, а також у штучному інтелекті — наприклад, у байєсівських мережах.

Дерева рішень

Ще одним інтуїтивно зрозумілим і наочним інструментом є дерева рішень [9]. Це графічні структури, які дозволяють відобразити всі можливі шляхи розвитку подій та варіанти рішень. Кожен вузол дерева — це точка вибору або випадкової події, а гілки — це наслідки. Кінцеві вершини (листи) містять результати.

Дерева рішень ефективні для:

покрокового аналізу складних процесів;

навчання машинного інтелекту (наприклад, у методах Decision Tree Learning);

візуалізації стратегії прийняття рішень.

Комбіновані методи та сучасні тенденції

У реальних системах підтримки прийняття рішень часто використовується гібридний підхід, коли методи комбінуються. Наприклад:

попередньо будується ієрархія рішень (AHP),

далі варіанти оцінюються методом зважених оцінок,

потім враховуються ймовірності сценаріїв,

а результати уточнюються за допомогою нечіткої логіки.

Крім того, сучасні СППР дедалі частіше інтегрують методи машинного навчання, обробки природної мови, нейронні мережі, що дозволяє автоматизувати обробку великих масивів даних і робити прогнози з використанням статистичних моделей.

Незважаючи на новітні підходи, основа ефективних СППР залишається незмінною: ретельне моделювання задачі, чітке формулювання критеріїв і використання перевірених методів прийняття рішень.

1.3 Проблема, яку вирішує система

Процес прийняття рішень супроводжує людину практично на кожному етапі її життя — як у побуті, так і в професійній діяльності. Сучасна людина

щодня змушена робити десятки рішень: обрати маршрут на роботу, придбати товар серед кількох альтернатив, обрати послугу, прийняти фінансове або організаційне рішення, що впливає на особисте чи колективне благополуччя. Зовні ці дії можуть здаватися простими, однак вони часто потребують урахування значної кількості факторів, критеріїв, обмежень і ризиків.

Особливо складними є ті випадки, коли не існує очевидно «правильного» варіанту, а всі альтернативи мають як переваги, так і недоліки. Наприклад, при покупці побутової техніки споживач змушений враховувати ціну, якість, надійність бренду, доступність сервісу, наявність гарантії, енергоефективність тощо. У випадку планування подорожі потрібно зважати на зручність маршруту, вартість квитків, тривалість переїзду, погодні умови, відгуки тощо. А в малому бізнесі — при виборі постачальника — критерії можуть включати не тільки ціну товару, а й терміни доставки, якість обслуговування, гнучкість співпраці, умови оплати, репутацію.

У подібних умовах виникає проблема багатокритеріальності, яка є складною для аналізу «на око». Люди, не маючи спеціальних інструментів, нерідко покладаються на інтуїцію, поради знайомих або випадковий вибір. Такий підхід часто призводить до нераціональних рішень, особливо в ситуаціях, коли результат має довготривалий ефект або пов'язаний з фінансовими витратами. Відсутність формалізованого методу оцінки альтернатив часто стає причиною незадоволення вибором у майбутньому, виникнення повторних витрат, втрати часу або репутаційних ризиків.

Незважаючи на те, що у науковій літературі та професійних інформаційних системах вже давно описані ефективні методи прийняття рішень (метод зважених оцінок, метод аналізу ієрархій, сценарний аналіз, нечітка логіка тощо), вони практично не інтегровані у повсякденне життя пересічного користувача. Причин кілька:

Більшість існуючих програм є надто складними для користування без спеціальної підготовки.

Інтерфейси орієнтовані на експертів або корпоративне середовище.

Немає адаптації під мобільні пристрої або мову повсякденного спілкування.

Часто відсутня підтримка українською мовою або інтуїтивна візуалізація результатів.

Звичайний користувач фактично позбавлений можливості користуватись науково обґрунтованими методами прийняття рішень, навіть у ситуаціях, коли це дійсно необхідно. Це створює очевидний розрив між теоретичними можливостями інформаційних технологій і практичним доступом до них у повсякденному середовищі.

Одним із можливих рішень є розроблення мобільного додатку, який дозволяє користувачу інтуїтивно і просто створити власну модель задачі прийняття рішення: ввести список альтернатив, задати критерії оцінки, вказати їхню важливість, надати оцінки — і миттєво отримати рекомендований вибір або рейтинг альтернатив. Подібний інструмент повинен бути (рис. 1.3):

Рисунок 1.3 – Проблеми який вирішує розроблений додаток

Простим у використанні — без потреби вивчення математичних моделей;

Зрозумілим і візуально зручним — щоб користувач міг швидко інтерпретувати результат;

Легким для адаптації — можливість змінювати критерії під конкретну ситуацію;

Мовно доступним — з українським інтерфейсом і локалізованими підказками;

Придатним для різних категорій користувачів — як для побутових потреб, так і для малого бізнесу.

Реалізація такого рішення у вигляді мобільного додатку на платформі Android є найбільш доцільною з огляду на поширеність цієї платформи, гнучкість у розробці, доступність для кінцевого користувача та можливість швидкого розгортання та тестування MVP-версії (Minimum Viable Product) [19].

Мотивацією до розробки є усунення бар'єру між науковими методами прийняття рішень і реальним життям пересічного користувача. Завдяки використанню сучасних інформаційних технологій можна досягти того, щоб кожна людина мала у своєму смартфоні доступ до простого і ефективного засобу вибору з декількох альтернатив — незалежно від того, чи йдеться про купівлю техніки, вибір туристичного маршруту або прийняття управлінського рішення у малому бізнесі.

1.4 Визначення основних вимог до комп'ютерної системи

Проєктування програмного забезпечення будь-якого рівня складності починається з формулювання чітких вимог. Саме вони визначають функціонал, обмеження, технічні особливості та критерії якості майбутньої системи. Для системи підтримки прийняття рішень (СППР), що реалізується у

вигляді мобільного додатку на платформі Android, необхідно враховувати як специфіку задачі, так і особливості цільової платформи.

Формулювання вимог має на меті не лише визначити, що саме повинен робити додаток, але й гарантувати відповідність очікуванням кінцевих користувачів, які можуть не мати технічної підготовки, але бажають мати доступ до зручного інструмента для логічного обґрунтування повсякденних або бізнес-рішень.

Вимоги до системи прийнято поділяти на функціональні та нефункціональні [14]. Функціональні вимоги описують, які саме дії повинен виконувати додаток, тоді як нефункціональні окреслюють якість реалізації функцій (швидкість, зручність, надійність тощо).

Функціональні вимоги

1. Введення варіантів рішень

Першим етапом у процесі прийняття рішень є визначення альтернатив. Додаток повинен надати користувачу можливість увести довільну кількість варіантів (наприклад, моделі телефонів, туристичних напрямків, постачальників товарів тощо), кожен з яких надалі буде аналізуватися за заданими критеріями.

Цей функціонал має бути реалізований через простий інтерфейс із кнопками «Додати варіант» / «Видалити варіант» та текстовими полями для введення назв альтернатив.

2. Введення критеріїв оцінювання

Кожна альтернатива має оцінюватися не загалом, а за конкретними критеріями. Наприклад, для вибору авто це можуть бути «ціна», «витрата пального», «рейтинг безпеки». Тому додаток має надати користувачу можливість створити перелік критеріїв, які він вважає важливими у конкретному випадку.

Інтерфейс повинен дозволяти вводити назву критерію, при необхідності — короткий опис. Кількість критеріїв не повинна обмежуватись жорстко, оскільки складність задачі може варіюватися.

3. Задавання ваги критеріям

Оскільки не всі критерії однаково важливі, система повинна дозволити користувачу задати вагу кожному з них. Вага — це числове значення, яке визначає вплив критерію на підсумковий результат. Наприклад, критерію «надійність» можна надати вагу 10, а «дизайну» — 5, якщо в конкретній задачі надійність є важливішою.

Варто використовувати шкалу від 1 до 10, яка є зрозумілою і достатньо гнучкою для побутових і бізнес-сценаріїв. Для спрощення введення можна використовувати повзунки або випадаючі списки.

4. Обчислення підсумкового результату

Основна функція додатку полягає у застосуванні методу зважених оцінок: обчисленні рейтингу кожної альтернативи на основі оцінок та ваг критеріїв. Результати мають генеруватися автоматично після заповнення даних, без потреби додаткових дій з боку користувача.

5. Виведення результатів у зручному вигляді

Користувач повинен бачити результат не лише у числовій формі, а й у зрозумілому та наочному представленні. Для цього варто реалізувати:

Таблицю з усіма альтернативами, критеріями, балами та підсумковими оцінками;

Гістограму або кругову діаграму з візуалізацією порівняння варіантів;

Вказівку на найкращий варіант відповідно до обчислень.

Результати мають оновлюватися в режимі реального часу після зміни будь-яких параметрів.

6. Можливість скидання або очищення даних

Для забезпечення зручності багаторазового використання, додаток повинен мати кнопку «Очистити дані», яка видаляє всі введені альтернативи, критерії, ваги та результати. Це дозволить швидко повторно скористатися додатком для нової задачі без потреби перезапуску програми.

Нефункціональні вимоги

1. Простий інтерфейс

Зважаючи на орієнтацію на масового користувача, який не має технічної освіти, інтерфейс повинен бути максимально інтуїтивним і доступним. Варто уникати складної термінології, надлишкових кнопок, багаторівневих меню. Усі функції повинні бути доступними за 1–2 натискання.

2. Автономна робота

Система повинна бути повністю офлайн — жодна з основних функцій не має залежати від підключення до Інтернету. Це забезпечить надійність, конфіденційність і доступність у будь-яких умовах, навіть за відсутності мережі.

3. Швидкість обчислень

Обчислення результатів мають відбуватись миттєво. Зважаючи на невелику кількість операцій (множення, підсумовування), додаток повинен демонструвати високу продуктивність навіть на пристроях середнього класу. Затримка між введенням даних і виведенням результату не повинна перевищувати 0.5 секунди.

4. Адаптивність до розміру екрану

Додаток повинен коректно відображатися на різних пристроях Android — смартфонах з різною діагоналлю, роздільною здатністю та орієнтацією (портретною чи ландшафтною). Для цього необхідно застосувати адаптивні компоненти інтерфейсу, такі як ConstraintLayout або Jetpack Compose з підтримкою responsive-дизайну [15].

5. Надійність роботи

Найважливіша вимога для будь-якого мобільного додатку — стійкість до помилок і аварій. Програма не повинна «вилітати» при типових діях користувача: введенні великої кількості критеріїв, випадковому видаленні елементів, помилковому натисканні. Також важливо передбачити базову перевірку введених даних (наприклад, недопущення порожніх полів або некоректних значень).

Чітке формулювання вимог дозволяє закласти основу для ефективного проєктування та розробки мобільної системи підтримки прийняття рішень, яка

буде одночасно функціонально повною, зручною для користувача та технічно стабільною. Наступним кроком є розроблення архітектури програми та моделювання її структури у вигляді діаграм.

У результаті проведеного аналізу встановлено, що прийняття рішень в умовах невизначеності є актуальною задачею, з якою щодня стикаються як пересічні користувачі, так і бізнес-структури. Існуючі методи, зокрема метод зважених оцінок, демонструють високу ефективність у багатокритеріальному аналізі, але залишаються малодоступними для широкого кола користувачів через складність реалізації. На основі аналізу проблемної області та сформульованих функціональних і нефункціональних вимог обґрунтовано доцільність створення мобільної комп’ютерної системи у вигляді Android-додатку, яка має інтуїтивний інтерфейс та реалізує простий і зрозумілий алгоритм вибору. Це створює підґрунтя для наступного етапу — проектування архітектури системи, вибору інструментів розробки та моделювання її функціональної структури.

2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Вибір середовища розробки та технічних засобів

Успішна реалізація будь-якого програмного продукту залежить не лише від коректної постановки задачі та моделювання архітектури, а й від правильного вибору технологічного стеку — мов програмування, середовища розробки, додаткових бібліотек, платформ для тестування та відлагодження. Це особливо актуально у випадку мобільної розробки, яка має низку специфічних вимог, пов’язаних із обмеженими ресурсами пристроїв, різноманіттям екранів та необхідністю адаптації під різні версії операційної системи.

Оскільки метою даного проєкту є створення мобільного застосунку для підтримки прийняття рішень користувачами, що не мають спеціальної підготовки, було обрано найпопулярнішу у світі мобільну платформу — Android. Це рішення продиктовано як ринковими, так і технічними факторами, розглянемо їх детальніше.

Вибір платформи Android

Android є домінуючою мобільною операційною системою, яка підтримується на переважній більшості смартфонів, планшетів та навіть побутових пристроїв. За даними StatCounter Global Stats, станом на 2024 рік Android займає понад 70% світового ринку мобільних операційних систем, що забезпечує надзвичайно широку аудиторію потенційних користувачів додатка.

Серед ключових переваг Android як платформи для розробки:

Відкрита архітектура. На відміну від закритих платформ (наприклад, iOS), Android дозволяє повний доступ до системних функцій, що забезпечує гнучкість у реалізації функціоналу.

Велика спільнота розробників. Існує велика кількість відкритих форумів, навчальних курсів, документації та прикладів коду, які спрощують навчання і пришвидшують процес розробки.

Широкий спектр підтримуваних пристроїв. Додаток, створений під Android, потенційно може працювати на десятках тисяч моделей смартфонів різних виробників, що розширює його застосовність.

Гнучкість у розповсюдженні. Завантаження додатку можливе як через офіційний магазин Google Play, так і вручну — у форматі APK-файлу, що спрощує тестування та розгортання на закритих пристроях.

Усі ці переваги роблять платформу Android логічним вибором для реалізації MVP (Minimum Viable Product) — тобто початкової версії програмного забезпечення з базовим функціоналом, який можна поступово розширювати.

Середовище розробки: Android Studio

Для безпосередньої розробки було обрано Android Studio — офіційне інтегроване середовище розробки (IDE), рекомендоване компанією Google [11]. Android Studio базується на платформі IntelliJ IDEA і є повноцінним інструментом, який об’єднує в собі:

Візуальний редактор інтерфейсу користувача (UI Designer) — дозволяє створювати макети за допомогою перетягування елементів.

Емулятор Android-пристроїв — надає можливість тестування застосунку без фізичного смартфона.

Інструменти налагодження та аналізу продуктивності — Android Profiler, Logcat, Debugger.

Гнучку систему складання проектів на основі Gradle — дозволяє автоматизувати складання APK, підключення бібліотек тощо.

Вбудовану підтримку мов Java та Kotlin — з можливістю переходу між ними.

Android Studio є кросплатформним середовищем і може бути встановлена на Windows, Linux або macOS. Воно постійно оновлюється, має підтримку сучасних бібліотек Android Jetpack, що дозволяє реалізовувати адаптивні інтерфейси, навігацію, збереження даних та інші компоненти за найкращими практиками розробки.

Вибір мови програмування

Хоча Google активно просуває мову Kotlin як основну для Android-розробки, у цьому проекті була використана Java — з огляду на її зрозумілість, гнучкість і популярність серед початківців. Серед переваг Java [12]:

Великий обсяг доступної документації;

Величезна база відкритих бібліотек;

Можливість швидкої інтеграції сторонніх рішень;

Легкість підтримки коду;

Зручність для налагодження.

Крім того, Android SDK зберігає повну сумісність із Java-кодом, що дозволяє використовувати його у всіх частинах застосунку — від бізнес-логіки до роботи з базами даних та елементами інтерфейсу.

Технічні засоби розробки

Розробка програмного забезпечення відбувалася на персональному комп’ютері з наступними характеристиками:

Процесор: Intel Core i7-9700K;

Оперативна пам’ять: 32 ГБ;

Операційна система: Microsoft Windows 10;

Накопичувач: 2 ТБ;

Таке апаратне забезпечення дозволяє комфортно запускати Android Studio, швидко збирати проект (build), запускати кілька емуляторів одночасно та проводити тестування без затримок.

Засоби тестування

Тестування додатку проводилося на вбудованому Android-емуляторі, який входить до складу Android Studio. Це дозволило:

Швидко імітувати поведінку різних пристроїв із різними розмірами екрана та версіями Android (від 8.0 до 13.0);

Відлагоджувати додаток у режимі реального часу (hot reload);

Перевірити адаптивність інтерфейсу;

Виявити помилки без потреби в підключенні фізичного смартфона.

Крім того, реалізація додатку у вигляді APK-файлу дозволяє передавати його на фізичні пристрої для тестування в умовах реального використання.

Для реалізації проєкту було обрано перевірене, стабільне та підтримуване середовище Android Studio, яке забезпечує весь цикл розробки — від створення інтерфейсу до тестування. Мова програмування Java дозволила швидко реалізувати необхідний функціонал і забезпечити зрозумілу архітектуру коду. Технічні характеристики ПК відповідають сучасним вимогам, а засоби тестування дозволили перевірити додаток у реальних сценаріях. Обраний стек технологій повністю відповідає цілям дипломного проєкту та дозволяє у майбутньому масштабувати розробку.

2.2 Обґрунтування вибору Android-платформи і мови програмування Java

Процес створення інформаційної системи, зокрема мобільного додатку, передбачає ухвалення низки ключових рішень, що впливають як на технічну реалізацію проєкту, так і на подальшу зручність користування програмним продуктом. Одне з таких рішень — вибір операційної платформи, середовища розробки та мови програмування, які найкраще відповідають поставленим цілям і обмеженням.

У межах цієї дипломної роботи було прийнято рішення розробляти програму саме для мобільної платформи Android. Це обумовлено низкою вагомих факторів, які стосуються поширеності, доступності інструментів, технічної відкритості системи, а також простоти навчання у рамках освітнього процесу.

Android — найпопулярніша мобільна платформа у світі

На сьогодні Android є найбільш поширеною мобільною операційною системою, що функціонує на смартфонах, планшетах, смарт-годинниках, телевізорах та інших пристроях. За даними таких авторитетних аналітичних компаній, як StatCounter [10] та IDC, частка Android на глобальному ринку

мобільних пристроїв перевищує 70% (станом на 2024 рік). Це означає, що програма, створена для цієї платформи, матиме потенціал охопити найширше коло користувачів без необхідності адаптації під інші менш поширені операційні системи.

З погляду кінцевого користувача, це дозволяє застосунку бути доступним на пристроях різного цінового сегмента — від бюджетних до флагманських, що є перевагою при розробці для широкої аудиторії. З погляду розробника, це дозволяє не обмежувати застосування рішень лише для власників специфічних пристроїв (наприклад, iPhone або iPad), як у випадку з платформою iOS.

Відкритість платформи та доступ до інструментів

Ще однією важливою перевагою Android є її відкрита архітектура, що забезпечує широкі можливості для розробників. Google надає у відкритому доступі SDK (Software Development Kit), API, документацію та численні приклади реалізації. Це дозволяє навіть студентам з мінімальним досвідом програмування швидко входити в розробку і реалізовувати прототипи застосунків.

У порівнянні з альтернативними платформами, такими як iOS, розробка під Android не потребує спеціалізованого обладнання (наприклад, комп’ютерів Apple), платних ліцензій або обмеженого доступу до середовищ розробки. Все необхідне для створення додатку — Android Studio, SDK, емулятор, офіційна документація — доступне безкоштовно та легально, що ідеально підходить для освітніх і дослідницьких цілей.

Крім того, Android підтримує розповсюдження програмних продуктів через альтернативні канали — не лише через офіційний магазин Google Play, але й шляхом прямого встановлення APK-файлів. Це значно спрощує процес тестування та демонстрації застосунку без необхідності публікації в онлайн-магазині.

Android Studio — офіційне та ефективне середовище розробки

Для створення мобільного додатку було обрано Android Studio — офіційне інтегроване середовище розробки (IDE), рекомендоване Google. Воно має низку суттєвих переваг:

Підтримка візуального дизайну інтерфейсу завдяки Layout Editor;

Вбудовані емулятори мобільних пристроїв різних розмірів і версій ОС;

Інтегровані інструменти для налагодження, тестування і профілювання;

Підтримка обох основних мов програмування Android — Java та Kotlin;

Можливість використання сучасних бібліотек Android Jetpack [17].

Android Studio постійно оновлюється, має розширену базу плагінів і підтримує найкращі практики проєктування [18] (MVVM, DI, Material Design тощо). Його використання дозволяє реалізувати додаток згідно з вимогами сучасного програмного забезпечення, не виходячи за рамки навчального проєкту.

Обґрунтування вибору мови програмування Java

Для реалізації дипломного проєкту було обрано мову програмування Java. Хоча останніми роками компанія Google активно просуває мову Kotlin, Java зберігає лідерські позиції в навчальному процесі та у великій кількості проектів із відкритим кодом. Основні причини такого вибору наведено нижче.

1. Довготривала підтримка Android

Java була першою основною мовою для розробки під Android. Протягом багатьох років вона використовувалася у всіх офіційних прикладах, документації та навчальних курсах. Навіть сьогодні велика частина доступних відкритих ресурсів — бібліотеки, API, плагіни — мають реалізації на Java.

2. Простота засвоєння

Java є мовою з чітким, структурованим синтаксисом, який легко засвоюється початківцями. У контексті навчального проєкту це забезпечує швидкий старт, легке виявлення помилок і високу читабельність коду. Для студентів це критично важливо, особливо в умовах обмеженого часу на опанування нового інструментарію.

3. Доступність навчальних матеріалів

Java має широку освітню підтримку. Доступні безкоштовні онлайн-курси (наприклад, на платформах Coursera, Udemy, Stepik), навчальні відео, документація від Oracle, книжки, а також велика кількість прикладів і готових рішень. Це значно полегшує самостійне вивчення мови та розробку застосунку без необхідності постійного звернення до викладача.

4. Підтримка інтеграції з Kotlin

Важливо зазначити, що використання Java не обмежує майбутній розвиток проєкту. Android Studio підтримує змішане використання Java та Kotlin, а це означає, що за потреби окремі модулі можна буде реалізувати на новій мові, не переписуючи всю програму.

Вибір Android як цільової платформи, Android Studio як середовища розробки та Java як основної мови програмування є обґрунтованим, доцільним і практичним рішенням для реалізації мобільної системи підтримки прийняття рішень у рамках дипломного проєкту. Це рішення дозволило:

Забезпечити доступність продукту для більшості користувачів смартфонів;

Забезпечити гнучкість і технічну простоту реалізації;

Оптимізувати навчальний процес для розробника;

Забезпечити масштабованість проєкту у майбутньому.

2.3 Архітектура комп’ютерної системи

У процесі створення інформаційних систем особливе значення має правильна побудова архітектури програмного забезпечення. Від неї залежить не лише функціональність програми, а й зручність у підтримці, масштабованість, ефективність налагодження та швидкість розвитку системи в майбутньому. Для розробленого в межах цієї дипломної роботи мобільного додатку було обрано модульну архітектуру, яка передбачає поділ логіки на окремі, слабо зв’язані компоненти, кожен з яких виконує чітко визначену функцію.

Такий підхід дозволяє:

ізолювати функціональність між модулями;

спростити тестування кожного компонента окремо;

забезпечити повторне використання коду;

полегшити внесення змін до окремих частин програми без впливу на решту системи.

Загальна структура системи

Архітектура додатку передбачає наявність п’яти основних модулів, які взаємодіють між собою у чіткій послідовності (рис.2.4).

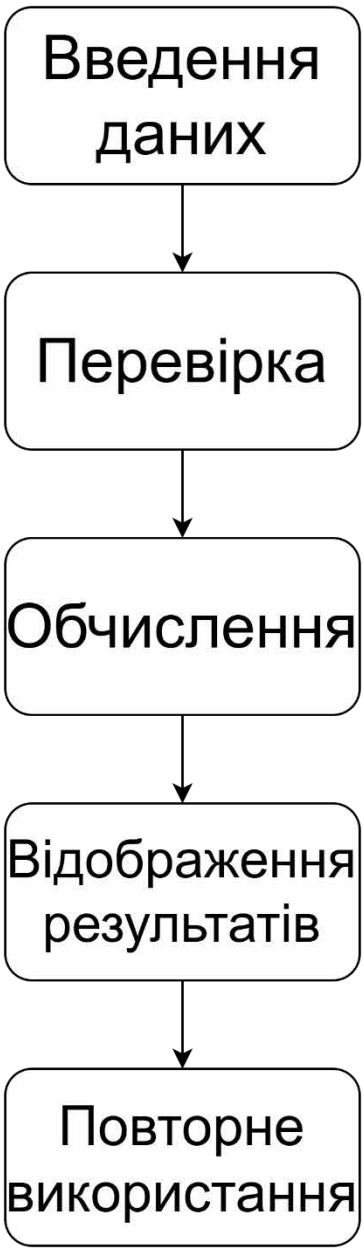

Рисунок 2.4 - Загальна архітектура

Уся логіка роботи користувача з додатком ґрунтується на простому алгоритмі: введення даних → перевірка → обчислення → відображення результатів → повторне використання. Кожен етап реалізований окремим функціональним блоком.

Нижче наведено опис кожного модуля.

1. Інтерфейс користувача (UI)

Цей модуль відповідає за взаємодію програми з користувачем. Він реалізований у вигляді окремих екранів (activity або fragment у термінах Android), які містять:

Текстові поля для введення назв альтернатив та критеріїв;

Випадаючі списки або повзунки для визначення ваги кожного критерію;

Кнопки для підтвердження введення, запуску розрахунку та очищення форми;

Вікна виведення результатів у табличній або графічній формі.

Інтерфейс спроектовано з урахуванням принципів юзабіліті та адаптивності, що дозволяє додатку коректно відображатися на різних мобільних пристроях з різною діагоналлю екрана.

Цей модуль не містить бізнес-логіки, а лише реєструє дії користувача та передає їх до відповідних обробників (контролерів або сервісів).

2. Модуль введення та перевірки даних

Відповідає за збір і первинну обробку даних, що надходять від користувача. Основні функції:

зберігання введених значень альтернатив та критеріїв;

перевірка на повноту та правильність введення (наприклад, заборона введення порожніх назв, перевірка числових значень ваг);

забезпечення логічної цілісності (кожен критерій повинен мати вагу, а кожна альтернатива — оцінки за всіма критеріями).

У разі виявлення помилки (наприклад, порожнього поля або нечислового значення) користувач отримує інформативне повідомлення, після чого дані не передаються на наступний етап.

3. Обчислювальний модуль

Центральна частина додатку — модуль розрахунку підсумкових результатів. Він реалізує метод зважених оцінок, який полягає у визначенні загального балу для кожної альтернативи на основі введених оцінок і ваг критеріїв.

4. Модуль очищення даних

Завершальний компонент системи — модуль очищення введених даних. Його призначення полягає у:

очищенні всіх текстових полів;

обнуленні оцінок і ваг;

скиданні стану інтерфейсу до початкового вигляду.

Цей модуль активується натисканням кнопки «Очистити» і дозволяє повторно використати програму без її перезапуску.

Послідовність роботи системи

Логіка роботи мобільного застосунку організована за лінійною структурою з можливістю повернення назад. Взаємодія між модулями відбувається у наступному порядку:

Користувач запускає програму та вводить необхідні дані (альтернативи, критерії, ваги).

Модуль валідації перевіряє правильність введених значень.

У разі успішної перевірки, дані передаються до модуля обчислень.

Після завершення розрахунку результати передаються до модуля виведення, де виводяться у табличній та графічній формі.

За потреби користувач може скористатися модулем очищення для повторного введення даних.

2.4 Проєктування структури даних і алгоритмів

Проєктування структури даних і реалізація алгоритмів обробки введеної інформації є ключовим етапом у створенні системи підтримки прийняття

рішень. У межах розробки мобільного застосунку для Android було прийнято рішення побудувати логіку обчислень на основі одного з найпростіших та найефективніших методів багатокритеріального вибору — методу зваженої суми (Weighted Sum Method). Цей метод реалізований у програмному коді з використанням елементарних структур даних і повністю функціонує в оперативній пам’яті пристрою без залучення баз даних чи зовнішніх джерел.

Алгоритм прийняття рішень

В основі функціональності додатку лежить принцип обчислення зваженого балу для кожного варіанту вибору, який користувач вводить вручну через інтерфейс. Програма запитує:

перелік варіантів рішень;

набір критеріїв (із зазначенням важливості кожного у вигляді ваги);

числові оцінки варіантів за кожним критерієм.

Після введення всіх даних користувач натискає кнопку для запуску обчислення, і додаток обробляє введену інформацію наступним чином:

Для кожного варіанту виконується множення кожної оцінки на відповідну вагу критерію.

Отримані добутки підсумовуються, утворюючи загальний бал.

Усі варіанти ранжуються за зменшенням цього балу.

Результат відображається у вигляді табличного списку.

Цей механізм дозволяє користувачу швидко і наочно побачити, який варіант є найкращим з урахуванням усіх заданих критеріїв.

Принципи обробки

Жодного збереження на диск не відбувається. Всі змінні та масиви існують лише під час роботи програми.

Перевірка правильності введення відбувається лише частково: за замовчуванням система очікує, що дані введені у правильному числовому форматі.

Очищення даних відбувається шляхом натискання кнопки "Очистити", яка перезапускає активність.

Спрощена логіка — перевага для мобільного середовища

Таке рішення не потребує бази даних, складного контролю станів чи асинхронної обробки. Завдяки цьому:

додаток швидко завантажується;

не потребує доступу до інтернету;

не займає багато пам’яті;

легко тестується та розширюється.

Це робить реалізацію зручною не лише для розробника-початківця, а й для кінцевого користувача, який не має досвіду роботи зі складними системами.

Проєктування структури даних у мобільному додатку було здійснено на основі мінімально необхідних структур, достатніх для реалізації алгоритму зваженої суми. Програма не використовує базу даних або збереження файлів, працюючи повністю в оперативній пам’яті. Це забезпечує швидкість, простоту та стабільність роботи, що є ключовими факторами у створенні мобільних рішень, особливо у навчальних або прикладних проектах.

Таким чином, архітектура даних та алгоритм обробки у додатку є оптимальними для поставленої задачі — забезпечення зручного вибору найкращого рішення на основі кількох критеріїв у зрозумілому та швидкому форматі.

2.5 Проєктування інтерфейсу користувача

Інтерфейс користувача (UI) є важливою складовою будь-якої прикладної програми, особливо у випадках, коли програмне забезпечення призначене для широкого кола користувачів. У межах розробки мобільного застосунку для підтримки прийняття рішень в умовах невизначеності особливу увагу було приділено простоті, доступності та інтуїтивності інтерфейсу. Основне завдання полягало в тому, щоб зробити процес введення даних і отримання

результатів максимально зручним і зрозумілим навіть для користувача без технічної підготовки.

Загальні принципи побудови інтерфейсу

Під час проєктування інтерфейсу було використано підхід, який відповідає сучасним рекомендаціям щодо мобільних додатків для Android. Основними принципами стали:

Мінімалізм — інтерфейс не перевантажений елементами, на екрані одночасно відображаються лише необхідні поля;

Логічна послідовність дій — кожен етап роботи з додатком представлений окремим екраном;

Зрозуміле групування полів — поля введення об’єднані у вертикальні блоки з підписами;

Адаптивність — програма підтримує довільну кількість введених варіантів і критеріїв;

Прокрутка — використано ScrollView, що дозволяє комфортно працювати навіть з великою кількістю полів;

Світла тема з контрастними елементами — забезпечує хорошу читабельність на екранах різних розмірів.

Структура інтерфейсу

Інтерфейс додатку поділений на три основні екрани (activity), кожен з яких виконує окрему функцію в загальному сценарії користувача.

1. Головне меню (MainActivity)

Це початковий екран, на якому користувач вводить кількість варіантів і критеріїв. Інтерфейс складається з таких елементів:

EditText для введення кількості варіантів;

EditText для введення кількості критеріїв;

Button «Перейти», яка запускає перехід до екрана введення даних (InputActivity);

Усі елементи розміщено у LinearLayout з вертикальною орієнтацією.

Цей екран є максимально простим і виконує роль налаштування наступних етапів роботи програми.

2. Введення даних (InputActivity)

Другий екран призначений для введення основної інформації, яка використовується в обчисленнях. Всі елементи створюються динамічно через Java-код залежно від кількості, яку користувач вказав раніше. Це дозволяє реалізувати масштабований інтерфейс, який працює з будь-якою кількістю альтернатив та критеріїв.

Структура включає:

Поля EditText для введення назв альтернатив;

Таблицю з EditText для назв критеріїв;

Таблицю для введення ваг критеріїв (одне поле на кожен критерій);

Таблицю оцінок (матриця альтернатива × критерій);

Button «Розрахувати результат», яка відкриває ResultActivity.

Для зручності реалізовано ScrollView, який дозволяє прокручувати вміст при перевищенні розміру екрана.

3. Результати (ResultActivity)

Третій екран відповідає за відображення підсумкових результатів розрахунків. На цьому екрані:

Виводиться список усіх альтернатив разом із їхніми підсумковими балами;

Інформація подається у вигляді TextView, сформованого як таблиця у вигляді тексту;

Усі результати відсортовано за спаданням, тобто найкращий варіант знаходиться зверху.

Інтерфейс цього екрану максимально лаконічний — лише результат, без додаткових елементів.

Візуальні особливості

Інтерфейс побудовано з використанням базових елементів Android:

EditText — для введення чисел, назв варіантів і критеріїв;

Button — для підтвердження дій;

TextView — для відображення результатів;

LinearLayout — для компонування елементів;

ScrollView — для прокручування вмісту при великій кількості полів.

Кольорова схема — світла тема. Основні кнопки, які відповідають за перехід або запуск обчислення, стилізовані зеленим кольором для візуального підкреслення завершення етапу (згідно з логікою — «готово», «продовжити»).

Підказки та зручність введення

Окрему увагу було приділено зручності — програму можуть використовувати користувачі без досвіду роботи з інформаційними системами. Тому:

У полях введення є hint-підказки: наприклад, «Наприклад: 3» для кількості;

Розміщення елементів логічне — кнопки завжди розташовано нижче відповідних груп полів, що відповідає природному порядку взаємодії користувача з екраном;

Усі компоненти мають відступи та вирівнювання по центру, що покращує візуальне сприйняття;

Відсутність зайвих іконок, випадаючих меню чи анімацій забезпечує швидку роботу навіть на слабких пристроях.

Весь процес взаємодії побудовано за сценарієм «від простого до складного»: від вибору кількості до виводу результатів (рис 2.5). Інтерфейс є повністю адаптивним і динамічним — завдяки створенню полів у Java-коді програма може працювати з будь-якою кількістю вхідних даних.

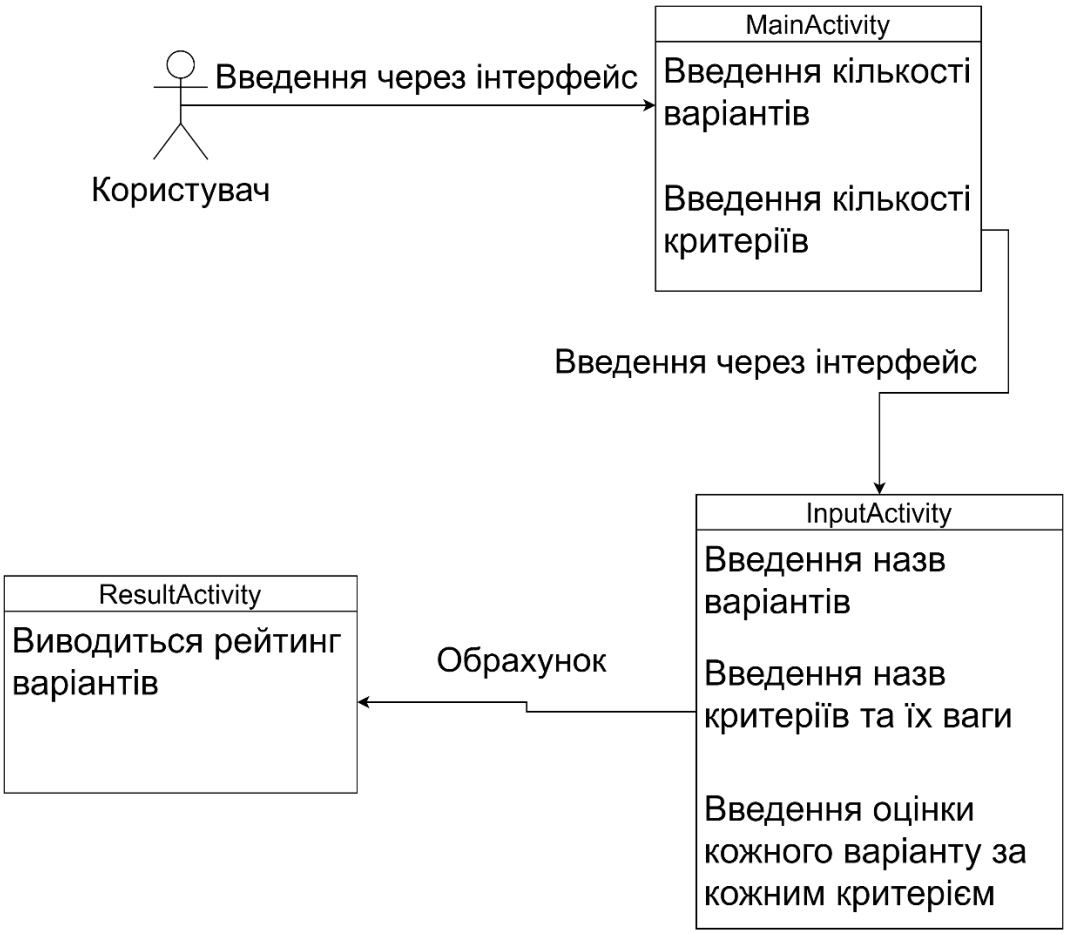

Рисунок 2.5 – Загальний сценарій

Такий підхід не лише спрощує розробку та супровід додатку, а й робить його зручним для кінцевого користувача. Це повністю відповідає вимогам до сучасних мобільних додатків, особливо в освітніх, прикладних або побутових умовах.

У ході проектування комп’ютерної системи було обґрунтовано вибір мобільної платформи Android та мови програмування Java як найбільш доцільних з огляду на доступність інструментів, популярність серед користувачів і простоту реалізації. Визначено модульну архітектуру, яка забезпечує зручність підтримки та масштабованість додатку. Розроблено структуру даних і алгоритм зваженої суми, що дозволяє ефективно обробляти введені користувачем оцінки. Продумано інтерфейс користувача відповідно до принципів мінімалізму та адаптивності. Таким чином, на основі проектних рішень створено чітку технічну основу.

3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Створення базового проєкту у середовищі Android Studio

На першому етапі реалізації мобільної комп’ютерної системи підтримки прийняття рішень в умовах невизначеності було створено базовий каркас проєкту в офіційному середовищі розробки Android Studio. Цей етап є обов’язковим для будь-якого Android-додатку, оскільки саме тут визначається початкова структура, мова програмування, тип активності, налаштовуються залежності, теми оформлення та ресурси інтерфейсу.

Вибір середовища розробки

Android Studio було обрано з кількох причин:

це офіційне IDE від Google, яке має повну підтримку інструментів розробки для Android;

забезпечує інтеграцію з Gradle, систему збирання, що дозволяє легко керувати залежностями;

надає зручні засоби для візуального редагування макетів інтерфейсу (layout editor);

має вбудований емулятор Android Virtual Device (AVD) для тестування застосунку без фізичного пристрою [16].

Під час налаштування Android Studio були встановлені всі рекомендовані компоненти SDK, включно з Android SDK Tools, Emulator, Build-Tools та Image-системами.

Створення проєкту

Для створення нового Android-додатку було обрано шаблон Empty Activity. Цей шаблон створює мінімальну стартову конфігурацію з одним екраном (активністю) і є зручним для проектів, у яких розробник самостійно вибудовує архітектуру застосунку з нуля.

У процесі створення було вказано наступні параметри:

Назва застосунку: DecisionSupportApp — назва, яка відображатиметься на екрані пристрою.

Назва пакета: автоматично згенерована IDE.

Мова програмування: Java — з огляду на сумісність, документацію та простоту для студентського проєкту.

Мінімальна версія SDK: Android 5.0 (API 21) — забезпечує сумісність із понад 90% пристроїв.

Тип активності: Empty Activity — стартова точка для розгортання власного інтерфейсу.

Назва основної активності: MainActivity.

Після підтвердження налаштувань Android Studio автоматично згенерувала базову структуру проєкту та початкові файли.

Структура проєкту

Файли проєкту було організовано згідно зі стандартами Android-розробки:

/java/com.example.decisionsupportapp/ — містить усі Java-файли активностей:

MainActivity.java — головна активність, яка керує першим екраном;

InputActivity.java — логіка другого етапу (введення даних);

ResultActivity.java — екран виводу результатів.

/res/layout/ — XML-файли інтерфейсу:

activity_main.xml — макет головного меню;

activity_input.xml — макет форми введення;

activity_result.xml — макет результатів.

/res/values/:

strings.xml — текстові ресурси;

colors.xml — палітра кольорів для кнопок, фону, тексту;

themes.xml — загальні стилі застосунку (на основі Theme.MaterialComponents.Light).

AndroidManifest.xml — файл конфігурації додатку, де задекларовано всі активності, а також головну активність, яка запускається першою (MainActivity).

build.gradle (Module: app) — конфігурація залежностей та налаштування збірки. Тут за замовчуванням вказані:

версія SDK: compileSdk 33, minSdk 21;

залежності appcompat, material, constraintlayout;

використання Java 8 (sourceCompatibility JavaVersion.VERSION_1_8).

Налаштування віртуального пристрою

Для тестування додатку було створено емулятор Android Virtual Device (AVD). Цей інструмент дозволив проводити перевірку роботи застосунку на віртуальному смартфоні без потреби в реальному пристрої.

Параметри емулятора:

Пристрій: Google Pixel 4

ОС: Android 11 (API 30)

Роздільна здатність: 1080x2280 px

Пам’ять RAM: 1536 МБ

Це дало змогу перевірити зовнішній вигляд інтерфейсу, реакцію на натискання кнопок, працездатність Java-коду та коректність передачі даних між активностями.

Налаштування зовнішнього вигляду

Після створення каркасу проєкту було виконано базове стилістичне налаштування:

обрана світла тема (Light Theme) із системної палітри Android Material Design;

змінено основний колір елементів керування (кнопки — зелений відтінок);

у файлі colors.xml додано нові значення для primaryColor, secondaryColor, buttonColor;

у themes.xml налаштовано розміри шрифтів і фон для вікон.

Ці дії дозволили сформувати стильну та сучасну візуальну оболонку для подальшої реалізації функціональності.

Оновлення компонентів

На цьому етапі було також оновлено наступні компоненти:

Android SDK Tools, Build-Tools, Platform-Tools;

Gradle Plugin до останньої стабільної версії;

перевірено доступність останніх версій бібліотек support (androidx.appcompat, material, constraintlayout).

Також було протестовано збирання проєкту через Gradle Build (Build > Rebuild Project), щоб переконатися у відсутності конфліктів чи помилок налаштування середовища.

Результатом першого етапу реалізації стало створення повноцінної структури проєкту в Android Studio з правильною архітектурою, розділенням на активності та ресурсами. Усі компоненти працюють у тестовому середовищі, і система повністю готова до подальшого заповнення функціональністю.

Цей етап закладає фундамент усієї програми, забезпечуючи зручність розробки, тестування та супроводу. Наступним кроком є реалізація внутрішньої логіки введення, обробки та виведення даних, що буде детально описано в наступних підрозділах.

3.2 Реалізація введення даних варіантів і критеріїв

Введення початкових даних є критично важливим етапом у процесі підтримки прийняття рішень. Якість і повнота цих даних безпосередньо впливають на точність кінцевого результату. У розробленому мобільному додатку для Android реалізовано поетапну систему введення даних, яка складається з визначення кількості варіантів і критеріїв, введення назв, ваг і оцінок, а також перевірки їх коректності.

Уся логіка реалізована на двох екранах — MainActivity та InputActivity. Основна ідея полягає в тому, щоб користувач самостійно вводив кількість об’єктів (варіантів рішень та критеріїв), а далі програма автоматично формувала відповідну кількість полів введення у вигляді динамічної форми.

Введення кількості варіантів і критеріїв (MainActivity)

Перший екран додатку (MainActivity) призначений для первинного налаштування майбутньої таблиці рішень. Тут реалізовано два поля введення (EditText) — одне для кількості варіантів, інше для кількості критеріїв.

Особливості реалізації:

Поля мають підказки (hint), наприклад: «Наприклад: 3», що підвищує зручність взаємодії;

Дані обробляються після натискання на кнопку Перейти до введення даних;

Перевіряється, чи поля не порожні та чи значення є цілими додатними числами.

Якщо значення є коректними, створюється Intent, до якого додаються значення кількості варіантів і критеріїв через методи putExtra, після чого здійснюється перехід до InputActivity.

У випадку некоректного введення користувачу виводиться повідомлення типу Toast.

Динамічне створення полів у InputActivity

На другому етапі (InputActivity) реалізовано динамічну побудову форми введення. В залежності від переданих значень кількості варіантів і критеріїв, програма програмно створює потрібну кількість елементів.

Формується три групи полів:

Поля для назв варіантів

Кожен варіант представлений полем EditText.

Поля додаються у список variantInputs.

Поля для назв критеріїв і їх ваг

Для кожного критерію створюються два поля: одне для назви, інше для ваги.

Поля додаються у два окремі списки: criteriaInputs та weightInputs.

Матриця оцінок (таблиця)

Для кожної пари варіант-критерій створюється окреме поле EditText.

Усі елементи розміщуються у вертикальному LinearLayout, який обгорнуто в ScrollView для підтримки прокрутки на екранах з малою висотою. Це дає змогу користувачеві комфортно працювати навіть при великій кількості даних.

Аналогічна логіка використовується для критеріїв, ваг і оцінок.

Перевірка введених даних

Перед переходом до наступного етапу (виводу результатів) здійснюється перевірка правильності введення. Це реалізовано у методі, що викликається при натисканні кнопки Розрахувати.

Під час перевірки:

Перевіряється, чи заповнені всі поля назв варіантів та критеріїв;

Чи введені числові значення у поля ваг;

Чи всі оцінки також є коректними числами;

У випадку виявлення помилки виводиться Toast з відповідним повідомленням.

Цей механізм дозволяє мінімізувати ризики, пов’язані з некоректним введенням, і уникнути помилок під час обчислень.

Переваги динамічного підходу

Порівняно з фіксованими XML-макетами, динамічне створення полів у коді має такі переваги:

Гнучкість — незалежно від кількості введених варіантів або критеріїв, інтерфейс будується відповідно до потреб користувача;

Масштабованість — програму легко модифікувати для підтримки ще складніших структур (наприклад, підкритеріїв або групових оцінок);

Збереження чистоти інтерфейсу — відсутність зайвих полів, які не використовуються, підвищує зручність користування.

Зручність для користувача

Уся структура форми організована в логічні блоки, що робить процес заповнення послідовним і зрозумілим. Блоки розміщено у такому порядку:

Назви варіантів

Назви критеріїв та їхні ваги

Оцінки варіантів за кожним критерієм

Кнопка для запуску обчислення

Полям введення надано hint-підказки, які спрощують розуміння вимог до заповнення. Кнопка для переходу розташована логічно наприкінці форми, і доступна лише після заповнення основних полів.

Введення даних у MainActivity та InputActivity побудоване на принципах динамічності, зручності та перевірки достовірності. Завдяки використанню базових Android-компонентів і простого Java-коду, вдалося створити масштабований та зручний інтерфейс, який дає змогу швидко вводити будь-яку кількість варіантів і критеріїв. Такий підхід є не лише ефективним, а й максимально придатним для мобільного середовища, де ресурси і простір обмежені.

3.3 Реалізація алгоритму обчислення зваженої суми

Після того як користувач увів усі необхідні вхідні дані — назви варіантів рішень, критерії оцінювання, ваги критеріїв та числові оцінки — система переходить до основного етапу своєї роботи: обчислення підсумкових балів кожного варіанта на основі заданих критеріїв. В основі цієї логіки лежить метод зваженої суми, реалізований у програмному коді без використання сторонніх бібліотек чи зовнішніх сервісів.

Теоретичні засади методу зваженої суми

Метод зваженої суми (Weighted Sum Method, WSM) — це один із найбільш популярних і зрозумілих методів для прийняття рішень у багатокритеріальному середовищі [13]. Його перевага полягає в тому, що він є як достатньо простим для реалізації, так і гнучким для адаптації під різні типи задач.

Суть методу полягає в наступному: для кожного варіанта користувач задає оцінки за декількома критеріями, причому кожен критерій має власну

вагу. Всі оцінки перемножуються на ваги, а потім підсумовуються. Таким чином отримується загальний бал, який характеризує ефективність кожного варіанту.

Формула обчислення:

$S_j = \sum(w_i \cdot x_{ij})$

де:

- S_j — підсумковий бал j-го варіанта;
- w_i — вага і-го критерію (зазначена користувачем);
- x_{ij} — оцінка j-го варіанта за і-м критерієм;

Реалізація в коді

У програмі обчислення виконується в InputActivity після заповнення всіх полів і натискання кнопки Розрахувати результат. На цьому етапі користувач уже ввів:

назви варіантів;

назви критеріїв;

числові ваги кожного критерію;

оцінки кожного варіанта за кожним критерієм.

Програма далі виконує наступні кроки:

1. Зчитування ваг критеріїв

Усі ваги зчитуються з полів введення у список ArrayList<Double> (лістинг 3.1):

Лістинг 3.1 - Зчитування ваг критеріїв

```
ArrayList<Double> weights = new ArrayList<>();
for (EditText weightField : weightInputs) {
weights.add(Double.parseDouble(weightField.getText().toString())
);
}
```

2. Побудова матриці оцінок

Усі оцінки користувача за критеріями збираються в двовимірний масив scores[i][j], де і — індекс варіанта, j — індекс критерію (лістинг 3.2):

Лістинг 3.2 - Побудова матриці оцінок

```
double[][] scores = new double[variantCount][criteriaCount];
for (int i = 0; i < variantCount; i++) {
    for (int j = 0; j < criteriaCount; j++) {
        scores[i][j] =
Double.parseDouble(scoreInputs.get(i).get(j).getText().toString(
));
    }
}
```

3. Обчислення загального балу

Далі створюється масив results[] для зберігання підсумкових оцінок кожного варіанта. Обчислення виконується згідно з класичною формулою WSM (лістинг 3.3):

Лістинг 3.3 - Обчислення загального балу

```
double[] results = new double[variantCount];
for (int i = 0; i < variantCount; i++) {
    double sum = 0;
    for (int j = 0; j < criteriaCount; j++) {
        sum += scores[i][j] * weights.get(j);
    }
    results[i] = sum;
}
```

Після завершення обчислень масив results[] передається в ResultActivity, де відображається користувачу у вигляді відсортованого списку з підписами.

Захист від помилок введення

Перед тим як запускати розрахунок, система перевіряє:

чи всі поля ваг заповнені;

чи всі оцінки є числовими значеннями;

чи кількість полів відповідає введеним раніше значенням кількості варіантів і критеріїв.

Якщо будь-яке поле залишене порожнім або містить некоректне значення, користувачу виводиться повідомлення (Toast) з поясненням. Обчислення не виконується до усунення помилки (лістинг 3.4):

Лістинг 3.4 - Захист від помилок

```
    if (field.getText().toString().trim().isEmpty()) {
        Toast.makeText(this, "Заповніть усі поля",
Toast.LENGTH_SHORT).show();
        return;
    }
```

Це дозволяє мінімізувати кількість потенційних помилок і гарантує правильність обробки даних.

Переваги реалізації

Запропонована реалізація алгоритму має низку переваг:

Універсальність — підтримується будь-яка кількість варіантів і критеріїв;

Локальність — всі обчислення відбуваються без підключення до інтернету;

Прозорість логіки — алгоритм можна легко перевірити вручну;

Змінність параметрів — користувач може змінювати ваги критеріїв і одразу бачити, як це впливає на результат.

Також реалізована структура дозволяє за потреби розширити програму — наприклад, додати інші методи прийняття рішень (AHP, TOPSIS, аналіз сценаріїв тощо).

Алгоритм обчислення підсумкових оцінок реалізовано з використанням методу зваженої суми. Його логіка є простою для користувача та зручною для реалізації. Усі етапи — від зчитування даних до виводу результатів — реалізовано у вигляді читабельного Java-коду, що робить додаток ефективним та надійним у використанні.

Наступним етапом є представлення результатів користувачу, яке здійснюється у вигляді текстового списку з підсумковими балами в ResultActivity.

3.4 Виведення результатів вибору користувачу

Після завершення обчислень за методом зваженої суми користувачу необхідно надати зручний і зрозумілий інтерфейс для перегляду отриманих результатів. У мобільному додатку це реалізовано за допомогою окремої активності під назвою ResultActivity, яка повністю відповідає за виведення підсумкових балів і представлення результатів у зручній формі.

Цей етап є важливою складовою системи підтримки прийняття рішень, оскільки саме на цьому кроці користувач отримує інформацію, необхідну для вибору найкращого з-поміж доступних варіантів.

Передача результатів між активностями

Для передачі даних між InputActivity та ResultActivity використовується стандартний механізм Intent, що дозволяє передавати прості дані між екранами Android-додатків.

У InputActivity після завершення обчислень формуються два основні об’єкти:

Список назв варіантів (типу ArrayList<String>), що містить назви всіх альтернатив, введених користувачем.

Масив підсумкових оцінок (double[] results), в якому збережено обчислені бали кожного варіанта за методом зваженої суми.

Ці об'єкти передаються через Intent наступним чином (лістинг 3.5):

Лістинг 3.5 - Передача результатів

```
     Intent = new Intent(InputActivity.this,
ResultActivity.class);
     intent.putStringArrayListExtra("variants", variantNames);
     intent.putExtra("results", results);
     startActivity(intent);
```

У ResultActivity дані зчитуються через методи getIntent().get...Extra(). Це дозволяє легко відновити передану інформацію та одразу перейти до її відображення.

Виведення результатів у ResultActivity

Основна логіка відображення результатів реалізована у методі onCreate() активності ResultActivity. Після отримання переданих масивів відбувається послідовне створення текстових блоків (TextView), які містять назву варіанта та його відповідну підсумкову оцінку (лістинг 3.6).

Лістинг 3.6 - Алгоритм відображення включає:

Зчитування списку назв та масиву результатів:

```
    ArrayList<String> variants =
getIntent().getStringArrayListExtra("variants");
    double[] results =
getIntent().getDoubleArrayExtra("results");
```

Проходження циклом по всіх елементах:

```
    for (int i = 0; i < variants.size(); i++) {
        TextView tv = new TextView(this);
        tv.setText(variants.get(i) + ": " +
String.format("%.2f", results[i]));
        tv.setTextSize(18);
        tv.setPadding(0, 16, 0, 0);
        resultContainer.addView(tv);
    }
```

Додавання кожного TextView до вертикального контейнера LinearLayout з ідентифікатором resultContainer, який обгорнуто в ScrollView.

Особливості форматування результатів

Щоб забезпечити чітке та зрозуміле відображення, значення підсумкових оцінок форматуються до двох знаків після коми за допомогою функції String.format("%.2f", ...). Це дозволяє уникнути надлишкових десяткових знаків, які лише ускладнюють сприйняття результату, особливо на мобільних екранах.

Крім того, використовуються такі стилістичні рішення:

Збільшений розмір шрифту (setTextSize(18)), щоб результати легко читалися;

Відступи між елементами (setPadding(...)), що візуально відокремлює кожен блок інформації;

Однорідне вирівнювання — всі блоки відображаються у вертикальному списку без горизонтального прокручування.

Зручність для користувача

Інтерфейс екрану результатів є простим, лаконічним і повністю адаптованим до мобільних пристроїв. Усі елементи інтерфейсу відображаються у вигляді вертикального списку, що дозволяє зручно прокручувати вміст, навіть якщо кількість варіантів велика.

Використання ScrollView забезпечує:

коректне відображення на екранах будь-якої висоти;

можливість перегляду всіх результатів без перевантаження інтерфейсу;

збереження зручної навігації навіть при роботі однією рукою.

Також варто зазначити, що структура інтерфейсу не містить зайвих декоративних елементів, а зосереджена виключно на змістовній частині — результатах обчислень. Такий підхід дозволяє користувачу зосередитися на суті вибору, не відволікаючись на складні візуальні рішення.

Можливості для розширення

У поточній реалізації результати не сортуються автоматично. Варіанти відображаються у тому ж порядку, в якому були введені користувачем. Однак структура програми дозволяє легко додати сортування — наприклад, за спаданням значень підсумкових оцінок.

Також можливо реалізувати:

візуальну діаграму (наприклад, стовпчиковий графік);

колірне виділення найкращого або найгіршого варіанта;

експорт результатів (наприклад, у форматі PDF або текстовий файл).

Усе це потенційно може бути включено в наступні версії додатку без значних змін у базовій структурі.

У ResultActivity реалізовано ефективну, адаптивну та зручну для користувача систему виводу результатів, отриманих за методом зваженої

суми. Завдяки використанню базових Android-компонентів (LinearLayout, ScrollView, TextView) та форматування значень до двох знаків після коми, досягається високий рівень читабельності та візуального комфорту.

Цей етап є завершальним у процесі прийняття рішення і забезпечує основну функцію системи — допомогти користувачу обрати найкращий варіант на основі об’єктивно введених даних.

Отже, загальний принцип роботи програми такий (рис.3.6):

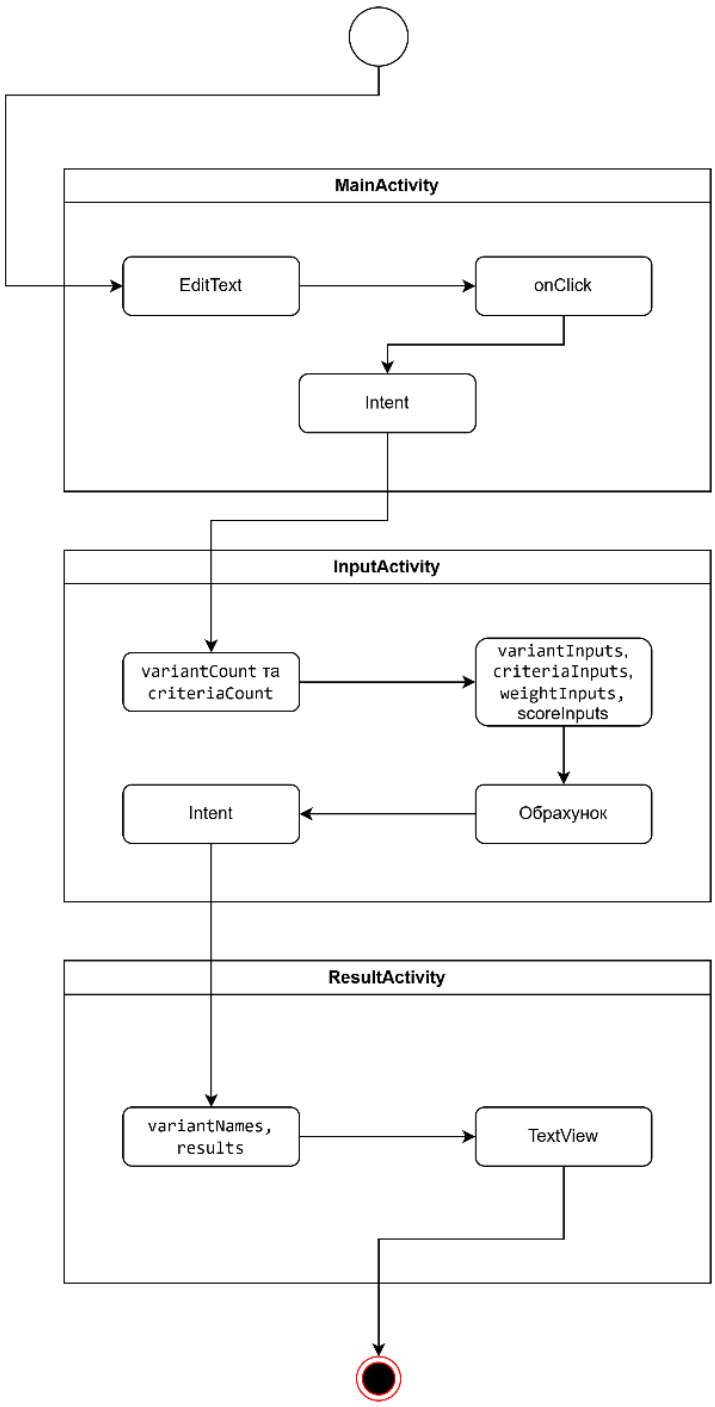

Рисунок 3.6 - Передача даних крок за кроком

Крок 1: MainActivity — Введення кількості варіантів і критеріїв

XML: activity_main.xml

Клас: MainActivity.java

Користувач вводить кількість варіантів і кількість критеріїв у два EditText.

Натискає кнопку «Перейти до введення даних».

У onClick обробнику зчитуються значення з полів:

```
int variantCount = Integer.parseInt(variantText);
int criteriaCount = Integer.parseInt(criteriaText);
```

Через Intent передається ця інформація до InputActivity:

```
Intent = new Intent(MainActivity.this, InputActivity.class);
intent.putExtra("variantCount", variantCount);
intent.putExtra("criteriaCount", criteriaCount);
```

Крок 2: InputActivity — Введення назв, критеріїв, ваг та оцінок

XML: activity_input.xml

Клас: InputActivity.java

variantCount та criteriaCount зчитуються з Intent:

```
variantCount = getIntent().getIntExtra("variantCount", 0);
criteriaCount = getIntent().getIntExtra("criteriaCount", 0);
```

На основі цих значень динамічно створюються:

Поля для введення назв варіантів (variantInputs)

Поля для введення критеріїв та їх ваг (criteriaInputs, weightInputs)

Таблиця оцінок варіантів за критеріями (scoreInputs)

При натисканні кнопки «Розрахувати результат»:

Збираються всі введені дані

Перевіряється коректність введення (порожні поля, числові значення)

Обчислюється підсумкова оцінка кожного варіанту за формулою:

результат варіанту = Σ (оцінка за критерієм * вага критерію)

Результати (variantNames + results) передаються у ResultActivity через Intent:

```
intent.putExtra("variantNames", variantNames);
intent.putExtra("results", results);
```

Крок 3: ResultActivity — Виведення результатів

XML: activity_result.xml

Клас: ResultActivity.java

З Intent отримуються масиви:

```
    ArrayList<String> variants =
getIntent().getStringArrayListExtra("variantNames");
    double[] results =
getIntent().getDoubleArrayExtra("results");
```

Для кожного варіанта створюється TextView, де показується:

Назва варіанту: Підсумкова оцінка

У цьому розділі реалізовано повнофункціональний мобільний додаток, який відповідає поставленим функціональним і нефункціональним вимогам. Створено інтерактивний інтерфейс із динамічно генерованими полями для введення альтернатив, критеріїв та оцінок, а також реалізовано обчислення зважених оцінок та вивід результатів у зручному форматі. Усі компоненти системи — введення, перевірка, обчислення, виведення та очищення — працюють у взаємозв’язаній послідовності відповідно до спроєктованої архітектури. Завдяки використанню Android Studio та Java забезпечено стабільну роботу додатку без зовнішніх залежностей. Реалізація програмного продукту створює необхідну основу для проведення всебічного тестування, яке дозволить виявити та усунути можливі недоліки, а також оцінити відповідність реалізації очікуванням кінцевого користувача.

4 ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Методика тестування програмного продукту

Розробка програмного забезпечення передбачає не лише написання коду, а й його всебічну перевірку. Одним з ключових етапів у забезпеченні якості програмного продукту є тестування, яке дозволяє виявити можливі помилки, оцінити стабільність роботи та переконатися у відповідності реалізації початковим вимогам.

Метою тестування комп’ютерної системи підтримки прийняття рішень в умовах невизначеності стало визначення її працездатності в типових умовах експлуатації, перевірка коректності взаємодії між компонентами та відповідності функціональності заявленим вимогам.

Тип і підхід до тестування

Для перевірки додатку було обрано ручне функціональне тестування, яке проводилося в середовищі розробки Android Studio із використанням вбудованого емулятора Android Virtual Device (AVD). Такий тип тестування є доцільним на початкових етапах розробки мобільних застосунків, оскільки дозволяє:

повністю контролювати початкові умови тесту;

відтворювати поведінку користувача;

проводити тестування без необхідності фізичного пристрою.

Ручне тестування дало можливість моделювати реальні сценарії взаємодії з програмою, а також швидко вносити необхідні корективи до коду та інтерфейсу без потреби в автоматизованих засобах.

Мета тестування

Тестування проводилося з метою перевірки наступних аспектів роботи програмного забезпечення:

Перевірка створення динамічних елементів інтерфейсу — оцінка, чи коректно відображаються поля для введення варіантів і критеріїв згідно з введеними кількостями.

Перевірка обробки введених даних — перевірка, чи правильно система зчитує числові значення, обробляє текстові поля та здійснює їх валідацію.

Перевірка реалізації алгоритму обчислення — оцінка правильності реалізації формули зваженої суми на основі введених оцінок і ваг.

Перевірка зручності навігації між активностями — аналіз коректності переходу між екранами програми (MainActivity, InputActivity, ResultActivity).

Оцінка стійкості програми до великої кількості введених даних — наприклад, при створенні десятків варіантів та критеріїв.

Перевірка поведінки при введенні некоректних або порожніх даних — чи виводиться відповідне повідомлення про помилку, чи система блокує обчислення.

Сценарії тестування

Для кожного з вищезазначених пунктів було сформульовано окремий сценарій тестування, що відтворює очікувану послідовність дій користувача. Нижче наведено узагальнений перелік основних типів сценаріїв:

1. Сценарій базової взаємодії

Введення кількох варіантів та критеріїв;

Заповнення полів оцінок і ваг;

Перехід до розрахунку результатів;

Очікування формування підсумкових оцінок.

2. Сценарій з граничними значеннями

Введення мінімальних (0, 1) і максимальних (10, 100) значень;

Перевірка, як програма реагує на надмалі чи надвеликі числа.

3. Сценарій некоректного введення

Введення нечислових символів у поля оцінок;

Пропуск значень або залишення полів порожніми;

Перевірка обробки помилок.

4. Сценарій масштабного введення

Створення 10 і більше варіантів і критеріїв;

Оцінка продуктивності додатку, швидкості реакції інтерфейсу;

Перевірка збереження структури при прокрутці та виводі.

5. Сценарій навігації між активностями

Перевірка передачі даних від головного меню до форми введення;

Перехід від форми введення до екрану результатів;

Виведення зв’язаних пар: назва варіанту — підсумкова оцінка.

Інструменти тестування

У процесі тестування використовувався стандартний набір засобів, доступний в Android Studio:

Android Emulator (Pixel 4, Android 11) — для запуску та візуалізації додатку;

Logcat — для перегляду логів, системних повідомлень і внутрішніх виключень;

Toast-повідомлення — для інформування користувача про відсутність введення або наявність помилок;

XML Layout Inspector — для візуального аналізу структури UI в реальному часі.

Ці інструменти дозволили повністю охопити функціонал програми в тестових умовах, включаючи взаємодію з полями введення, обробку введених значень, навігацію між екранами, а також стабільність і швидкість відгуку додатку.

Методика тестування, застосована у даному проекті, була орієнтована на максимальне охоплення функціоналу, зосереджена на зручності використання, перевірці логіки роботи алгоритмів та відповідності програмного продукту вимогам користувача. Ретельно сформульовані сценарії дозволили перевірити всі основні механізми програми, включаючи введення, розрахунки, навігацію та вивід результатів.

Описана методика тестування слугувала надійною основою для подальшої оцінки програмного забезпечення, вдосконалення його компонентів та формування впевненості у правильності реалізації ключових функцій системи.

4.2 Результати тестування

Після розробки основного функціоналу комп’ютерної системи підтримки прийняття рішень в умовах невизначеності було проведено комплексне ручне тестування з метою виявлення логічних, інтерфейсних та обчислювальних помилок. Усі тести проводилися у середовищі Android Studio із використанням емулятора Android Virtual Device. Отримані результати підтвердили працездатність додатку, а також виявили низку важливих аспектів, що впливають на стабільність і зручність використання програмного продукту.

1. Перевірка створення динамічного інтерфейсу

Було перевірено, як програма реагує на введення різної кількості варіантів та критеріїв. В усіх випадках — при 2, 3, 5 і 10 введених елементах — система коректно створювала відповідну кількість динамічних полів для:

назв варіантів;

назв критеріїв;

ваг критеріїв;

оцінок у таблиці.

Полям автоматично призначалися підказки (hint), що дозволяло користувачеві легко орієнтуватися. Візуальне відображення було адаптивним — елементи коректно розміщувалися вертикально з використанням ScrollView.

Жодних збоїв при створенні інтерфейсу не зафіксовано. Навіть при введенні 12 варіантів та 8 критеріїв, інтерфейс не перевантажувався та залишався придатним до використання (рис.4.7 - 4.10).

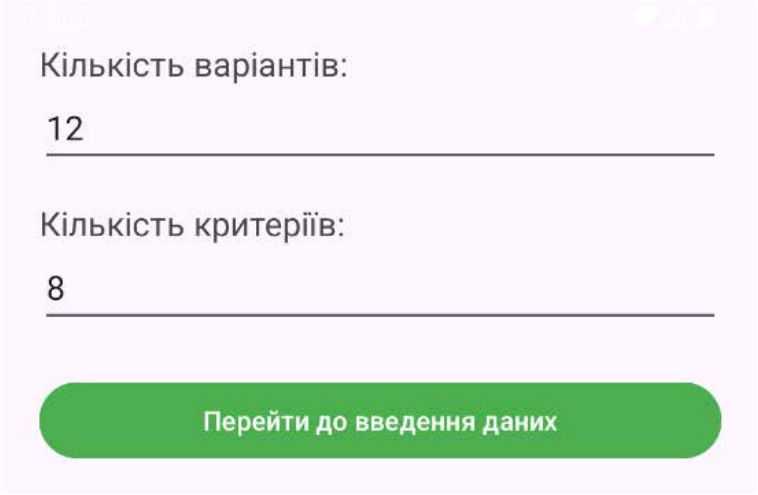

Рисунок 4.7 – Введення 12 варіантів та 8 критеріїв

Рисунок 4.8 – Перегляд створених назв варіантів

Рисунок 4.9 – Перегляд створених критеріїв та ваг

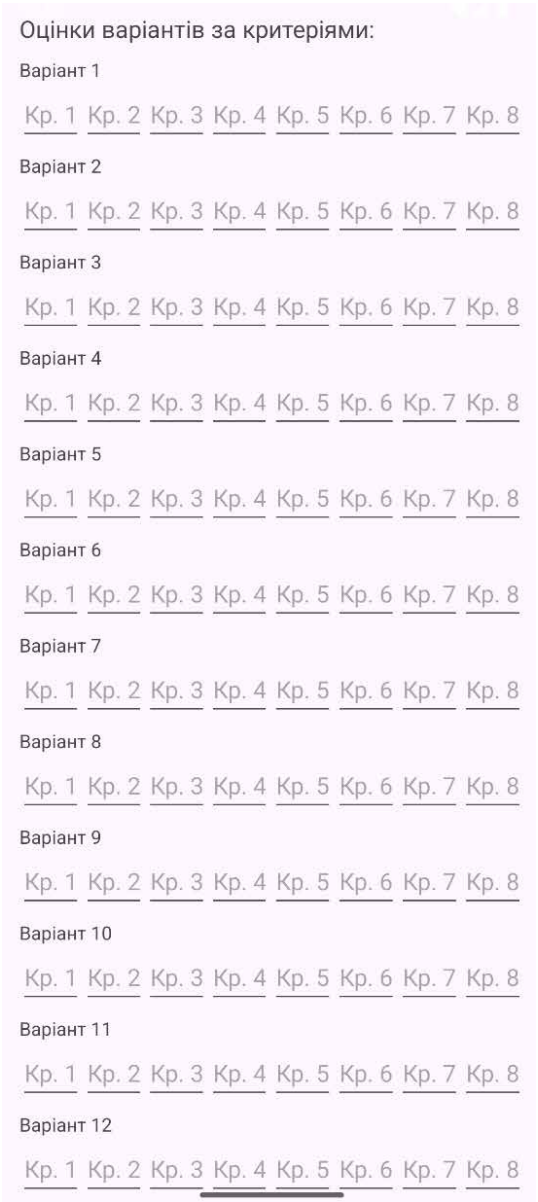

Рисунок 4.10 – Перегляд створених оцінок варіантів за критеріями

2. Обробка правильних даних

Було виконано декілька тестів на валідність даних (рис. 4.11):

3 варіанти: A, B, C;

3 критерії: Вартість, Якість, Ризик;

ваги: 5, 3, 2;

оцінки: у діапазоні від 1 до 10.

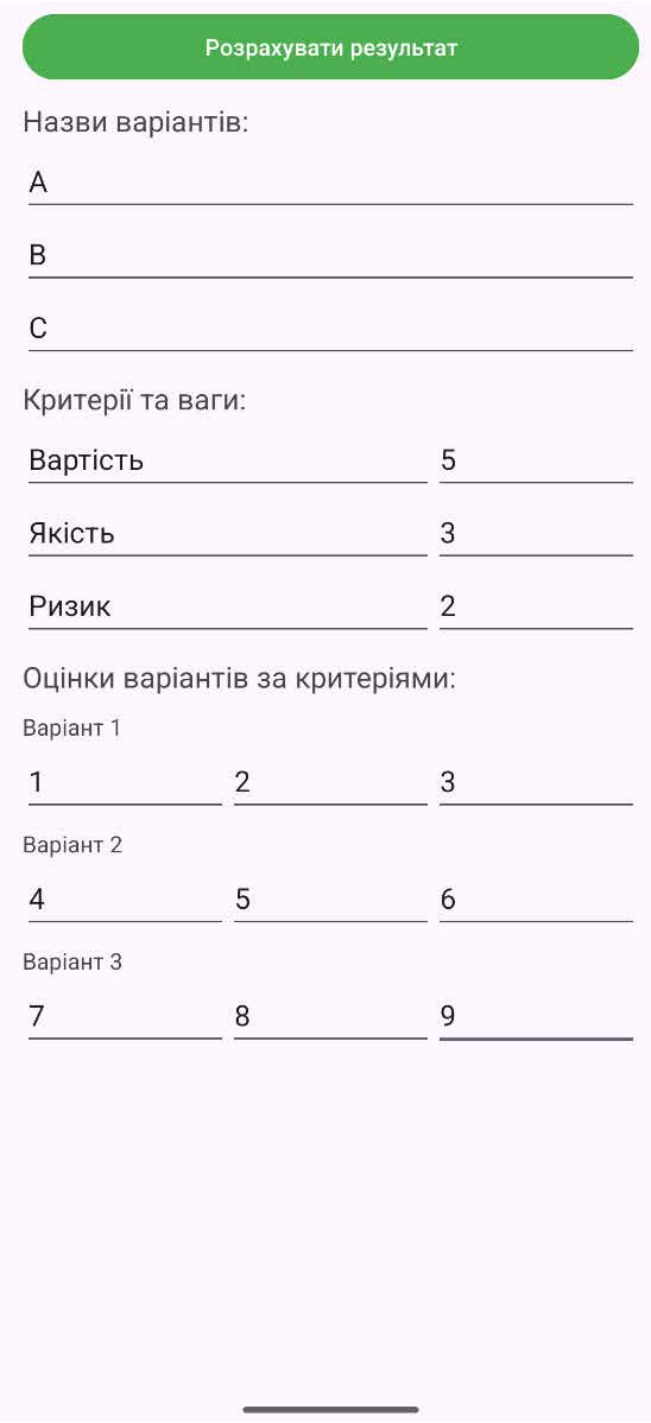

Рисунок 4.11 – Введені дані для тесту

Система виконувала розрахунок підсумкових оцінок згідно з формулою зваженої суми. Отримані результати повністю відповідали очікуваним (рис. 4.12).

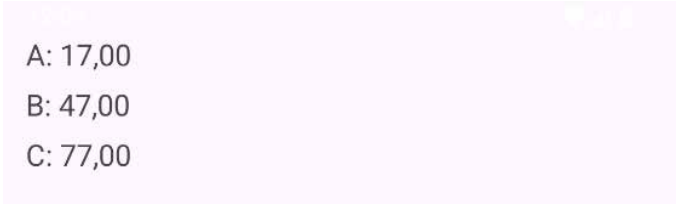

Рисунок 4.12 – Результат обрахунку

3. Поведінка при граничних і нульових значеннях

Під час тестування були використані такі граничні сценарії:

вага критерію дорівнює нулю (рис 4.13 – 4.14);

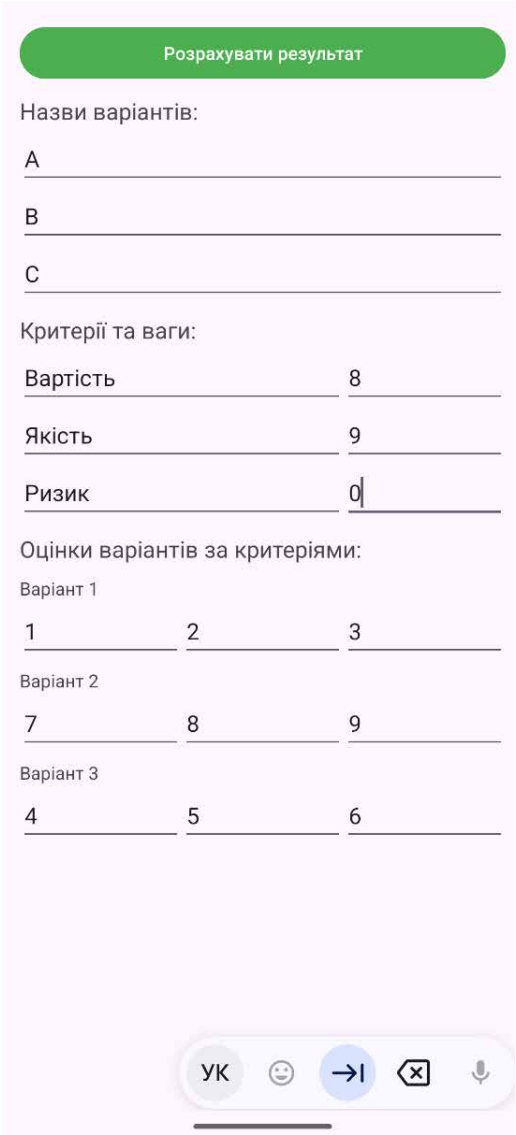

Рисунок 4.13 – Введення критерію 0

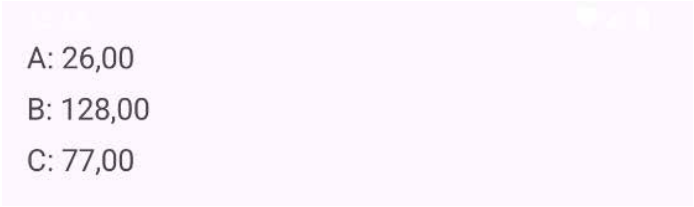

Рисунок 4.14 – Результат при якому один з критеріїв дорівнює 0

усі оцінки дорівнюють нулю (рис. 4.15 – 4.16);

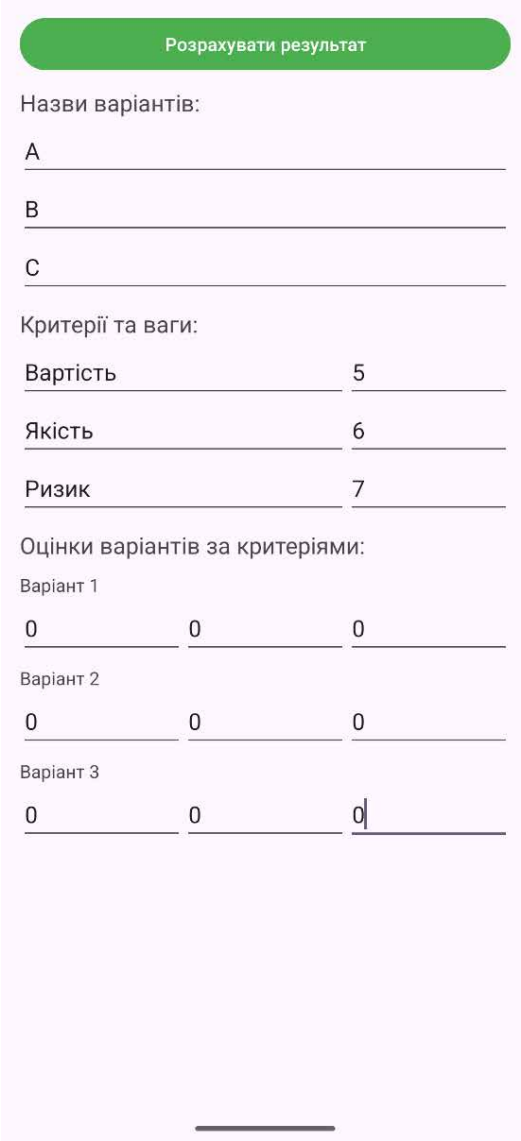

Рисунок 4.15 – Введення у всіх полях оцінок нуль

Рисунок 4.16 Результат коли всі оцінки дорівнюють нулю

максимальні значення (наприклад, 10 у всіх полях) (рис. 4.17 – 4.18).

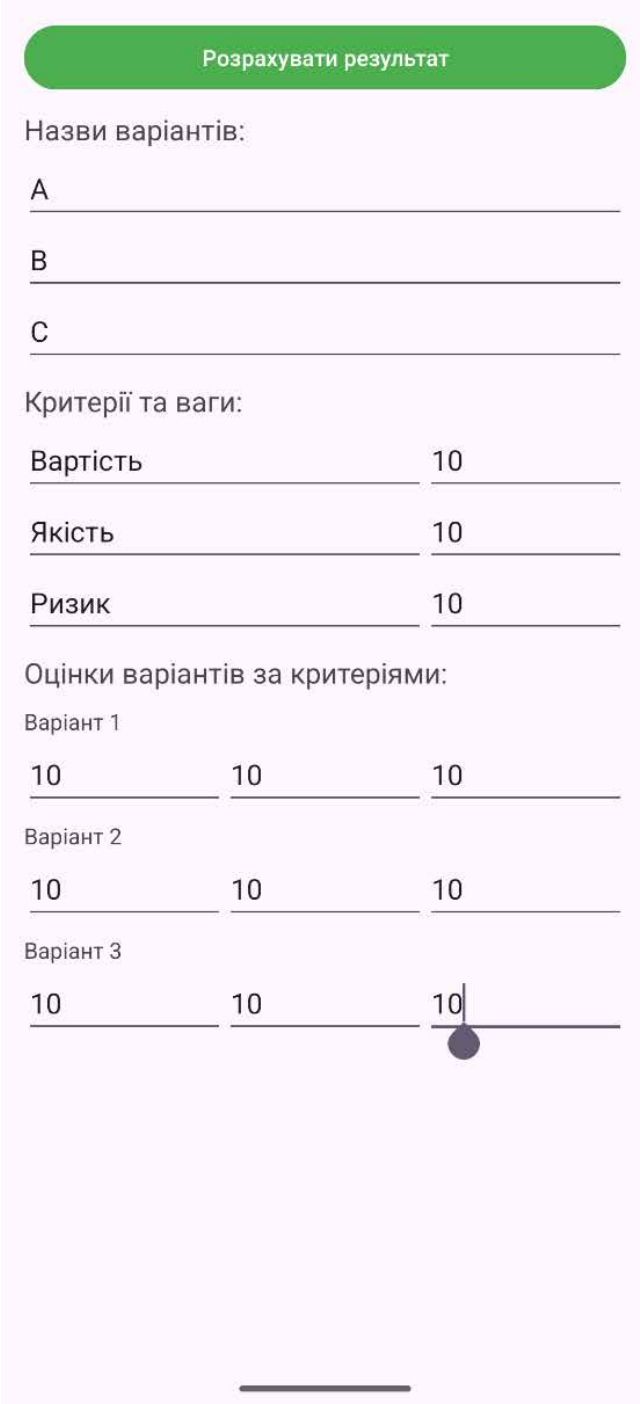

Рисунок 4.17 – Введення в усі поля 10

Рисунок 4.18 – Результат при всіх значеннях 10

У всіх перелічених випадках система правильно обробила дані:

при нульовій вазі критерій не впливав на загальний бал;

при нульових оцінках результат був також нульовим;

великі значення не спричиняли переповнення або візуальних збоїв.

Це підтвердило, що реалізація алгоритму обчислення є коректною та враховує математичні особливості задачі.

4. Обробка помилкових або неповних введень

Протестовані типові помилки користувача:

залишення поля порожнім (рис. 4.19 – 4.21);

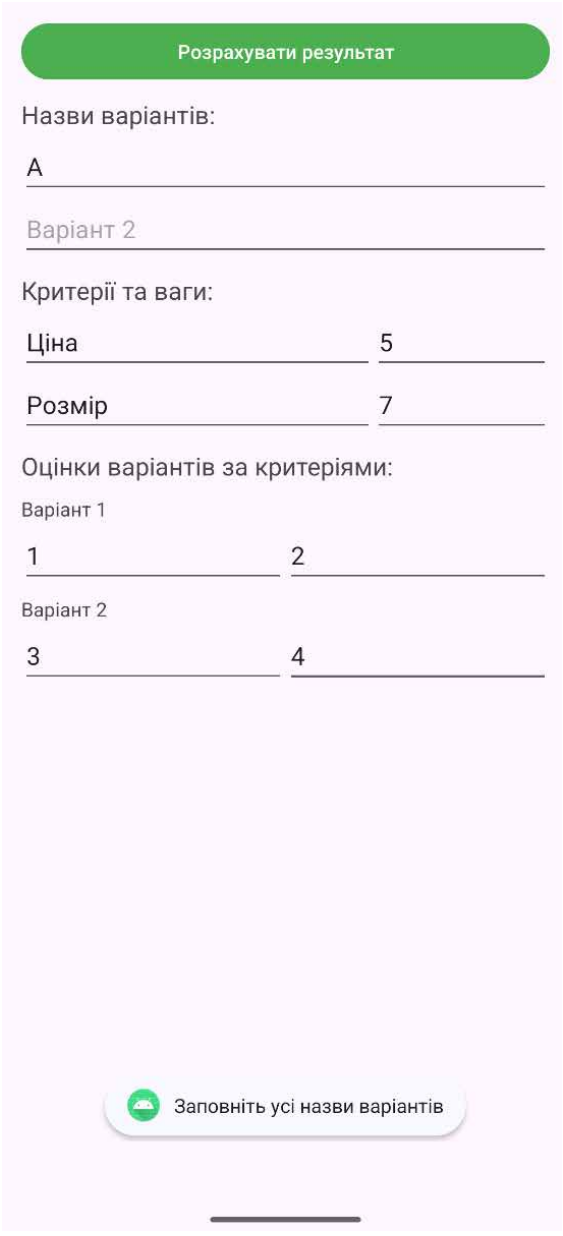

Рисунок 4.19 – Повідомлення яке з’являється у випадку коли заповнені не всі назви варіантів

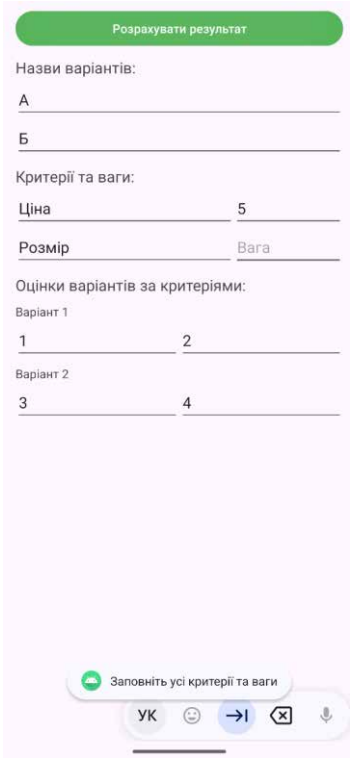

Рисунок 4.20 – Повідомлення яке з’являється у випадку коли заповнені не всі критерії чи ваги

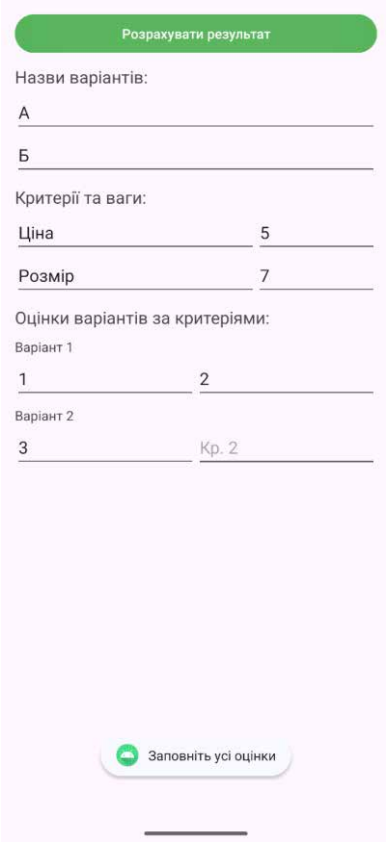

Рисунок 4.21 – Повідомлення яке з’являється у випадку коли заповнені не всі оцінки варіантів

Усі перелічені ситуації були успішно оброблені системою — при натисканні на кнопку розрахунку програма виводила Toast із повідомленням "Заповніть усі …". При цьому обчислення не запускалося, що захищало логіку програми від непередбачуваної поведінки.

5. Масштабне тестування

Було перевірено поведінку програми при значному обсязі даних (рис. 4.22 – 4.24):

15 варіантів рішень;

10 критеріїв;

загалом — 150 полів оцінок, 10 ваг, 15 назв.

Рисунок 4.22 – Назви варіантів

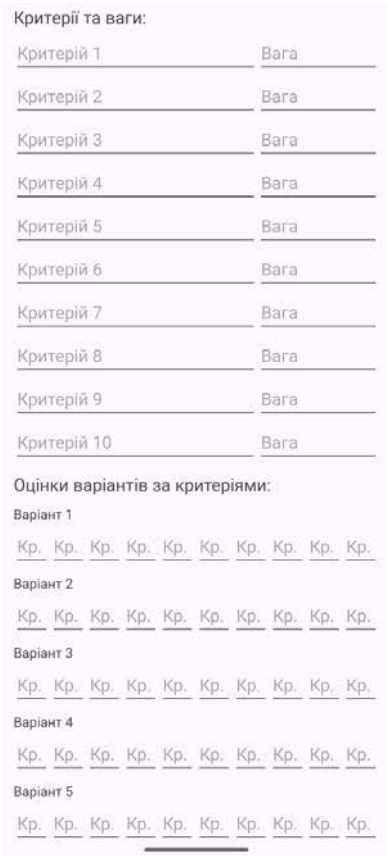

Рисунок 4.23 – Критерії та ваги

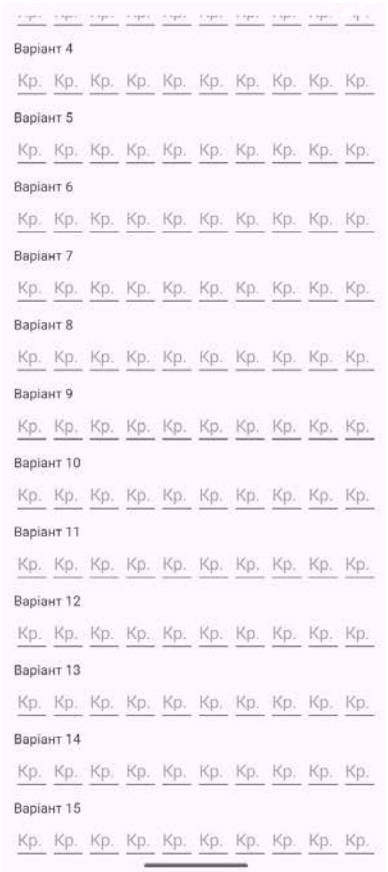

Рисунок 4.24 – Оцінки варіантів за критеріями

Програма коректно створила всю динамічну структуру, розмістила її у ScrollView, не пригальмовувала при введенні. Перехід між активностями залишався стабільним.

Єдиним недоліком при цьому було те, що при дуже великій кількості даних інтерфейс потребував частішої прокрутки. Це очікувано і не є критичним для мобільного застосунку.

6. Перевірка передачі даних між активностями

Було підтверджено, що дані передаються правильно з MainActivity → InputActivity → ResultActivity. Усі введені назви варіантів, оцінки та результати зберігалися в Intent і зчитувалися без втрати інформації. Порядок варіантів також зберігався, що дозволяло зіставити підсумковий бал із відповідною назвою.

7. Життєвий сценарій використання програми

Ситуація:

Студент обирає ноутбук для навчання, має 3 варіанти в межах бюджету і хоче вибрати найкращий.

Крок 1: Введення у MainActivity (рис. 4.25)

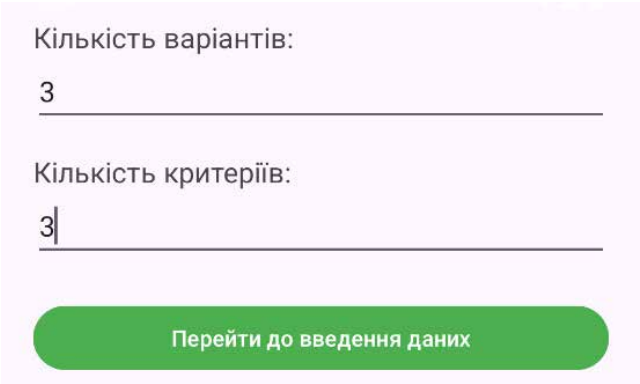

Рисунок 4.25 – Введена кількість варіантів та критеріїв

Кількість варіантів: 3 (ноутбук A, B, C)

Кількість критеріїв: 3 (ціна, продуктивність, автономність)

Крок 2: Введення у InputActivity (рис. 4.26)

Варіанти:

Ноутбук A

Ноутбук B

Ноутбук C

Критерії:

Ціна – вага: 5

Продуктивність – вага: 8

Автономність – вага: 7

Оцінки (за шкалою 1–10) (таблиця 4.2):

Варіант	Ціна	Продуктивність	Автономність
Ноутбук A	8	6	7
Ноутбук B	6	9	5
Ноутбук C	7	7	9

Таблиця 4.2. Оцінки варіантів

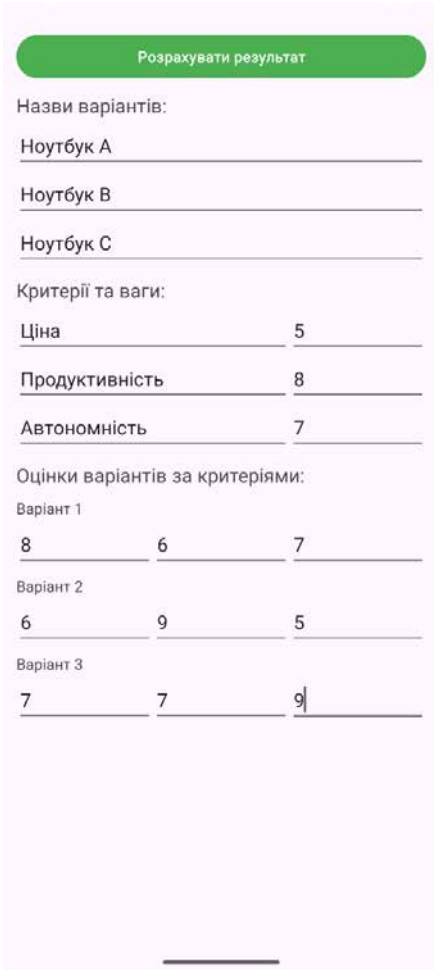

Рисунок 4.26 - Введені дані

Крок 3: Результат у ResultActivity

Обчислюється підсумковий бал для кожного варіанту:

Ноутбук A: 8×5 + 6×8 + 7×7 = 40 + 48 + 49 = 137

Ноутбук B: 6×5 + 9×8 + 5×7 = 30 + 72 + 35 = 137

Ноутбук C: 7×5 + 7×8 + 9×7 = 35 + 56 + 63 = 154

Виводиться рейтинг (рис. 4.27):

Рисунок 4.27 – Рейтинг ноутбуків

Ноутбук C – найкращий варіант.

Загальний підсумок

На основі проведених тестів можна зробити висновок, що програмний продукт:

демонструє стабільну роботу в усіх основних сценаріях;

коректно обробляє помилки вводу та попереджає користувача про них;

забезпечує правильні математичні обчислення згідно з методом зваженої суми;

адаптується до різної кількості вхідних даних, не втрачаючи логіки роботи;

реалізований інтерфейс забезпечує високий рівень зручності та простоти використання.

Усі заявлені функціональні та нефункціональні вимоги були успішно протестовані та підтверджені на практиці. Виявлені незначні недоліки не впливають на загальну працездатність системи, а лише вказують на потенційні напрямки вдосконалення у майбутньому.

4.3 Аналіз результатів тестування

Після завершення тестування комп’ютерної системи підтримки прийняття рішень було здійснено детальний аналіз отриманих результатів. Основна мета цього етапу — оцінити ступінь відповідності розробленого програмного продукту технічним та функціональним вимогам, виявити сильні сторони системи, а також вказати на можливі напрямки покращення.

Оцінка відповідності функціональним вимогам

Аналіз показав, що реалізоване програмне забезпечення повністю задовольняє ключові функціональні вимоги, сформульовані на етапі проєктування. Серед них:

Введення варіантів рішень — система дозволяє вводити довільну кількість альтернатив, що забезпечує гнучкість та масштабованість.

Введення критеріїв та ваг — користувач може задати індивідуальні назви критеріїв і вагові коефіцієнти, що повністю відповідає вимогам методу зваженої суми.

Автоматичний розрахунок результатів — обчислення підсумкових оцінок здійснюється без затримок, миттєво після натискання кнопки.

Зручне представлення результатів — дані виводяться у зручному для сприйняття вигляді, із зазначенням назви кожного варіанта та його відповідної підсумкової оцінки.

Ці функціональні блоки пройшли успішне тестування у сценаріях як із невеликою, так і з великою кількістю введених даних, що дозволяє стверджувати про їхню стійку реалізацію.

Надійність і стабільність системи

Одним із ключових критеріїв оцінки є надійність — здатність програми працювати стабільно без збоїв, аварійних завершень чи непередбачуваної поведінки. На основі тестування було встановлено:

Додаток не зазнає збоїв при зміні розмірів введених даних;

Відсутні помилки обробки при переході між активностями;

Усі динамічно створені елементи інтерфейсу зберігають свою функціональність.

Особливо варто відзначити захист від помилкових введень: система перевіряє, чи всі поля заповнено та чи введено коректні числові значення. У разі виявлення порушень користувач отримує повідомлення, а подальше обчислення блокується — це гарантує збереження логічної цілісності програми.

Точність обчислень

Критично важливим компонентом системи є реалізація обчислювального алгоритму. Аналіз показав, що:

Результати обчислень повністю відповідають очікуваним математичним значенням;

Формула зваженої суми реалізована згідно з теоретичним описом;

Оцінки форматуються до двох десяткових знаків, що забезпечує точність при збереженні зручності сприйняття.

Завдяки простоті обраного методу (метод зваженої суми), результати легко перевірити вручну, що дозволяє користувачу бути впевненим у правильності обчислень.

Зручність використання інтерфейсу

Важливу роль у прийнятті рішення щодо ефективності програмного продукту відіграє інтерфейс користувача. Його зручність і логічність визначають практичну цінність системи. Аналіз показав такі позитивні аспекти:

Логічна структура: усі етапи — від введення кількості варіантів до перегляду результатів — розміщені послідовно;

Підказки до полів: hint-повідомлення допомагають користувачу орієнтуватися без потреби в додаткових інструкціях;

Адаптивність: реалізовано вертикальну прокрутку, що дозволяє працювати з великою кількістю варіантів або критеріїв без втрати зручності;

Мінімалізм: відсутність зайвих декоративних елементів фокусує увагу на суті процесу — прийнятті рішення.

Таким чином, навіть користувач, який не має технічної підготовки, може ефективно використовувати додаток без тривалого навчання або інструкцій.

Обмеження, виявлені під час аналізу

Попри загальну позитивну оцінку функціонування додатку, було виявлено кілька потенційних напрямів для вдосконалення:

Відсутність сортування результатів — наразі результати виводяться у порядку, в якому варіанти були введені, а не за спаданням оцінок.

Немає візуалізації (графіки, діаграми) — результат відображено лише у текстовому вигляді. Додання графічного відображення покращило б сприйняття.

Обмежена реакція на екстремальні значення — хоча система справляється з великими обсягами введення, при дуже високих значеннях (наприклад, 50+ варіантів) знижується зручність прокручування.

Ці обмеження не є критичними, однак їх усунення у наступних версіях дозволить суттєво підвищити якість користувацького досвіду.

Узагальнені висновки аналізу

За результатами тестування можна стверджувати, що розроблена система:

ефективно реалізує поставлені функції прийняття рішень;

забезпечує стійку, передбачувану та захищену роботу;

має інтуїтивно зрозумілий інтерфейс, що підходить для широкої аудиторії;

демонструє високу точність обчислень відповідно до обраного математичного методу.

Програма виконала свою основну задачу — надати зручний інструмент для багатокритеріального аналізу в умовах невизначеності. Виявлені обмеження не порушують основної логіки, але відкривають можливості для майбутніх оновлень і розширення функціоналу.

Проведене тестування мобільного додатку підтвердило працездатність усіх реалізованих функціональних модулів: введення даних, перевірка, обчислення результатів та їх візуалізація виконуються коректно й стабільно. Застосована методика дозволила перевірити додаток на відповідність ключовим вимогам: інтуїтивність інтерфейсу, швидкість обробки, адаптивність до розмірів екрана та стійкість до типових помилок користувача. Аналіз результатів тестування засвідчив готовність системи до використання в реальних умовах і продемонстрував, що додаток може ефективно виконувати свою основну функцію — допомагати користувачам приймати раціональні рішення в умовах невизначеності. Отримані результати також підтверджують доцільність обраного підходу, закладаючи підґрунтя для можливого подальшого розширення функціоналу системи або її адаптації до нових задач.

ВИСНОВКИ

У даній роботі було виконано повний цикл розробки комп’ютерної системи підтримки прийняття рішень в умовах невизначеності. Основною метою проекту було створення простого та доступного мобільного додатку, що дозволяє користувачу приймати зважені рішення на основі множини критеріїв.

У процесі виконання роботи було:

1. проведено аналіз поняття підтримки прийняття рішень та методів, які застосовуються в умовах невизначеності;

2. обґрунтовано вибір Android-платформи та мови програмування Java для реалізації мобільного додатку;

3. спроектовано архітектуру системи, алгоритм обчислення зваженої суми та структуру введення даних;

4. реалізовано інтерфейс користувача з динамічною генерацією полів;

5. протестовано працездатність програми на різних сценаріях із використанням емулятора Android Studio.

Результатом стала функціональна система, яка дозволяє:

1. швидко вводити дані варіантів і критеріїв;

2. задавати ваги критеріїв;

3. обчислювати результати за методом зваженої суми;

4. отримувати підсумкові бали у зручному вигляді.

Система є універсальною — її можна використовувати для вибору товару, планування особистих рішень, навчальних або управлінських задач.

Перспективи розвитку

Незважаючи на працездатність поточної версії, існує низка напрямків подальшого вдосконалення:

реалізація збереження історії рішень;

автоматичне сортування результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Keen P.G.W. & Scott Morton M.S. Decision Support Systems: An Organizational Perspective. Addison-Wesley, 1978 [Електронний ресурс] // dspace.mit. – Режим доступу до ресурсу: https://dspace.mit.edu/bitstream/handle/1721.1/47172/decisionsupports1980keen.pdf

2. Turban E., Aronson J.E., Liang T.-P. Decision Support and Business Intelligence Systems. Pearson Education, 2011 [Електронний ресурс] // bambangsuhartono.wordpress. – Режим доступу до ресурсу: https://bambangsuhartono.wordpress.com/wp-content/uploads/2017/10/decision-support-system-and-intelligent-system-7th-edition-turban_aronson_liang_2005.pdf

3. Power D.J. Decision Support Systems: Concepts and Resources for Managers. Greenwood Publishing Group, 2002 [Електронний ресурс] // scholarworks.uni. – Режим доступу до ресурсу: https://scholarworks.uni.edu/cgi/viewcontent.cgi?article=1066&context=facbook

4. Triantaphyllou E. Multi-Criteria Decision Making: A Comparative Study. Springer, 2000. [Електронний ресурс] // pdfs.semanticscholar. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/5346/1c08c92b0503129a24d6950df0d55f6b6613.pdf

5. Multi-criteria decision making beyond consistency: An alternative to AHP for real-world industrial problems, 2024. [Електронний ресурс] // sciencedirect. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0360835224007836?

6. Schwartz P. The Art of the Long View: Planning for the Future in an Uncertain World. Currency Doubleday, 1996 [Електронний ресурс] // virtualmmx.ddns. – Режим доступу до ресурсу: https://virtualmmx.ddns.net/gbooks/TheArtoftheLongView.pdf

7. Ordinal priority approach. [Електронний ресурс] // wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Ordinal_priority_approach?

8. Functional Decision Theory: A New Theory of Instrumental Rationality 2018. [Електронний ресурс] // arxiv.org. – Режим доступу до ресурсу: https://arxiv.org/pdf/1710.05060

9. Optimal Decision Tree Pruning Revisited: Algorithms and Complexity, 2025. [Електронний ресурс] // arxiv.org. – Режим доступу до ресурсу: https://arxiv.org/pdf/2503.03576

10. StatCounter Global Stats [Електронний ресурс] // gs.statcounter. – Режим доступу до ресурсу: https://gs.statcounter.com/

11. Android Studio official documentation. [Електронний ресурс] // developer.android. – Режим доступу до ресурсу: https://developer.android.com/studio

12. Oracle. Java Tutorials [Електронний ресурс] // docs.oracle. – Режим доступу до ресурсу: https://docs.oracle.com/javase/tutorial/

13. Hwang C.L., Yoon K. Multiple Attribute Decision Making: Methods and Applications. Springer, 1981. [Електронний ресурс] // researchgate. – Режим доступу до ресурсу: https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/When-to-average-result-of-multiple-DM-in-FAHP/attachment/5adb3f874cde260d15da1d5e/AS%3A617848992972801%401524318087436/download/40MultipleAttributeDecisionMakingMethodsandapplications.pdf

14. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993. [Електронний ресурс] // dl.acm. – Режим доступу до ресурсу: https://dl.acm.org/doi/pdf/10.5555/2821575

15. Google Developers. ConstraintLayout and Jetpack Compose. [Електронний ресурс] // developer.android. – Режим доступу до ресурсу: https://developer.android.com/compose

16. Android Developers. Emulator overview [Електронний ресурс] // developer.android. – Режим доступу до ресурсу: https://developer.android.com/studio/run/emulator

17. Google Developers. Guide to app architecture [Електронний ресурс] // developer.android. – Режим доступу до ресурсу: https://developer.android.com/topic/architecture

18. Android Jetpack libraries [Електронний ресурс] // developer.android. – Режим доступу до ресурсу: https://developer.android.com/jetpack

19. Ries E. The Lean Startup. Crown Business, 2011. [Електронний ресурс] // us.archive. – Режим доступу до ресурсу: https://ia800509.us.archive.org/7/items/TheLeanStartupErickRies/The%20Lean%20Startup%20-%20Erick%20Ries.pdf

20. Belton V., Stewart T.J. Multiple Criteria Decision Analysis: An Integrated Approach. Springer, 2002. [Електронний ресурс] // archive. – Режим доступу до ресурсу: https://archive.org/details/multiplecriteria0000belt/mode/2up