**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інтелектуальних інформаційних систем**

ДОПУЩЕНО ДО ЗАХИСТУ

В. о. завідувача кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

# ІНФОРМАЦІЙНА СИСТЕМА З ОЦІНКИ ГОТОВНОСТІ СТАРТАПУ ДО МАСШТАБУВАННЯ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

***Здобувач*** _____ Ірина ШАВРІНА

«___»_____ 2025 р.

***Керівник*** канд. фіз.-мат. наук, доцент _____ Інесса КУЛАКОВСЬКА

«___»_____ 2025 р.

**Миколаїв – 2025**

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

| | |
|---|---|
| Факультет | Комп'ютерних наук |
| Кафедра | Інтелектуальних інформаційних систем |
| Рівень вищої освіти | Другий (магістерський) |
| Освітній ступень | Магістр |
| Спеціальність | 122 Комп'ютерні науки |
| Освітня програма | Інтелектуальні інформаційні системи |

ЗАТВЕРДЖУЮ

В. о. завідувача кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 2025 р.

**ЗАВДАННЯ**
**на кваліфікаційну роботу здобувача**

**Шавріної Ірини Олександрівни**
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «ІНФОРМАЦІЙНА СИСТЕМА З ОЦІНКИ ГОТОВНОСТІ СТАРТАПУ ДО МАСШТАБУВАННЯ»

Керівник роботи: Кулаковська Інесса Василівна, доцент кафедри ІІС, канд. фіз.-мат. наук, доцент
*(прізвище, ім'я, по батькові, посада, науковий ступінь, вчене звання)*

Затверджена наказом ЧНУ ім. Петра Могили від «07» 07 2025 р. № 184.

2. Строк представлення кваліфікаційної роботи «16» 12 2025 р.

3. Очікуваний результат роботи та початкові дані: методології оцінки життєвого циклу та зрілості стартапів; набори критеріїв оцінки інвестиційної привабливості проєктів; математичні методи підтримки прийняття рішень (AHP, TOPSIS); засоби розробки програмного забезпечення (Python, Telegram API, SQLite); тестові набори даних про стан стартапів різних галузей.
Очікуваний результат: інформаційна система підтримки прийняття рішень для оцінки готовності стартапу до масштабування.

4. Перелік питань, що підлягають розробці: аналіз предметної області, визначення особливостей життєвого циклу стартапів та ключових ризиків етапу масштабування; огляд та порівняння існуючих моделей оцінки зрілості бізнесу та методів багатокритеріального аналізу; обґрунтування вибору гібридної моделі AHP-TOPSIS та розробка ієрархічної системи критеріїв оцінки (продукт, ринок, команда, фінанси); розробка математичної моделі та алгоритмів виявлення структурних дисбалансів (ризиків) у розвитку проєкту; проєктування архітектури та програмна реалізація інформаційної системи у вигляді Telegram-бота; тестування розробленої системи на тестових сценаріях та оцінка ефективності сформованих рекомендацій.

5. Перелік графічних матеріалів: сторінок – 96, таблиць – 4, рисунків – 20, посилань – 50, додатків – 5.

**Керівник роботи** _____ Інесса КУЛАКОВСЬКА
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ Ірина ШАВРІНА
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «28» червня 2025 р.

**КАЛЕНДАРНИЙ ПЛАН**
**кваліфікаційної роботи**

Тема: Інформаційна система з оцінки готовності стартапу до масштабування

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| **1.** | Отримання завдання на виконання КР | 27.06.2025 | 30.06.2025 | Виконано |
| **2.** | Аналіз предметної області та постановка задачі | 01.07.2025 | 20.07.2025 | Виконано |
| **3.** | Огляд літературних джерел за темою кваліфікаційної роботи | 21.07.2025 | 30.07.2025 | Виконано |
| **4.** | Аналіз існуючих підходів та методів для оцінки готовності стартапів | 01.09.2025 | 25.10.2025 | Виконано |
| **5.** | Реалізація обраних методів з аналізом отриманих результатів | 26.10.2025 | 21.11.2025 | Виконано |
| **6.** | Перший попередній захист КР на засіданні комісії кафедри | 25.11.2025 | 26.11.2025 | Виконано |
| **7.** | Корегування роботи за результатами попереднього захисту | 27.11.2025 | 05.12.2025 | Виконано |
| **8.** | Другий попередній захист КР на засіданні комісії кафедри | 09.12.2025 | 09.12.2025 | Виконано |
| **9.** | Доробка та остаточне оформлення КР | 10.12.2025 | 12.12.2025 | Виконано |
| **10** | Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 15.12.2025 | 16.12.2025 | Виконано |

**Керівник роботи** _____ _Інесса КУЛАКОВСЬКА____
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

**Здобувач** _____ ____Ірина ШАВРІНА _____
*(Особистий підпис)* *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану

«8» липня 2025 р.

# АНОТАЦІЯ

**кваліфікаційної роботи студентки групи 601 ЧНУ ім. Петра Могили**

**Шавріної Ірини Олександрівни**

**Тема: «Інформаційна система з оцінки готовності стартапу до масштабування»**


Проблема прийняття рішень щодо масштабування бізнесу є однією з найбільш критичних в управлінні інноваційними проєктами. Статистика свідчить, що передчасне масштабування є причиною краху значної частини стартапів, які вичерпують ресурси до моменту досягнення відповідності продукту ринку. У зв'язку з цим, розробка автоматизованих інструментів для комплексної діагностики зрілості бізнесу та підтримки прийняття рішень є важливим напрямом підвищення життєздатності інноваційних екосистем.

**Метою роботи** є вдосконалення та спрощення процесу оцінювання готовності стартап-проектів до етапу масштабування шляхом створення програмного засобу підтримки прийняття рішень.

Для досягнення поставленої мети використано методи системного аналізу, метод аналізу ієрархій (AHP), метод TOPSIS та методи об'єктно-орієнтованого програмування.

**Об'єктом** дослідження є процеси прийняття управлінських рішень у стартап-компаніях на етапі підготовки до масштабування.

**Предметом** дослідження є методи та моделі багатокритеріального оцінювання, зокрема гібридний підхід на основі методу аналізу ієрархій (AHP) та методу TOPSIS, а також програмні засоби їх реалізації.

Методи дослідження. У роботі використано методи системного аналізу, метод аналізу ієрархій (AHP) для визначення ваг критеріїв, модифікований метод TOPSIS із застосуванням некомпенсаторної логіки («вето-порогів») для ранжування альтернатив, а також методи об'єктно-орієнтованого програмування.

Результати дослідження. У першому розділі проаналізовано життєвий цикл стартапів та виявлено, що ключовим ризиком є дисбаланс між фінансовими ресурсами та відповідністю продукту ринку. У другому розділі розроблено математичну модель, яка, на відміну від класичних підходів, блокує високі оцінки стартапів за наявності критичних недоліків у продукті. У третьому та четвертому розділах спроєктовано та реалізовано інформаційну систему у вигляді Telegram-бота мовою Python (з використанням бібліотек NumPy, Matplotlib та СУБД SQLite). Система включає модуль евристичного аналізу для генерації стратегічних рекомендацій. У п'ятому розділі проведено ретроспективну валідацію системи на реальних

кейсах (Quibi та Grammarly), що підтвердило здатність алгоритму коректно ідентифікувати ризики передчасного масштабування.

В результаті розроблено інформаційну систему, яка здатна виконувати експрес-діагностику стартап-проєктів, візуалізувати результати оцінки та надавати обґрунтовані рекомендації для зниження ризиків масштабування.

**Наукова новизна** полягає у вдосконаленні методу TOPSIS шляхом введення штрафних коефіцієнтів для критичних метрик стартапу.

**Практичне значення.** Розроблена система дозволяє виконувати експрес-діагностику проєктів, візуалізувати результати у вигляді радарних діаграм та надавати обґрунтовані рекомендації засновникам та інвесторам.

Кваліфікаційна робота містить 96 сторінок, 20 рисунків, 4 таблиці, 50 використаних джерел та 5 додатків.

**Ключові слова:** стартап, масштабування, система підтримки прийняття рішень, AHP, TOPSIS, Python, Telegram-бот, оцінка ризиків.

# ABSTRACT

**qualification work of a student of group 601 of BSNU named after Petro Mohyla**

**Shavrina Iryna**

**Topic: " Information system for assessing startup readiness for scaling "**

The problem of decision-making regarding business scaling is one of the most critical in innovation project management. Statistics show that premature scaling causes the failure of a significant portion of startups that exhaust resources before achieving product-market fit. In this regard, the development of automated tools for complex diagnostics of business maturity and decision support is an important direction for increasing the viability of innovation ecosystems.

**The goal of the work** is to improve and simplify the process of assessing the readiness of startup projects for the scaling stage by creating a decision support software tool.

To achieve this goal, methods of system analysis, the Analytic Hierarchy Process (AHP), the TOPSIS method, and object-oriented programming methods were used.

**The object of the research** is the processes of managerial decision-making in startup companies at the stage of preparation for scaling.

**The subject of the research** is the methods and models of multi-criteria assessment, specifically a hybrid approach based on the Analytic Hierarchy Process (AHP) and the TOPSIS method, as well as software tools for their implementation.

Research methods. The work utilizes system analysis methods, the Analytic Hierarchy Process (AHP) for determining criterion weights, a modified TOPSIS method applying non-compensatory logic ("veto thresholds") for ranking alternatives, as well as object-oriented programming methods.

Research results. The first chapter analyzes the startup life cycle and identifies the imbalance between financial resources and product-market fit as a key risk. In the second chapter, a mathematical model is developed which, unlike classical approaches, blocks high scores for startups in the presence of critical product flaws. In the third and fourth chapters, an information system is designed and implemented as a Telegram bot using Python (utilizing NumPy, Matplotlib libraries, and SQLite DBMS). The system includes a heuristic analysis module for generating strategic recommendations. In the fifth chapter, a retrospective validation of the system was conducted on real cases (Quibi and Grammarly), which confirmed the algorithm's ability to correctly identify premature scaling risks.

As a result, an information system has been developed that is capable of performing express diagnostics of startup projects, visualizing assessment results, and providing grounded recommendations to reduce scaling risks.

Scientific novelty lies in the improvement of the TOPSIS method by introducing penalty coefficients for critical startup metrics.

Practical value. The developed system allows for express diagnostics of projects, visualization of results in the form of radar charts, and provision of grounded recommendations to founders and investors.

The qualification work comprises 96 pages, 20 figures, 4 tables, 50 cited sources, and 5 appendices.

**Keywords:** startup, scaling, decision support system, AHP, TOPSIS, Python, Telegram bot, risk assessment.

ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

| | |
|---|---|
| БД | – база даних |
| ІС | – інформаційна система |
| ПЗ | – програмне забезпечення |
| СППР | – система підтримки прийняття рішень |
| СУБД | – система управління базами даних |
| | |
| AHP | – Analytic Hierarchy Process |
| API | – Application Programming Interface |
| CR | – Consistency Ratio |
| IDE | – Integrated Development Environment |
| JSON | – JavaScript Object Notation |
| MCDM | – Multi-Criteria Decision Making |
| MRL | – Market Readiness Level |
| MVP | – Minimum Viable Product |
| PMF | – Product-Market Fit |
| SaaS | – Software as a Service |
| SQL | – Structured Query Language |
| TOPSIS | – Technique for Order of Preference by Similarity to Ideal Solution |
| TRL | – Technology Readiness Level |
| UI | – User Interface |
| UX | – User Experience |

# ВСТУП

З розвитком інноваційної економіки та збільшенням кількості технологічних компаній питання ефективного управління стартапами набувають дедалі більшої актуальності. Однією з найпоширеніших проблем у сучасному бізнес-середовищі є високий рівень невдач стартап-проєктів [1, 2] на етапі масштабування, коли компанія переходить від пошуку бізнес-моделі до її активного зростання. Щороку тисячі стартапів припиняють існування через передчасне вичерпання ресурсів та стратегічні помилки у плануванні розвитку.

Існуючі рішення у сфері оцінки стартапів мають певні обмеження. Багато традиційних методів базуються на суто фінансових показниках, які є нерелевантними на ранніх стадіях, або на суб'єктивних експертних думках, що робить їх дорогими та важкодоступними для широкого загалу. Методи аналізу, які використовуються у простих онлайн-чеклістах, часто виявляються недостатніми для комплексної діагностики готовності до масштабування, що створює додаткові ризики для засновників та інвесторів.

Провідні аналітичні агенції та акселератори, такі як Startup Genome та Y Combinator, активно працюють над удосконаленням методологій оцінки зрілості бізнесу, впроваджуючи метрики продуктової та ринкової відповідності. Проте, незважаючи на значний прогрес, проблема наявності доступного автоматизованого інструменту для об'єктивної оцінки готовності стартапу залишається актуальною.

**Актуальність теми** полягає у тому, що передчасне масштабування є причиною краху близько 74% інтернет-стартапів [3], що завдає значних фінансових збитків екосистемі. У зв'язку з цим, розробка інтелектуальних систем для автоматичного оцінювання готовності стартапу до масштабування є важливим напрямом підвищення ефективності управлінських рішень та зниження інвестиційних ризиків.

Дана робота пов'язана з науковими дослідженнями у галузі системного аналізу та прийняття рішень, зокрема зі створенням систем підтримки прийняття

рішень (СППР) на основі методів багатокритеріального аналізу (MCDM) [13, 20]. Результати цієї роботи можуть бути використані для підвищення якості оцінювання стартапів та оптимізації стратегій їх розвитку.

**Метою роботи** є вдосконалення та спрощення процесу оцінювання готовності стартап-проектів до етапу масштабування шляхом створення програмного засобу підтримки прийняття рішень.

Для досягнення поставленої мети використано методи системного аналізу, метод аналізу ієрархій (AHP), метод TOPSIS та методи об’єктно-орієнтованого програмування.

**Об’єктом** дослідження є процеси прийняття управлінських рішень у стартап-компаніях на етапі підготовки до масштабування.

**Предметом** дослідження є методи та моделі багатокритеріального оцінювання, зокрема гібридний підхід на основі методу аналізу ієрархій (AHP) та методу TOPSIS, а також програмні засоби їх реалізації.

У процесі розробки інформаційної системи використовувались сучасні інструментальні засоби та платформи. Мова програмування Python застосована для реалізації математичних алгоритмів та логіки бекенду, бібліотека python-telegram-bot – для створення інтерактивного інтерфейсу користувача, СУБД SQLite – для збереження даних, а середовище VS Code/PyCharm забезпечує зручну розробку та налагодження проєкту.

Пояснювальна записка складається зі вступу, п’яти розділів, висновків та додатків. У першому розділі проаналізовано життєвий цикл стартапів та виявлено, що ключовим ризиком є дисбаланс між фінансовими ресурсами та відповідністю продукту ринку. У другому розділі розроблено математичну модель, яка, на відміну від класичних підходів, блокує високі оцінки стартапів за наявності критичних недоліків у продукті. У третьому та четвертому розділах спроєктовано та реалізовано інформаційну систему у вигляді Telegram-бота мовою Python (з використанням бібліотек NumPy, Matplotlib та СУБД SQLite). Система включає модуль евристичного аналізу для генерації стратегічних рекомендацій. У п’ятому

розділі проведено ретроспективну валідацію системи на реальних кейсах (Quibi та Grammarly), що підтвердило здатність алгоритму коректно ідентифікувати ризики передчасного масштабування.

В результаті розроблено інформаційну систему, яка здатна виконувати експрес-діагностику стартап-проєктів, візуалізувати результати оцінки та надавати обґрунтовані рекомендації для зниження ризиків масштабування, що є важливим кроком у забезпеченні сталого розвитку інноваційних підприємств.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ОЦІНКИ ГОТОВНОСТІ СТАРТАПІВ

## 1.1 Життєвий цикл стартапу та місце етапу масштабування

Для глибокого та системного розуміння проблеми оцінки готовності до масштабування необхідно спершу визначити місце цього етапу в загальній еволюції інноваційного проєкту. Стартап, на відміну від класичного малого бізнесу (SME), характеризується не лише створенням нового продукту чи послуги, але й, за визначенням Стіва Бланка [4], є тимчасовою організацією, створеною для пошуку відтворюваної та масштабованої бізнес-моделі в умовах екстремальної невизначеності. Цей процес трансформації від ідеї до стійкої компанії не є хаотичним; він підпорядковується певним закономірностям, які зазвичай описуються через моделі життєвого циклу.

В сучасній теорії інноваційного менеджменту існує декілька підходів до періодизації розвитку стартапів (модель Стіва Бланка «Customer Development», модель Еріка Ріса «Lean Startup») [5], проте однією з найбільш визнаних та емпірично обґрунтованих є модель, запропонована в рамках глобального дослідження «Startup Genome Report» [1]. Ця модель структурує еволюцію компанії через шість послідовних стадій розвитку, кожна з яких має свої унікальні цілі, метрики та ризики:

– відкриття: на цьому початковому етапі засновники формулюють ідею та початкові гіпотези щодо проблеми, яку вони планують вирішити. Основне завдання – створити перший прототип або Мінімально життєздатний продукт (Minimum Viable Product, MVP) для перевірки базових припущень;

– валідація: стартап тестує свій MVP на перших користувачах ("early adopters"), збирає зворотний зв'язок та ітераційно вдосконалює продукт. Головна мета цього етапу – досягти так званого Product-Market Fit – стану, коли продукт задовольняє реальну ринкову потребу, і за нього готова платити значна кількість клієнтів;

– ефективність: після знаходження Product-Market Fit, компанія фокусується на оптимізації бізнес-моделі та операційних процесів. На цьому етапі вдосконалюються канали залучення клієнтів, оптимізується ціноутворення та налагоджуються внутрішні процеси, щоб зробити їх повторюваними та прибутковими в невеликому масштабі;

– масштабування: це етап агресивного зростання. Маючи підтверджену та ефективну бізнес-модель, компанія залучає значні інвестиції для швидкого захоплення ринку, розширення команди, виходу на нові географії та інтенсифікації маркетингових зусиль;

– підтримка: компанія стає зрілою, фокусується на утриманні ринкових позицій, поступових інноваціях та конкуренції з іншими великими гравцями;

– збереження: етап, на якому компанія може бути поглинута більшим гравцем або сама починає купувати менші стартапи для підтримки інноваційності.

Важливо чітко розмежовувати поняття «Зростання» та «Масштабування». Зростання зазвичай передбачає лінійне збільшення доходів при пропорційному збільшенні витрат (наприклад, консалтинговий бізнес: більше клієнтів потребує більше консультантів). Натомість, масштабування – це здатність бізнесу збільшувати дохід експоненційно при мінімальному або лінійному зростанні операційних витрат (наприклад, SaaS-рішення: один раз написаний код продається мільйонам користувачів без значних додаткових витрат на копіювання).

Саме етап масштабування є переломним моментом і, по суті, «точкою неповернення» для стартапу. На відміну від попередніх етапів (Discovery, Validation), де панує філософія «Ощадливого стартапу», що заохочує швидкі, дешеві експерименти та сприймає помилки як невід'ємну частину навчання (концепція «Fail Fast») [7, 49], на етапі масштабування ціна помилки стає критично високою.

Якщо на ранніх стадіях стартап оперує з мінімальними ресурсами, і вартість невдалого експерименту вимірюється тижнями роботи невеликої команди, то на етапі Scale компанія оперує мільйонними бюджетами. Залучення значних раундів

фінансування спрямовується на найм десятків нових співробітників, оренду офісів, розгортання дорогої серверної інфраструктури та проведення глобальних рекламних кампаній. Передчасний перехід до цього етапу, коли продукт ще не стабільний, бізнес-модель не доведена (LTV < CAC), а операційні процеси хаотичні, неминуче призводить до ефекту «дірявого відра»: маркетинговий бюджет спалюється на залучення клієнтів, які швидко йдуть через недоліки продукту, що веде до касових розривів та, зрештою, до банкрутства компанії.

## 1.2 Ключові фактори успіху та причини невдач стартапів при масштабуванні

Для розробки ефективної моделі оцінки необхідно чітко розуміти, які саме фактори визначають успіх чи невдачу стартапу на етапі зростання. Аналіз звітів провідних дослідницьких компаній, таких як CB Insights та Startup Genome, дозволяє систематизувати ці фактори.

Згідно з фундаментальним дослідженням аналітичної агенції CB Insights [2] (див. рис. 1.1), яка проаналізувала пост-мортеми понад 100 стартапів, можна виділити топ-5 причин краху, кожна з яких набуває специфічного забарвлення на етапі масштабування:

– відсутність ринкової потреби (42%). Це найбільш поширена причина. На етапі масштабування вона проявляється у вигляді "хибно-позитивного" Product-Market Fit. Компанія може отримати перших ранніх послідовників, але масовий ринок залишається байдужим до продукту. Масштабування маркетингу в таких умовах призводить до катастрофічного зростання витрат без адекватного зростання виручки;

– закінчилися кошти (29%). Нездатність залучити нові інвестиції або досягти операційної прибутковості. На етапі масштабування витрати зростають експоненційно через найм персоналу та розгортання інфраструктури. Якщо юніт-економіка не сходиться (вартість залучення клієнта перевищує дохід від нього), кожний новий клієнт лише пришвидшує банкрутство, а не віддаляє його;

– неправильна команда (23%). Масштабування вимагає трансформації компетенцій. Команда генералістів-ентузіастів, яка успішно створила MVP, часто не має навичок системного менеджменту, побудови відділів продажів та управління великими колективами. Відсутність досвідчених C-level менеджерів (CTO, CMO, CFO) стає вузьким місцем розвитку;

– програш у конкуренції (19%). На ранніх стадіях стартап може ігнорувати конкурентів, але при виході на широкий ринок він потрапляє на "радари" великих гравців. Якщо стартап не сформував стійкої конкурентної переваги (технологічного бар'єру або мережевого ефекту) до початку масштабування, він ризикує бути витісненим ресурсно сильнішими опонентами;

– проблеми з ціноутворенням та бізнес-моделлю (18%). Нездатність знайти модель монетизації, яка покриває витрати. На етапі масштабування це проявляється у невідповідності між цінністю продукту та його собівартістю при масовому виробництві або обслуговуванні.

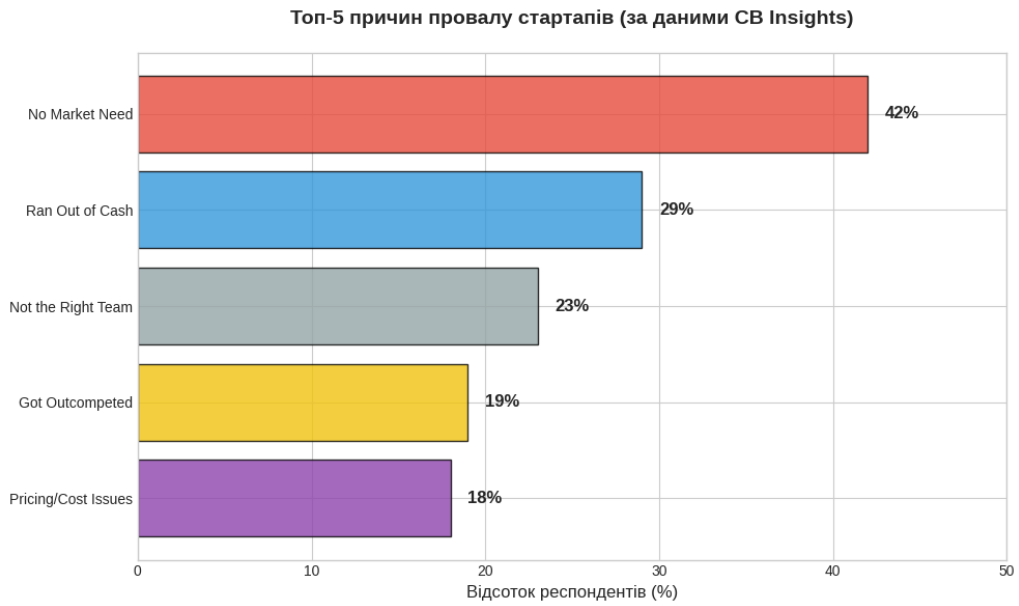


Рисунок 1.1 – Статистика причин провалу стартапів

Окрему увагу слід приділити концепції "Передчасного масштабування" [3], яку детально описано у звіті Startup Genome. Дослідники визначають стартап як

складну систему, що розвивається у п'яти вимірах: клієнти, продукт, команда, бізнес-модель та фінанси.

Передчасне масштабування виникає, коли один із вимірів розвивається значно швидше за інші. Наприклад:

– масштабування команди до масштабування продукту. Найм великої кількості персоналу, коли продукт ще сирий і потребує частих змін, призводить до організаційного хаосу;

– масштабування маркетингу до масштабування бізнес-моделі. Витрачання бюджету на рекламу до того, як вартість залучення клієнта (CAC) стане меншою за його життєву цінність (LTV), призводить до "спалювання" грошей.

Статистика свідчить, що 74% інтернет-стартапів з високим потенціалом зростання зазнають невдачі саме через передчасне масштабування. Ті ж компанії, що масштабуються своєчасно, зростають у 20 разів швидше.

З іншого боку, дослідження Startup Genome виділяє ключові фактори [1], що корелюють з успіхом стартап-екосистем та окремих компаній. Ці фактори включають:

– ринкова досяжність. Здатність стартапу виходити на глобальні ринки. Стартапи, які від початку орієнтуються на експорт, зростають у 2.1 рази швидше;

– збалансоване фінансування. Доступ до "розумних грошей" – інвестицій, що супроводжуються менторською підтримкою та нетворкінгом, особливо на ранніх стадіях (Seed/Series A);

– талант та досвід. Наявність у команді засновників людей з попереднім досвідом роботи у швидкозростаючих технологічних компаніях;

– технологічна захищеність. Наявність інтелектуальної власності (патентів) або унікальних технологічних рішень, що створюють бар'єри входу для конкурентів.

Порівняльний аналіз факторів успіху та причин невдач дозволяє зробити важливий методологічний висновок для даної роботи: критерії оцінки готовності

до масштабування не можуть бути одновимірними (наприклад, лише фінансовими). Вони повинні бути комплексними та охоплювати всі ключові аспекти діяльності стартапу (продукт, ринок, команда, фінанси), а також враховувати баланс між цими вимірами для запобігання ризику передчасного масштабування.

### 1.3 Огляд існуючих моделей та фреймворків оцінки зрілості

Ідея формальної оцінки готовності широко застосовується в інженерії та управлінні інноваціями. Аналіз існуючих фреймворків дозволяє запозичити перевірені підходи та адаптувати їх до специфіки стартапів.

Рівні готовності технологій (technology readiness levels, TRL)**.** Базовою моделлю оцінки зрілості інновацій є шкала Technology Readiness Levels (TRL) [23, 24], розроблена NASA у 1970-х роках для управління ризиками космічних програм. Пізніше ця методологія була адаптована Європейським Союзом для оцінки проєктів у рамках програми Horizon 2020. Шкала TRL дозволяє класифікувати стадію розвитку технології за дев'ятьма рівнями: від формулювання ідеї до повномасштабного впровадження.

Детальна характеристика рівнів TRL виглядає наступним чином:

– TRL 1. Фундаментальні дослідження. Спостерігаються та описуються базові принципи. Наукові дослідження переходять у прикладну площину. На цьому етапі формулюється лише концепція технології без її фізичної реалізації;

– TRL 2. Формулювання технологічної концепції. Визначено потенційне практичне застосування. Це все ще спекулятивний етап: концепція вже існує, але експериментальних доказів її працездатності ще немає;

– TRL 3. Експериментальне підтвердження концепції (Proof of Concept). Проведено аналітичні та лабораторні дослідження для підтвердження критичних функцій. Створено перший макет або математичну модель, що доводить принципову можливість реалізації;

– TRL 4. Лабораторний прототип. Окремі компоненти технології інтегровано та перевірено в лабораторних умовах. Прототип має низьку точність (low-fidelity), але демонструє працездатність основних вузлів;

– TRL 5. Валідація в релевантному середовищі. Технологія перевірена в умовах, наближених до реальних (симуляція або стендові випробування). Прототип стає більш надійним та наближеним до фінального продукту;

– TRL 6. Демонстрація прототипу. Прототип системи перевірено в операційному середовищі (наприклад, бета-тестування з обмеженою групою користувачів або польові випробування). Це критичний крок перед комерціалізацією;

– TRL 7. Демонстрація системи. Прототип системи пройшов повний цикл випробувань в реальних умовах експлуатації. Система готова до інженерного впровадження та сертифікації;

– TRL 8. Завершена та кваліфікована система. Технологія доведена до фінального вигляду. Пройдено всі необхідні сертифікаційні випробування та отримано дозволи. Система готова до серійного виробництва;

– TRL 9. Підтвердження в успішних місіях. Технологія успішно працює в реальних умовах протягом тривалого часу. Відбувається повне розгортання, масштабування та комерційна експлуатація.

Хоча TRL є стандартом для DeepTech стартапів, для класичних IT-продуктів (SaaS, Marketplace) ця шкала є недостатньою, оскільки вона фокусується виключно на технології, ігноруючи ринкові аспекти. Технічно досконалий продукт (TRL 9) може зазнати краху через відсутність попиту.

Рівні готовності ринку (market readiness level, MRL). Для компенсації "техноцентричності" TRL було розроблено шкалу Market Readiness Levels (MRL) [25]. Цей фреймворк оцінює зрілість ринку та бізнес-моделі. Він допомагає відповісти на питання: "Чи хтось готовий платити за цю технологію?". Шкала MRL включає наступні етапи:

– MRL 1. Гіпотеза проблеми. Сформульовано гіпотезу про проблему клієнта. Проведено кабінетні дослідження ринку (Desk Research) для підтвердження актуальності проблеми;

– MRL 2. Валідація проблеми. Проведено проблемні інтерв'ю (CustDev). Підтверджено наявність "болю" у цільової аудиторії. Ринок визначено кількісно (розраховано TAM/SAM/SOM);

– MRL 3. Гіпотеза рішення. Сформульовано ціннісну пропозицію (Value Proposition). Описано, як продукт вирішує проблему краще за існуючих конкурентів;

– MRL 4. Валідація рішення. Проведено решенческі інтерв'ю. Клієнти підтвердили зацікавленість у прототипі. Отримано перші листи про наміри (LOI) або попередні замовлення;

– MRL 5. Валідація Product-Market Fit. Продукт запущено. Є перші реальні продажі. Спостерігається повернення користувачів (Retention). Юніт-економіка на цьому етапі ще може бути негативною;

– MRL 6. Валідація бізнес-моделі. Юніт-економіка стає позитивною (LTV > CAC). Визначено ефективні канали продажів. Продукт готовий до повноцінної монетизації;

– MRL 7. Масштабування продажів. Канали залучення клієнтів працюють стабільно та прогнозовано. Початок агресивного маркетингу та швидке зростання доходу;

– MRL 8. Домінування на ринку. Продукт займає значну частку ринку. Бренд стає впізнаваним. Починається експансія на міжнародні ринки;

– MRL 9. Зрілий бізнес. Компанія стала публічною (IPO) або визнаним лідером галузі. Бізнес генерує стабільний та прогнозований прибуток.

В ідеальному сценарії розвиток стартапу має відбуватися синхронно за шкалами TRL та MRL. Розрив між ними (наприклад, TRL 9 при MRL 1) створює ризик "Technology Push" – створення нікому не потрібного продукту.

Моделі зрілості екосистем (ecosystem maturity model). Це академічні моделі, що використовуються для оцінки рівня розвитку цілих стартап-екосистем. Зазвичай вони пропонують набір факторів (доступ до фінансування, якість людського капіталу тощо) та оцінюють їх за декількома рівнями зрілості, наприклад, L1 (Зародження), L2 (Розвиток), L3 (Зрілість). Ідею оцінки кожного критерію за дискретною шкалою зрілості можна адаптувати для оцінки окремого стартапу.

Синтез цих підходів дозволяє сформувати обґрунтовану методологію. Ієрархічна структура критеріїв може бути запозичена з MRL та розширена на основі факторів успіху. Підхід до оцінки кожного критерію за бальною шкалою, що відповідає певному рівню зрілості, може бути адаптований з моделей зрілості екосистем.

## 1.4 Аналіз останніх досліджень та існуючих інформаційних систем-аналогів

Аналіз актуальних публікацій демонструє широке застосування методів багатокритеріального аналізу для вирішення складних задач вибору. Зокрема, гібридний підхід, що поєднує метод аналізу ієрархій (AHP) для визначення ваг критеріїв та метод TOPSIS для ранжування альтернатив, довів свою ефективність у багатьох галузях, від вибору постачальників до пріоритезації IT-проєктів [16, 17]. Дослідження де Соуза та ін. та Ондера і Сундуса обґрунтовують, що така комбінація дозволяє враховувати як суб'єктивні судження експертів, так і об'єктивні показники, що є критично важливим для оцінки стартапів [46]. Роботи Шена та ін. та Джеймса та ін. підтверджують універсальність та надійність даної методології для задач з великою кількістю критеріїв.

Водночас, аналіз ринку програмного забезпечення показує, що наразі відсутня універсальна та доступна інформаційна система, яка б повністю відповідала меті даної роботи. Існуючі рішення можна класифікувати наступним чином:

– загальні системи управління бізнесом: системи як Salesforce, HubSpot або SAP є потужними інструментами для управління операціями, але вони призначені для зрілих компаній і не мають вбудованих модулів для оцінки готовності до масштабування;

– пропрієтарні системи венчурних фондів: багато великих венчурних фондів розробляють власні внутрішні системи для аналізу стартапів. Ці системи є закритими, їхня методологія не є публічною, і вони недоступні для широкого загалу;

– прості онлайн-калькулятори та чек-лісти: в мережі можна знайти багато спрощених інструментів, що пропонують пройти тест з кількох питань. Зазвичай вони не мають наукового обґрунтування, використовують занадто узагальнені критерії та не дозволяють налаштовувати ваги критеріїв, що робить їхню оцінку поверхневою.

Таким чином, аналіз останніх досліджень підтверджує доцільність використання гібридних методів AHP-TOPSIS, а відсутність доступних аналогів обґрунтовує необхідність розробки нової інформаційної системи.

## 1.5 Постановка задачі дослідження

На основі проведеного аналізу предметної області, виявлених проблем та існуючих підходів, можна сформулювати основні завдання дослідження, спрямовані на розробку інформаційної системи для об'єктивної оцінки готовності стартапу до масштабування. До основних завдань дослідження належать:

– проаналізувати предметну область, визначити ключові фактори успіху та ризики на етапі масштабування для формування теоретичної основи дослідження;

– провести огляд та порівняння існуючих методів підтримки прийняття рішень для вибору оптимального наукового апарату, здатного враховувати як якісні, так і кількісні показники;

– обґрунтувати вибір та розробити математичну модель на основі гібридного методу AHP-TOPSIS для комплексної оцінки;

– сформувати ієрархічну структуру критеріїв оцінки, що всебічно охоплюють продуктові, ринкові, командні та фінансові аспекти готовності стартапу;

– розробити архітектуру інформаційної системи, що реалізує запропоновану модель, та обґрунтувати вибір технологічного стеку для її програмної реалізації.

Вирішення цих завдань дозволить створити інструмент, що заповнить існуючу нішу на ринку та надасть засновникам та інвесторам науково обґрунтовану методологію для прийняття зважених рішень щодо масштабування.

**Висновки до розділу 1**

У першому розділі виконано системний аналіз предметної області та існуючих підходів до оцінювання стартапів, що дозволило сформувати теоретико-методологічний базис для подальшої розробки інформаційної системи. За результатами проведеного дослідження зроблено наступні висновки:

– ідентифіковано критичну проблематику етапу масштабування. На основі аналізу життєвого циклу інноваційних проєктів встановлено, що етап масштабування (Scaling) є «точкою неповернення» для стартапу. Визначено, що явище «передчасного масштабування» (Premature Scaling), яке полягає у нарощуванні витрат до досягнення відповідності продукту ринку (Product-Market Fit), є причиною краху 74% інтернет-проєктів. Це обґрунтовує необхідність створення інструментів для точної діагностики готовності до переходу на цей етап;

– визначено ключові метрики успіху. Аналіз статистичних даних (звіти CB Insights, Startup Genome) дозволив виділити основні фактори ризику: відсутність ринкової потреби, фінансові розриви, дисбаланс команди та ігнорування конкуренції. На основі цього зроблено висновок про необхідність застосування комплексного багатокритеріального підходу до оцінювання, який має охоплювати чотири виміри: продукт, ринок, команду та фінанси;

– проаналізовано існуючі методології. Розглянуто моделі TRL (рівні готовності технологій) та MRL (рівні готовності ринку). Встановлено, що окреме використання цих шкал є недостатнім для стартапів, оскільки вони не враховують організаційні та фінансові аспекти. Обґрунтовано доцільність синтезу цих підходів у рамках єдиної ієрархічної моделі оцінки;

– виявлено прогалину в існуючому програмному забезпеченні. Огляд ринку IT-рішень показав відсутність доступної, універсальної та науково обґрунтованої системи підтримки прийняття рішень. Існуючі інструменти є або занадто спрощеними (онлайн-чеклісти), або закритими (внутрішні системи венчурних фондів), або неспеціалізованими (CRM/ERP). Це підтверджує актуальність розробки нової інформаційної системи;

– обґрунтовано вибір наукового апарату. Аналіз літературних джерел підтвердив ефективність використання методів багатокритеріального аналізу прийняття рішень (MCDM) для задач такого класу. Зокрема, визначено перспективність гібридного підходу, що поєднує метод аналізу ієрархій (AHP) для визначення пріоритетів та метод TOPSIS для ранжування альтернатив.

Таким чином, результати першого розділу підтверджують наявність науково-прикладної проблеми та формують чіткі вимоги до розробки інформаційної системи, що буде реалізована у наступних розділах роботи.

# 2 МЕТОДИ ТА МОДЕЛІ ОЦІНКИ ГОТОВНОСТІ СТАРТАПУ

## 2.1 Методологічні основи багатокритеріального аналізу в оцінці проєктів

Завдання оцінки готовності стартапу до масштабування є класичною задачею багатокритеріального аналізу (Multi-Criteria Decision Making, MCDM) [13, 19], оскільки вимагає врахування великої кількості різнорідних, часто суперечливих, критеріїв – як кількісних (фінансові показники, метрики зростання), так і якісних (досвід команди, якість продукту, ринкові ризики). Просте порівняння за одним параметром (наприклад, поточний дохід) є недостатнім, оскільки воно не враховує довгостроковий потенціал, стійкість бізнес-моделі чи технологічні ризики.

Методи MCDM надають науково обґрунтований інструментарій для структурування таких складних проблем, врахування суб'єктивних пріоритетів особи, що приймає рішення, та отримання інтегральної оцінки для порівняння альтернатив. Існує низка методів для вирішення таких задач, зокрема SAW (Simple Additive Weighting), ELECTRE (ELimination Et Choix Traduisant la REalité), PROMETHEE (Preference Ranking Organization METHod for Enrichment of Evaluations), AHP (Analytic Hierarchy Process), TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) та інші.

Особливістю оцінки стартапів на ранніх стадіях є високий рівень невизначеності та суб'єктивності вхідних даних. У традиційній (чіткій) логіці будь-яке твердження може бути або істинним, або хибним. Однак у реальному бізнес-середовищі такі поняття, як «компетентна команда», «перспективний ринок» або «високий ризик», важко описати бінарними величинами "так" або "ні".

Для математичної формалізації таких лінгвістичних невизначеностей доцільно використати апарат теорії нечітких множин, запропонований американським математиком Лотфі Заде [14]. Ця теорія дозволяє оперувати поняттями, що мають нечіткі межі, наближаючи комп'ютерну логіку до людського мислення.

Ключовим поняттям теорії є функція належності. На відміну від класичної теорії множин, де елемент або належить множині повністю (100%), або не належить їй зовсім (0%), у нечіткій логіці елемент може належати множині частково. Ступінь належності може набувати будь-якого значення від 0 до 1, що дозволяє моделювати плавні переходи між станами.

У контексті систем підтримки прийняття рішень (СППР) це дає можливість оперувати лінгвістичними змінними. Це змінні, значеннями яких є не числа, а слова або речення природної мови (терми). Наприклад, змінна «Рівень готовності» може приймати значення: «Низький», «Середній», «Високий». Кожен із цих термів описується своєю функцією належності.

Для відображення лінгвістичних термів у числові значення в інженерній практиці найчастіше використовуються два типи функцій:

– трикутна функція належності. Цей тип функції використовується для опису понять, які мають одне найбільш характерне значення (пік). Наприклад, якщо ми говоримо про оцінку "Приблизно 5 балів", то ступінь впевненості є максимальним саме в точці 5, і лінійно спадає до нуля при віддаленні від цього значення в бік менших або більших чисел. Графічна інтерпретація трикутної функції наведена на рисунку (див. рис. 2.1). Вона ідеально підходить для моделювання експертних оцінок, де є чітко виражений центр тяжіння думки;

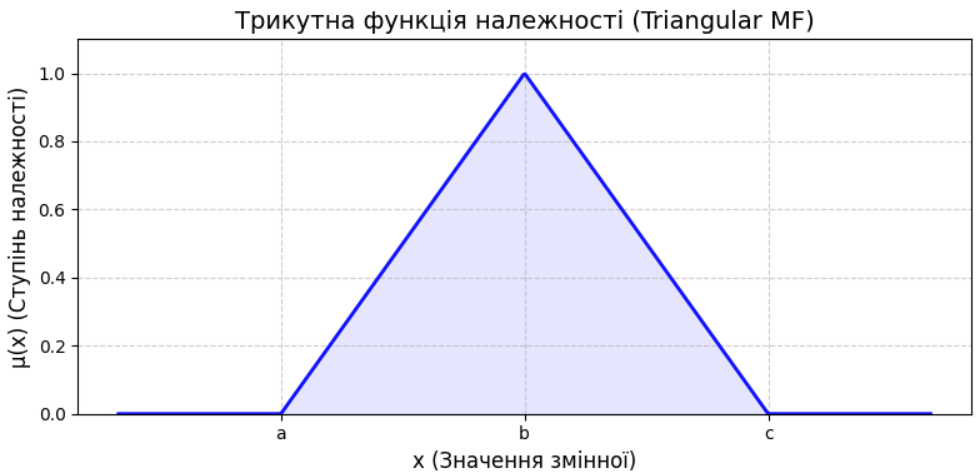


Рисунок 2.1 – Графічне представлення трикутної функції належності

– трапецієподібна функція належності. Цей вид функцій застосовується, коли поняття характеризується інтервалом повної впевненості ("ядром"). Наприклад, поняття "Прийнятний рівень прибутку" може означати будь-яке значення в діапазоні від 20% до 30%. У цьому проміжку функція належності дорівнює одиниці (максимум), а за його межами плавно знижується. Такий тип функцій (див. рис. 2.2) доцільно використовувати для моделювання стабільних станів або діапазонів допуску, що є характерним для фінансових показників.

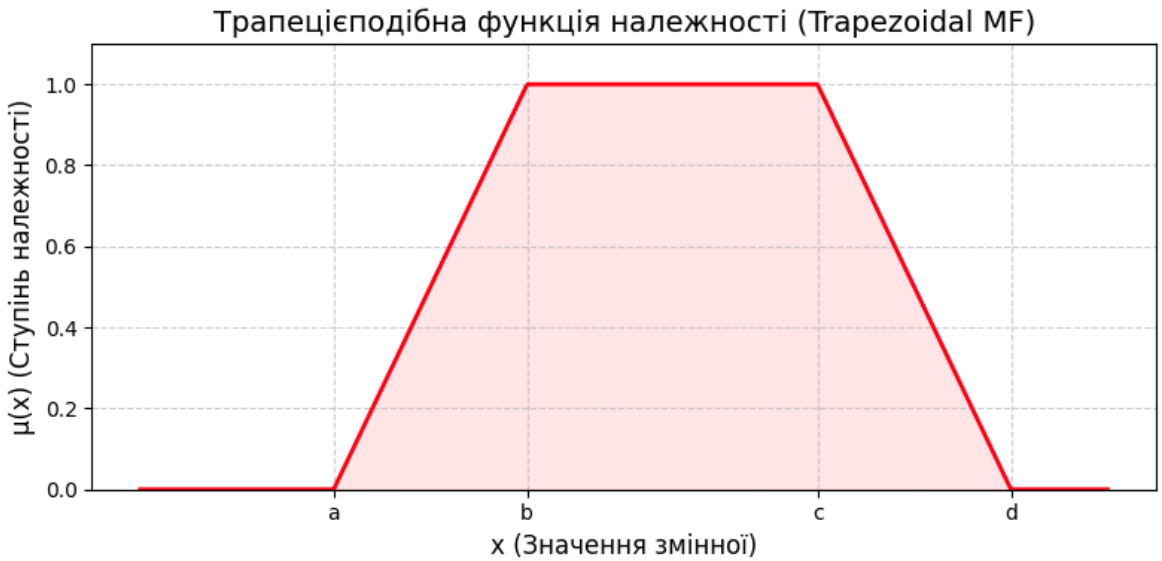


Рисунок 2.2 – Графічне представлення трапецієподібної функції належності

Застосування принципів нечіткої логіки у розроблюваній системі реалізується через використання багаторівневих шкал оцінювання. Такий підхід дозволяє поєднати математичну строгість методів багатокритеріального аналізу з гнучкістю людського мислення, що є критично важливим для адекватної діагностики інноваційних проєктів на ранніх стадіях їх розвитку.

Для даної роботи було обрано гібридний підхід, що поєднує методи AHP та TOPSIS. Цей вибір обґрунтований тим, що ці два методи ідеально доповнюють один одного, вирішуючи дві послідовні підзадачі в рамках загальної проблеми

оцінки: визначення стратегічних пріоритетів та об'єктивне ранжування альтернативи.

## 2.2 Обґрунтування вибору гібридної моделі AHP-TOPSIS

Вибір методологічного апарату для оцінки стартапів є складною науковою задачею, оскільки об'єкт дослідження характеризується дуалізмом: з одного боку, існують об'єктивні вимірювані показники (фінанси, метрики трафіку), а з іншого – глибоко суб'єктивні стратегічні пріоритети інвесторів (бачення, довіра до команди). Використання простих лінійних методів оцінювання не дозволяє врахувати цю багатовимірність. Тому в даній роботі обґрунтовується доцільність використання гібридного підходу, що поєднує переваги двох класичних методів багатокритеріального аналізу.

Метод аналізу ієрархій (AHP), розроблений американським вченим Томасом Сааті [11, 12], базується на особливостях людської психології сприйняття складних проблем. Його фундаментальна ідея полягає в тому, що людина не здатна одночасно оцінити вплив десятків факторів на кінцевий результат. Проте, людина здатна дуже точно порівняти два об'єкти між собою і сказати, який з них є важливішим.

Сутність методу полягає в декомпозиції складної проблеми на ієрархічну структуру. На вершині знаходиться головна мета (успішне масштабування), нижче – групи критеріїв, а ще нижче – конкретні показники.

Замість того, щоб намагатися прямо призначити вагу кожному критерію (наприклад, "Маркетинг – це 25%"), метод пропонує експерту здійснити серію попарних порівнянь. Експерт відповідає на прості питання: "Що важливіше для успіху: команда чи технологія? І наскільки?". Цей процес дозволяє перетворити якісні, інтуїтивні судження експерта на кількісні вагові коефіцієнти.

Критично важливою перевагою AHP є наявність вбудованого механізму перевірки логічної узгодженості. Якщо експерт стверджує, що критерій А важливіший за Б, Б важливіший за В, але при цьому В важливіший за А, метод

математично виявляє це протиріччя і сигналізує про необхідність перегляду оцінок. Це дозволяє гарантувати, що налаштована стратегія оцінювання є логічно цілісною, а не хаотичним набором думок.

Якщо AHP відповідає за налаштування "оптики" оцінювання (пріоритетів), то метод TOPSIS [13 ,15] використовується безпосередньо для ранжування стартапів.

Цей метод базується на геометричній концепції. Уявімо, що кожен стартап – це точка у багатовимірному просторі, де кожна вісь відповідає певному критерію оцінки (фінанси, продукт, ринок тощо).

Логіка методу полягає у визначенні двох гіпотетичних точок у цьому просторі:

– позитивне ідеальне рішення (PIS). Це уявний стартап, який має найкращі можливі значення за всіма критеріями одночасно (максимальний прибуток, найкраща команда, мінімальні ризики). У реальності такого проекту може не існувати, але він слугує орієнтиром;

– негативне ідеальне рішення (NIS). Це уявний "анти-стартап", який має найгірші значення за всіма параметрами.

Алгоритм обчислює геометричну відстань від реального стартапу, що оцінюється, до цих двох точок. Найкращим вважається той проект, який знаходиться геометрично найближче до "Ідеалу" і одночасно якнайдалі від "Анти-ідеалу".

Такий підхід дозволяє уникнути однобокості оцінювання. Наприклад, стартап з величезним прибутком, але жахливою репутацією і слабкою командою, може опинитися далеко від ідеалу, оскільки метод враховує баланс всіх показників. Фінальний результат представляється у вигляді коефіцієнта відносної близькості (від 0 до 100%), що є зрозумілим метрикою для прийняття рішень.

Впровадження гібридної моделі дозволяє розділити процес на два етапи, оптимізуючи взаємодію з користувачем:

– стратегічний етап (AHP). Виконується один раз (або рідко). Користувач налаштовує свою інвестиційну стратегію, визначаючи, що для нього важливіше (наприклад, "Команда" важливіша за "Ринок"). Це формує профіль оцінювання;

– операційний етап (TOPSIS). Виконується для кожного стартапу. Система бере налаштовані ваги з першого етапу і миттєво розраховує рейтинг конкретного проєкту на основі його показників.

Таким чином, комбінація AHP та TOPSIS створює логічно завершений, гнучкий та зручний інструмент. AHP забезпечує адаптивність під стратегію інвестора, а TOPSIS гарантує математичну об'єктивність та швидкість обробки даних, що є ідеальним рішенням для архітектури Telegram-бота.

## 2.3 Розробка ієрархічної моделі критеріїв оцінки

На основі аналізу, проведеного в Розділі 1, зокрема з урахуванням факторів успіху та провалу, фреймворку MRL [25], та практичного прикладу критеріїв, було розроблено дворівневу ієрархічну модель. Модель складається з чотирьох груп критеріїв першого рівня, які деталізуються у вісьмох критеріях другого рівня. Ця структура є ядром системи оцінки та основою для застосування методу AHP.

Ієрархічну структуру критеріїв оцінки готовності до масштабування наведено в таблиці 2.1.

Таблиця 2.1 – Ієрархічна структура критеріїв оцінки готовності до масштабування

| **Рівень 1.Група Критеріїв** | **Рівень 2. Критерій** | **Опис** |
|---|---|---|
| 1. Продукт та Технологія | 1.1.Зрілість продукту (Product-Market Fit) | Продукт вирішує реальну, підтверджену проблему для цільової аудиторії; є стабільний попит та низький відтік клієнтів. |
| | 1.2. Технічна масштабованість | Архітектура системи, бази даних та інфраструктура готові до експоненційного зростання навантаження; налагоджені DevOps процеси. |

Кінець таблиці 2.1

| **Рівень 1.Група Критеріїв** | **Рівень 2. Критерій** | **Опис** |
|---|---|---|
| 2. Ринок та Бізнес-модель | 2.1. Розмір та доступність ринку (TAM/SAM/SOM) | Загальний, доступний та реально досяжний обсяги ринку є достатньо великими для забезпечення росту; є чітка стратегія виходу на нові ринки/сегменти. |
| | 2.2. Валідована бізнес-модель | Модель монетизації перевірена на практиці; юніт-економіка є позитивною; процеси продажів та маркетингу є повторюваними та масштабованими. |
| 3. Команда та Операції | 3.1. Компетентність та повнота команди | Ключові ролі (технічні, продуктові, маркетингові, продажі) закриті досвідченими фахівцями; команда має досвід у масштабуванні або залучила відповідних партнерів. |
| | 3.2. Операційна готовність | Основні бізнес-процеси стандартизовані, задокументовані (SOP) та частково автоматизовані; впроваджено базові системи (CRM, аналітика). |
| 4. Фінанси та Юридичні аспекти | 4.1. Фінансова стабільність та планування | Компанія має фінансові резерви для покриття операційних витрат на 6-12 місяців; існує детальний фінансовий план масштабування. |
| | 4.2. Юридична готовність | Захищена інтелектуальна власність (патенти, торгові марки); структура компанії є прозорою для інвесторів; дотримання регуляторних норм (напр., GDPR). |

## 2.4 Математична модель та алгоритм комплексної оцінки

Алгоритм роботи системи складається з двох послідовних етапів, що реалізують методи AHP та TOPSIS.

Етап 1: визначення ваг критеріїв (AHP):

– формування матриць парних порівнянь. ОПР заповнює матриці парних порівнянь, використовуючи шкалу Сааті (від 1 до 9) для визначення відносної важливості критеріїв;

– розрахунок локальних векторів пріоритетів. Для кожної матриці A розраховується вектор локальних ваг w шляхом нормалізації її головного власного вектора, що відповідає максимальному власному значенню $\lambda_{max}$;

– перевірка узгодженості. Для кожної матриці розраховується індекс узгодженості (CI) та відношення узгодженості (CR):

$$\mathrm{CI} = \frac{\lambda_{max} - n}{n-1}, \tag{2.1}$$

$$\mathrm{CR} = \frac{\mathrm{CI}}{\mathrm{RI}}, \tag{2.2}$$

де n – розмірність матриці;

RI – випадковий індекс. Якщо CR > 0.1, судження вважаються неузгодженими;

– розрахунок глобальних ваг. Глобальна вага кожного критерію 2-го рівня обчислюється як добуток його локальної ваги на вагу відповідної групи 1-го рівня;

Етап 2: ранжування стартапу (TOPSIS):

– формування матриці рішень. Створюється матриця рішень X, яка в даному випадку є вектором-рядком $x = (x_1, x_2, \dots, x_n)$, де $x_j$ – це бальна оцінка стартапу (від 1 до 10) за j-м критерієм;

– нормалізація матриці. Матриця рішень нормалізується для приведення оцінок до діапазону [0, 1]. Оскільки використовується фіксована шкала, застосовується формула лінійної нормалізації:

$$n_{ij} = \frac{x_{ij}}{x_{max}}, \tag{2.3}$$

де $x_{max}$ = 10 (мак сильна можливо оцінка) ;

– побудова зваженої нормалізованої матриці. Елементи матриці $V = (v_{ij})$ обчислюються шляхом множення нормалізованих оцінок на глобальні ваги, отримані на етапі AHP:

$$v_{ij} = v_i \cdot r_{ij}, \quad (2.4)$$

– визначення ідеального (PIS, $A^+$) та анти-ідеального (NIS, $A^-$) рішень. Для забезпечення стабільності оцінок використовуються абсолютні точки відліку:

$$A^+ = \{v_1^+, v_2^+, \dots, v_n^+\} = \left\{\left(\max_i v_{ij} \mid j \in J_{benefit}\right), \left(\min_i v_{ij} \mid j \in J_{cost}\right)\right\}, \quad (2.5)$$

$$A^- = \{v_1^-, v_2^-, \dots, v_n^-\} = \left\{\left(\min_i v_{ij} \mid j \in J_{benefit}\right), \left(\max_i v_{ij} \mid j \in J_{cost}\right)\right\}, \quad (2.6)$$

– розрахунок відстаней. Обчислюються евклідові відстані від оцінки стартапу до ідеального ($S_i^+$) та анти-ідеального ($S_i^-$) рішень:

$$S_i^+ = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^+)^2}\,, \quad (2.7)$$

$$S_i^- = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^-)^2}\,, \quad (2.8)$$

– Розрахунок коефіцієнта близькості. Фінальна оцінка готовності стартапу ($C_i$) розраховується як коефіцієнт близькості до ідеального рішення:

$$C_i = \frac{S_i^-}{S_i^+ + S_i^-}\,, \quad (2.9)$$

Значення $C_i$ знаходиться в діапазоні (0;1). Чим ближче $C_i$ до 1, тим вища готовність стартапу до масштабування.

Класичний метод TOPSIS є компенсаторним, тобто дозволяє високим значенням одних критеріїв (наприклад, надлишкове фінансування) компенсувати низькі значення інших (наприклад, відсутність ринкового попиту). У контексті стартапів це створює ризик хибно-позитивних рекомендацій, оскільки певні диспропорції є критичними для життєздатності бізнесу.

Для усунення цього недоліку в роботі запропоновано модифікацію алгоритму шляхом введення механізму «вето-порогов».

Розрахунок фінального індексу готовності. Вводиться функція штрафу $K_{penalty}$, яка коригує базовий коефіцієнт $C_i^{base}$ у разі виявлення критичних значень ключових метрик (зокрема, Product-Market Fit):

$$C_i^{final} = C_i^{base} * K_{penalty}, \quad (2.10)$$

де коефіцієнт пеналізації визначається умовою:

$$K_{penalty} = \begin{cases} 0.6, \text{якщо } x_{pmf} < 3.0 \text{ (активація вето)} \\ 1.0, \text{якщо } x_{pmf} \geq 3.0 \end{cases}, \quad (2.11)$$

Застосування коефіцієнта 0.6 дозволяє примусово знизити інтегральний рейтинг стартапу до зони «ризику» (нижче 0.5), навіть якщо інші показники знаходяться на високому рівні.

Таким чином, розроблена математична модель поєднує гнучкість налаштування стратегії через AHP та жорсткість оцінки критичних ризиків через модифікований TOPSIS, що забезпечує високу достовірність результатів діагностики.

## Висновки до розділу 2

У другому розділі проведено теоретичне обґрунтування та розробку методичного забезпечення для створення інформаційної системи оцінки стартапів. За результатами досліджень зроблено наступні висновки:

– визначено методологічний підхід: встановлено, що задача оцінки готовності стартапу до масштабування належить до класу задач багатокритеріального прийняття рішень (MCDM) в умовах невизначеності. Для формалізації якісних експертних оцінок (лінгвістичних змінних) обґрунтовано доцільність використання елементів теорії нечітких множин із застосуванням трикутних та трапецієподібних функцій належності;

– обґрунтовано вибір математичної моделі: як базовий алгоритм обрано гібридну модель AHP-TOPSIS, що поєднує переваги двох класичних методів. Доведено, що метод аналізу ієрархій (AHP) дозволяє ефективно налаштувати стратегічні пріоритети інвестора та перевірити узгодженість суджень, а метод TOPSIS забезпечує об'єктивне ранжування проєктів на основі їхньої близькості до ідеального рішення;

– розроблено систему критеріїв: сформовано дворівневу ієрархічну модель оцінювання, яка включає 4 групи критеріїв («Продукт та Технологія», «Ринок та Бізнес-модель», «Команда та Операції», «Фінанси та Юридичні аспекти») та 8 деталізованих підкритеріїв. Така структура дозволяє комплексно охопити всі критичні аспекти життєдіяльності стартапу;

– удосконалено алгоритм оцінювання: формалізовано математичну модель та покроковий алгоритм розрахунків. Ключовою особливістю розробленої методики є модифікація класичного методу TOPSIS шляхом впровадження некомпенсаторної логіки (механізму «вето-порогов»). Введення функції штрафу та порогового значення для критичних метрик (зокрема Product-Market Fit) дозволяє усунути головний недолік лінійних методів згортки — можливість компенсації провалів у продукті за рахунок надлишкового фінансування. Це значно підвищує

достовірність діагностики та знижує ризик надання хибно-позитивних рекомендацій.

Таким чином, розроблена математична модель створює надійну теоретичну основу для програмної реалізації системи підтримки прийняття рішень, забезпечуючи гнучкість налаштувань та високу точність результатів.

# 3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ У ВИГЛЯДІ TELEGRAM-БОТА

## 3.1 Обґрунтування вибору архітектури Telegram-бота

При виборі платформи для реалізації інформаційної системи оцінки готовності стартапів було розглянуто декілька архітектурних підходів: веб-додаток, настільний (десктопний) додаток та чат-бот. Після аналізу переваг та недоліків кожного з них, для даного проєкту було обрано архітектуру Telegram-бота. Таке рішення обґрунтоване низкою ключових переваг, що відповідають меті та завданням роботи:

– кросплатформеність та доступність: telegram є месенджером, доступним на всіх основних платформах (iOS, Android, Windows, macOS, Linux) та у веб-версії. Це означає, що розроблена система буде автоматично доступною для широкого кола користувачів без необхідності створювати окремі версії для кожної операційної системи;

– відсутність потреби у встановленні: користувачам не потрібно завантажувати та встановлювати окремий додаток. Взаємодія з системою відбувається у звичному для них інтерфейсі месенджера, що значно знижує поріг входу та спрощує використання;

– простота інтерфейсу: інтерфейс чат-бота є інтуїтивно зрозумілим. Взаємодія відбувається через команди, текстові повідомлення та інтерактивні елементи (кнопки, inline-клавіатури), що дозволяє покроково вести користувача через складний процес оцінки (введення парних порівнянь, бальних оцінок), мінімізуючи ризик помилок;

– швидкість розробки та прототипування: використання готового API (Application Programming Interface) від Telegram [27] та спеціалізованих бібліотек для Python (наприклад, python-telegram-bot) дозволяє значно прискорити процес розробки, зосередившись на реалізації основної бізнес-логіки (алгоритми AHP-TOPSIS), а не на розробці GUI з нуля;

– можливості сповіщень: архітектура бота дозволяє в майбутньому реалізувати систему проактивних сповіщень, наприклад, нагадувань про необхідність оновити оцінку або повідомлень про нові рекомендації.

Таким чином, реалізація системи у вигляді Telegram-бота є оптимальним рішенням, що поєднує високу доступність для кінцевого користувача, швидкість розробки та гнучкість для подальшого розширення функціоналу.

### 3.2 Архітектура інформаційної системи

Проєктування архітектури програмного комплексу базувалося на принципах побудови розподілених клієнт-серверних систем із «тонким клієнтом». Вибір такої структури (див. рис. 3.1) обумовлений необхідністю забезпечення максимальної доступності сервісу для кінцевих користувачів через мобільні платформи (Telegram) при збереженні високої продуктивності обчислень, які виконуються на стороні сервера (Python Backend). Такий підхід дозволяє розвантажити пристрій користувача, переклавши виконання ресурсомістких математичних операцій на сервер.

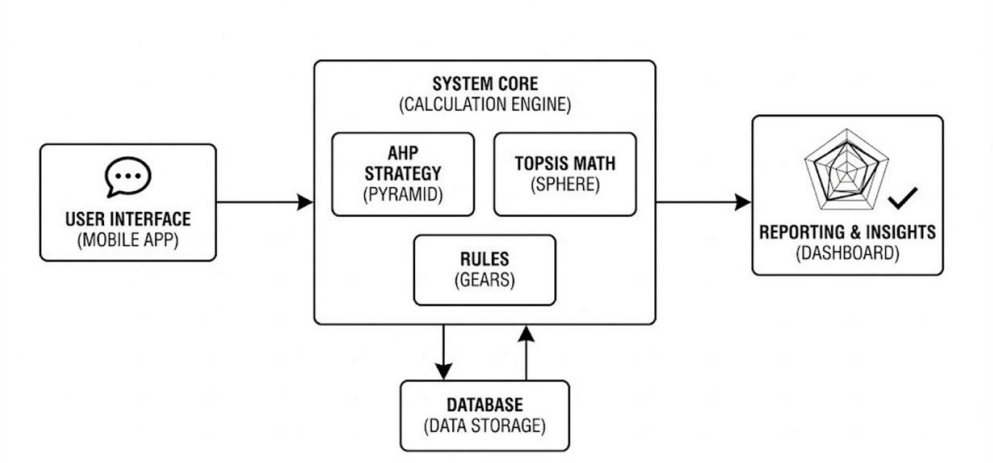


Рисунок 3.1 – Архітектура системи

Загальна концептуальна схема взаємодії компонентів та потоків даних у системі наведена нижче (див. рис. 3.2). Вона демонструє ієрархічну організацію програмних модулів та їх ізоляцію.

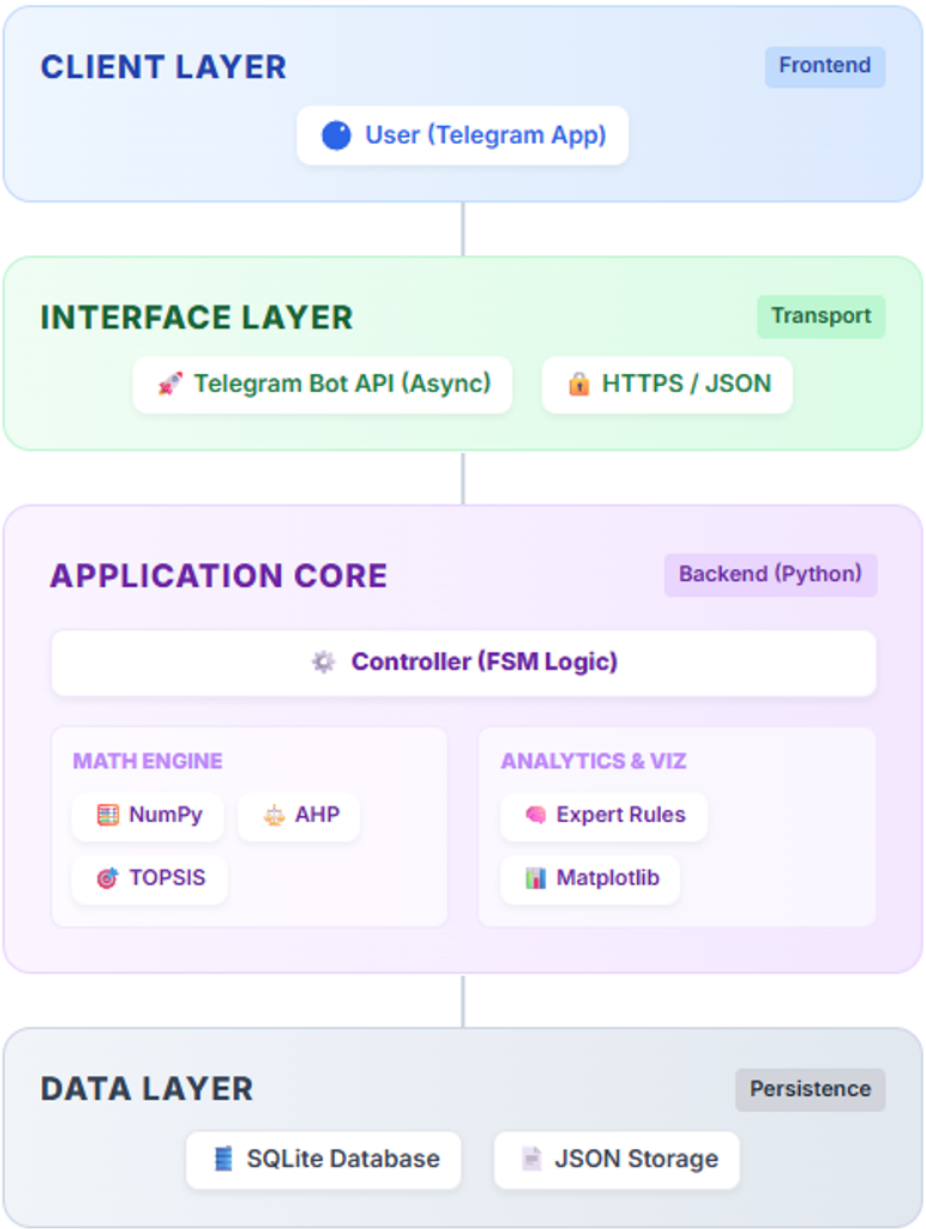


Рисунок 3.2 – Компонентна діаграма системи

Система реалізована за багатошаровою архітектурою (Layered Architecture) [37, 38], що складається з чотирьох логічних рівнів, кожен з яких має чітко визначену зону відповідальності:

– рівень клієнта (Client Layer): реалізований на базі кросплатформенного додатку Telegram. Цей рівень відповідає за презентацію даних (Frontend) та взаємодію з користувачем. Він не містить бізнес-логіки, а лише відправляє команди та відображає отримані звіти й візуалізації;

– рівень інтерфейсу (Interface Layer): забезпечує транспортний канал між клієнтом та сервером. Використовує асинхронний Telegram Bot API для обміну повідомленнями у форматі JSON. Цей шар абстрагує ядро системи від деталей мережевої взаємодії (HTTP-запитів, Webhook/Long Polling);

– рівень бізнес-логіки (Application Core): центральний елемент системи, реалізований мовою Python. Він містить головний контролер, що керує станами діалогу (FSM — Finite State Machine) та маршрутизує запити до відповідних обчислювальних модулів. Саме тут відбувається виклик алгоритмів AHP та TOPSIS, а також генерація аналітичних висновків;

– рівень даних (Data Layer): відповідає за персистентність (довготривале збереження) інформації. Реалізований на базі реляційної СУБД SQLite, що забезпечує надійне зберігання історії оцінювань, налаштувань профілів користувачів та проміжних результатів розрахунків.

Для деталізації внутрішньої структури додатку, проєктування класів та визначення взаємозв'язків між ними (залежностей, асоціацій) було розроблено діаграму класів (UML Class Diagram), яка наведена нижче (див. рис. 3.3).

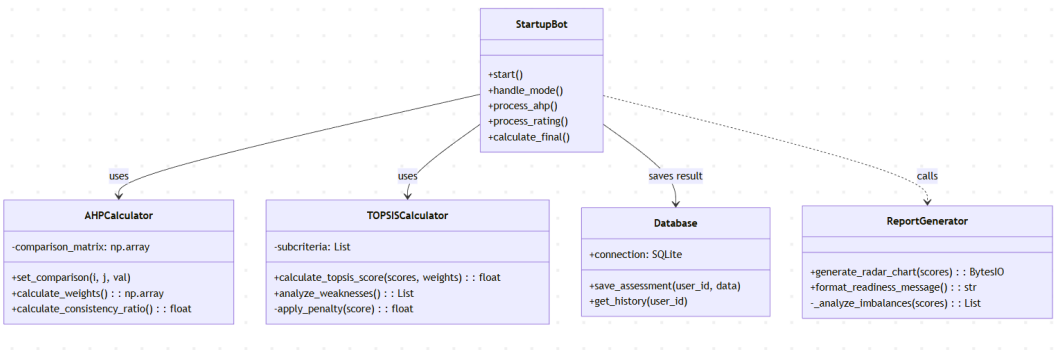


Рисунок 3.3 – Діаграма класів програмного комплексу

Діаграма відображає основні сутності системи, спроєктовані згідно з принципами об'єктно-орієнтованого програмування (ООП) [34]:

– StartupBot: головний клас-контролер (патерн Facade), що ініціалізує додаток, керує життєвим циклом бота та розподіляє вхідні події (Update) між відповідними обробниками;

– AHPCalculator: клас, що інкапсулює математичну логіку методу аналізу ієрархій. Він відповідає за формування матриць порівнянь, розрахунок власних векторів та перевірку узгодженості експертних оцінок (Consistency Ratio);

– TOPSISCalculator: клас, що реалізує алгоритм ранжування альтернатив. Його методи виконують нормалізацію матриці рішень, розрахунок евклідових відстаней до ідеальної точки та застосування штрафних коефіцієнтів (реалізація некомпенсаторної логіки);

– Database: клас для роботи з даними, реалізований за патерном Singleton. Це гарантує існування єдиного екземпляра підключення до БД у межах програми, що забезпечує безпеку транзакцій та ефективне використання ресурсів при виконанні CRUD-операцій;

– ReportGenerator: статичний клас-утиліта, відповідальний за візуалізацію даних (генерацію радарних діаграм за допомогою Matplotlib) та форматування текстових звітів для відправки користувачу.

Така архітектура забезпечує слабку зв'язність компонентів (Low Coupling) та високу модульність (High Cohesion). Це дозволяє змінювати або вдосконалювати окремі модулі (наприклад, замінити алгоритм розрахунку або СУБД) без необхідності вносити зміни в інші частини системи, що значно спрощує підтримку та масштабування програмного продукту.

### 3.3 Проєктування взаємодії з користувачем

Проєктування інтерфейсу взаємодії (User Interface Design) для системи підтримки прийняття рішень базувалося на принципах людино-орієнтованого

дизайну (Human-Centered Design) [43] та концепції розмовних інтерфейсів (Conversational UI). Оскільки процес багатокритеріального оцінювання є когнітивно складним завданням, критично важливим було мінімізувати ментальне навантаження на користувача. Для цього було обрано патерн «Wizard» (Покроковий майстер), який декомпозує складний процес на послідовність простих атомарних дій.

Точкою входу в систему є стандартна команда /start. На цьому етапі контролер бота виконує ініціалізацію профілю користувача та перевіряє наявність незавершених сесій. Для забезпечення ідемпотентності операцій, команда /new_assessment не просто починає діалог, а виконує повне очищення тимчасових буферів даних у оперативній пам'яті (скидання матриць парних порівнянь та векторів оцінок), що гарантує коректність розрахунків для нового проєкту. Система запитує ідентифікатор (назву) стартапу, який надалі використовується як тег для групування результатів у базі даних.

Етап налаштування ваг критеріїв є найбільш складним з точки зору UX, оскільки метод аналізу ієрархій вимагає проведення серії однотипних порівнянь:

– візуалізація шкали Сааті: замість ручного введення числових значень (1–9), що є незручним на мобільних пристроях, використано механізм InlineKeyboardMarkup. Інтерфейс пропонує користувачеві набір кнопок з лінгвістичними змінними (наприклад, «Лівий критерій значно важливіший»), які програмно трансформуються у відповідні числові коефіцієнти (наприклад, 5 або 1/5);

– послідовна подача: алгоритм бота автоматично генерує пари критеріїв згідно з комбінаторною формулою N(N-1)/2. Це дозволяє користувачеві фокусуватися лише на одному порівнянні в конкретний момент часу, що знижує ймовірність помилок неузгодженості суджень;

– зворотний зв'язок: після кожного вибору бот оновлює повідомлення (editMessageText), фіксуючи вибір користувача, що забезпечує візуальне підтвердження дії без захаращення історії чату.

Після визначення вагових коефіцієнтів система переходить до етапу оцінювання конкретного проєкту (TOPSIS Rating):

– контекстні підказки: для кожного з 8 критеріїв (наприклад, «Product-Market Fit») бот виводить не лише назву, а й розширений опис метрики з «якірними» значеннями для шкали 1–10 (наприклад, «1 = Тільки ідея, 10 = Стабільні продажі»). Це дозволяє стандартизувати суб'єктивні оцінки користувачів;

– валідація вводу (Input Validation): використання клавіатури з фіксованими значеннями (1–10) фізично обмежує можливість введення некоректних даних (наприклад, тексту або чисел поза діапазоном). На рівні контролера додатково реалізовано перевірку типів даних, що захищає математичне ядро від виключних ситуацій.

Фінальний етап взаємодії спрямований на інтерпретацію отриманих результатів. Замість виведення "сухих" цифр, система генерує комплексний мультимедійний звіт:

– візуалізація: бот надсилає згенероване зображення (PNG) радарної діаграми, яка накладає профіль поточного проєкту на еталонний профіль ("Benchmark"). Це дозволяє користувачеві миттєво візуально ідентифікувати слабкі сторони ("провали" на графіку);

– інтерпретація: інтегральний індекс TOPSIS супроводжується текстовим вердиктом (Status), який визначається на основі порогових правил бази знань (наприклад, «NEEDS OPTIMIZATION» при значенні індексу 0.5–0.75);

– рекомендації: система автоматично ранжує критерії за ступенем їх негативного впливу на загальний результат та формує список пріоритетних дій для покращення стану стартапу.

Така організація інтерфейсу забезпечує високу ергономічність системи, дозволяючи проводити складний математичний аналіз у інтуїтивно зрозумілій формі месенджера, що є критичним фактором для цільової аудиторії (засновників стартапів), яка цінує мобільність та швидкість прийняття рішень.

## 3.4 Вибір інструментів та технологій розробки

Обґрунтування вибору технологічного стеку для реалізації інформаційної системи базувалося на комплексному аналізі функціональних та нефункціональних вимог, серед яких ключовими були швидкість розробки прототипу, надійність математичних обчислень, можливість горизонтального масштабування та наявність розвиненої екосистеми бібліотек для наукового аналізу даних. Головним критерієм вибору стала необхідність ефективної реалізації складної математичної логіки методів AHP та TOPSIS у легкому та доступному для кінцевого користувача інтерфейсі Telegram-бота.

Як основний інструмент розробки серверної частини було обрано мову програмування Python версії 3.10 або вище [26]. Цей вибір зумовлений насамперед фактичним домінуванням Python у сфері Data Science та наукових обчислень. Наявність високооптимізованих бібліотек дозволяє реалізувати складні матричні операції, необхідні для методу аналізу ієрархій, такі як пошук власних векторів та власних чисел, з максимальною обчислювальною ефективністю, що було б значно складніше та ресурсозатратніше реалізувати на інших мовах програмування загального призначення, таких як Java або C#. Крім того, лаконічність синтаксису та динамічна типізація Python дозволяють значно скоротити час на написання бізнес-логіки та налагодження алгоритмів. Важливим фактором також стала вбудована підтримка асинхронного програмування у сучасних версіях мови, що є критично важливим для задач, пов’язаних з інтенсивним вводом-виводом, зокрема для обробки тисяч одночасних запитів до API месенджера.

Для реалізації взаємодії з Telegram Bot API було обрано бібліотеку python-telegram-bot [28]. У процесі проєктування розглядалися альтернативні варіанти, такі як aiogram та pyTelegramBotAPI, проте вибір було зроблено на користь першої завдяки наявності високорівневих абстракцій. Зокрема, використання класу ConversationHandler дозволяє реалізувати патерн «Кінцевий автомат» для організації покрокового збору оцінок від користувача, ефективно керуючи станами

діалогу без необхідності ручного збереження контексту в базі даних на кожному кроці. Архітектура цієї бібліотеки повністю відповідає принципам об'єктно-орієнтованого програмування, що спрощує підтримку коду та його подальше розширення, а велика спільнота розробників гарантує стабільність та оперативну сумісність з оновленнями платформи Telegram.

Організація збереження даних у системі реалізована за гібридною схемою, що враховує різні етапи життєвого циклу програмного забезпечення. Для етапу розробки та тестування MVP оптимальним рішенням стала вбудована реляційна СУБД SQLite [31]. Її головною перевагою є відсутність необхідності в адмініструванні окремого сервера та збереження всієї бази в одному файлі, що дозволяє легко переносити проєкт між середовищами розробки та спрощує процедури резервного копіювання. Водночас архітектура системи спроєктована з урахуванням майбутнього масштабування та переходу на промислову експлуатацію, що передбачає безшовну міграцію на PostgreSQL. Ця об'єктно-реляційна СУБД забезпечує вищу надійність, підтримує велику кількість одночасних транзакцій та має розвинені засоби для роботи з JSON-структурами, що є критично важливим для ефективного збереження векторів експертних оцінок.

Для програмної реалізації математичного ядра та роботи з даними було залучено спеціалізовані бібліотеки. Зокрема, фундаментальна бібліотека NumPy [29] використовується для роботи з багатовимірними масивами та матрицями, забезпечуючи векторизацію обчислень, що робить розрахунок рейтингів TOPSIS у десятки разів швидшим порівняно з використанням стандартних циклів. Візуалізація результатів покладена на бібліотеку Matplotlib [30], яка використовується для динамічної генерації радарних діаграм, що наочно демонструють профіль готовності стартапу. Взаємодія з базою даних здійснюється через ORM SQLAlchemy, що дозволяє працювати з даними через об'єкти Python, абстрагуючись від синтаксису SQL-запитів. Такий підхід не лише підвищує безпеку, забезпечуючи автоматичний захист від SQL-ін’єкцій, але й значно

спрощує міграцію між різними типами СУБД без необхідності внесення змін у програмний код.

Як основне інтегроване середовище розробки обрано Visual Studio Code, перевагами якого є легкість, кросплатформеність та наявність потужних розширень для статичного аналізу коду та налагодження. Управління версіями коду здійснюється за допомогою розподіленої системи Git, що є індустріальним стандартом для командної розробки та дозволяє ефективно відстежувати історію змін, створювати гілки для тестування нових функцій та забезпечувати цілісність коду при розгортанні. Таким чином, обраний технологічний стек є сучасним, збалансованим та повністю відповідає поставленим завданням, забезпечуючи високу продуктивність математичних обчислень та зручність користувацького інтерфейсу.

**Висновки до розділу 3**

У третьому розділі виконано комплексне проєктування архітектури та інтерфейсу інформаційної системи підтримки прийняття рішень для оцінки стартапів. За результатами проведених робіт зроблено наступні висновки:

– обґрунтовано вибір платформи: на основі порівняльного аналізу визначено, що реалізація системи у вигляді Telegram-бота є найбільш ефективним рішенням. Такий підхід забезпечує кросплатформеність, мінімізує бар’єри входу для користувачів (відсутність необхідності встановлення додаткового ПЗ) та дозволяє використовувати нативні механізми взаємодії месенджера для реалізації складних сценаріїв діалогу;

– спроєктовано архітектуру системи: розроблено багаторівневу клієнт-серверну архітектуру, побудовану за принципами слабкої зв'язності (Low Coupling). Виділено чотири логічні рівні: клієнтський, інтерфейсний, рівень бізнес-логіки та рівень даних. Для реалізації внутрішньої логіки застосовано об'єктно-орієнтований підхід та патерни проєктування (Facade, Singleton), що забезпечує

масштабованість системи та можливість незалежної модернізації окремих модулів (наприклад, заміни математичного ядра або СУБД);

– розроблено UX-сценарій: спроєктовано ергономічний інтерфейс користувача на основі патерну «Wizard» (покроковий майстер). Запропонований сценарій взаємодії, керований кінцевим автоматом (Finite State Machine), дозволяє декомпозувати складний процес багатокритеріального оцінювання на послідовність простих кроків. Використання інтерактивних елементів (inline-клавіатур) та механізмів валідації вводу гарантує цілісність вхідних даних для математичних алгоритмів;

– визначено технологічний стек: обґрунтовано вибір сучасного набору інструментів для реалізації системи. Ядром серверної частини обрано мову Python з використанням асинхронного фреймворку python-telegram-bot для обробки високого навантаження. Для реалізації математичних методів AHP/TOPSIS залучено бібліотеку NumPy, що забезпечує високу продуктивність векторних обчислень. Система збереження даних спроєктована за гібридною схемою (SQLite/PostgreSQL), що враховує потреби як етапу прототипування, так і промислової експлуатації.

Таким чином, розроблені проєктні рішення створюють надійний фундамент для програмної реалізації системи, забезпечуючи баланс між зручністю для користувача, точністю математичних розрахунків та технічною ефективністю виконання.

# 4 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 4.1 Реалізація математичного ядра системи (Модулі розрахунків)

Програмна реалізація системи виконана у вигляді набору незалежних мікромодулів, кожен з яких інкапсулює окрему частину бізнес-логіки. Такий підхід відповідає принципам об'єктно-орієнтованого програмування (ООП) та SOLID [33], що дозволяє легко масштабувати систему, додаючи нові методи оцінки без зміни існуючого коду.

Ядро системи складається з трьох ключових компонентів, реалізованих у файлах ahp.py, topsis.py та utils.py.

Модуль стратегічного планування (ahp.py) відповідає за обробку матриць парних порівнянь. Основним класом є AHPCalculator. Для забезпечення високої точності обчислень, особливо при знаходженні власних векторів матриць, використано бібліотеку наукових обчислень NumPy.

Ключовим методом класу є функція calculate_weights, яка реалізує алгоритм Сааті. Замість наближених методів (наприклад, середнього геометричного), у роботі використано точний метод пошуку головного власного вектора через функцію numpy.linalg.eig.

Фрагмент коду, що реалізує цей алгоритм, наведено нижче (див. рис. 4.1).

```
def calculate_weights(self) -> np.ndarray:
    eigenvalues, eigenvectors = np.linalg.eig(self.comparison_matrix)
    max_idx = np.argmax(eigenvalues.real)
    self.lambda_max = eigenvalues[max_idx].real

    principal_eigenvector = eigenvectors[:, max_idx]
    principal_eigenvector = np.abs(np.real(principal_eigenvector))

    self.weights = principal_eigenvector / principal_eigenvector.sum()
    return self.weights
```

Рисунок 4.1 – Програмна реалізація методу розрахунку вагових коефіцієнтів (клас AHPCalculator)

Як видно з лістингу, алгоритм виконує наступні кроки:

– обчислення власних чисел та векторів матриці порівнянь;

– визначення індексу максимального власного числа (max_idx), що відповідає головній компоненті

– виділення відповідного власного вектора (principal_eigenvector);

– взяття модуля дійсних частин чисел для уникнення помилок, пов'язаних з комплексними числами, які можуть виникати при чисельних методах розв'язання;

– нормалізація вектора, щоб сума всіх ваг дорівнювала одиниці.

Також у цьому модулі реалізовано клас ExpertProfiles (див. рис. 4.2), який зберігає словник попередньо налаштованих стратегій (бенчмарків) для різних типів бізнесу. Це дозволяє системі працювати у двох режимах: ручному та автоматичному.

```
class ExpertProfiles:
    @staticmethod
    def get_profiles() -> Dict[str, Dict]:
        return {
            "b2b_saas": {
                "name": "💎 B2B SaaS / Enterprise",
                "desc": "Стратегія: Акцент на продажах та утриманні.",
                "weights": [0.30, 0.40, 0.20, 0.10]
            },
            "marketplace": {
                "name": "🌐 Marketplace / Platform",
                "desc": "Стратегія: Ефект мережі та масштабування.",
                "weights": [0.15, 0.45, 0.25, 0.15]
            }
        }
```

Рисунок 4.2 – Структура збереження експертних профілів

Модуль оцінки та ранжування (topsis.py) містить клас TOPSISCalculator, який реалізує модифікований алгоритм TOPSIS. Ключовою особливістю даної реалізації є відмова від повільних ітеративних циклів Python на користь векторизованих

операцій над масивами numpy.ndarray. Це дозволяє обробляти великі масиви даних миттєво.

Метод calculate_topsis_score виконує нормалізацію оцінок користувача, зважування матриці та розрахунок евклідових відстаней до ідеального ($PIS$) та анти-ідеального ($NIS$) рішень. Оскільки всі вхідні дані знаходяться у фіксованому діапазоні (1-10), застосовано лінійну нормалізацію для збереження масштабу даних.

Особливу увагу приділено реалізації механізму "вето" (некомпенсаторної логіки), про яку йшлося у другому розділі. Фрагмент коду, що відповідає за розрахунок рейтингу та застосування штрафних санкцій, наведено нижче (див. рис. 4.3).

```
def calculate_topsis_score(self, user_scores: np.ndarray, criteria_weights: np.ndarray):
    normalized_scores = user_scores / 10.0
    ideal_norm = np.ones(self.n)
    anti_ideal_norm = np.full(self.n, 0.1)

    local_weights = np.zeros(self.n)
    group_counts = np.bincount(self.criteria_groups)
    for i, group_idx in enumerate(self.criteria_groups):
        local_weights[i] = criteria_weights[group_idx] / group_counts[group_idx]

    weighted_user = normalized_scores * local_weights
    weighted_ideal = ideal_norm * local_weights
    weighted_anti_ideal = anti_ideal_norm * local_weights

    d_plus = np.sqrt(np.sum((weighted_user - weighted_ideal) ** 2))
    d_minus = np.sqrt(np.sum((weighted_user - weighted_anti_ideal) ** 2))
    score = d_minus / (d_plus + d_minus) if (d_plus + d_minus) != 0 else 0

    pmf_index = 2
    if user_scores[pmf_index] < 3.0:
        score *= 0.6

    return score, normalized_scores, {}
```

Рисунок 4.3 – Реалізація алгоритму TOPSIS з логікою штрафів

У наведеному фрагменті видно, що після розрахунку базового коефіцієнта score, система перевіряє значення критичної метрики (у даному випадку Product-

Market Fit). Якщо оцінка за цим критерієм нижча за порогове значення (3.0 бали), до фінального результату застосовується понижуючий коефіцієнт 0.6. Це програмно гарантує, що стартап зі слабким продуктом не отримає високий рейтинг лише за рахунок фінансів.

Третій компонент ядра – модуль utils.py, що відповідає за візуалізацію. Для побудови радарних діаграм використано бібліотеку Matplotlib.

Важливим архітектурним рішенням є використання бекенду Agg (matplotlib.use('Agg')). Стандартний режим роботи Matplotlib намагається створити вікно графічного інтерфейсу, що неможливо на сервері без монітора. Режим Agg дозволяє рендерити зображення безпосередньо в буфер оперативної пам'яті.

Код генерації діаграми наведенонижче (див. рис. 4.4).

```
@staticmethod
def generate_radar_chart(categories: List[str], user_scores: np.ndarray) -> io.BytesIO:
    labels = categories
    num_vars = len(labels)
    angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

    values = np.concatenate((user_scores, [user_scores[0]]))
    angles += angles[:1]
    ideal = np.full(num_vars + 1, 10.0)

    fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'))
    ax.plot(angles, ideal, color='#2ECC71', linewidth=1, linestyle='--')
    ax.plot(angles, values, color='#2980B9', linewidth=2.5, linestyle='solid')
    ax.fill(angles, values, color='#3498DB', alpha=0.35)

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(labels, size=10, weight='bold')

    buf = io.BytesIO()
    plt.tight_layout()
    plt.savefig(buf, format='png', dpi=150, bbox_inches='tight')
    buf.seek(0)
    plt.close()
    return buf
```

Рисунок 4.4 – Метод генерації радарної діаграми в буфер пам'яті

Використання об'єкта io.BytesIO дозволяє передавати згенероване зображення безпосередньо в Telegram API, оминаючи повільні операції запису та

читання файлів на жорсткому диску (Disk I/O), що значно підвищує швидкодію системи при великій кількості одночасних запитів.

### 4.2 Модуль аналітики, візуалізації та генерації звітів

Ключовою перевагою розробленої системи над простими калькуляторами оцінок є наявність інтелектуального модуля аналітики, реалізованого у файлі utils.py. Цей модуль відповідає за інтерпретацію числових результатів оцінювання, візуалізацію даних та формування текстових рекомендацій, зрозумілих для кінцевого користувача. Архітектурно модуль побудований як набір статичних методів класу ReportGenerator, що забезпечує їх легкий виклик з будь-якої частини програми без необхідності створення екземплярів класу.

Однією з найскладніших задач при оцінці стартапів є виявлення прихованих структурних дисбалансів, коли високі показники за одними критеріями маскують критичні провали за іншими. Для вирішення цієї проблеми у системі реалізовано механізм евристичного аналізу, який базується на продукційних правилах.

Метод _analyze_imbalances отримує на вхід вектор оцінок користувача і послідовно перевіряє його на відповідність низці заздалегідь визначених шаблонів ризику. Алгоритм працює за принципом перехресного аналізу (Cross-Impact Analysis). Наприклад, для виявлення ризику передчасного масштабування ("Premature Scaling") система не просто дивиться на низький бал "Product-Market Fit", а аналізує його у поєднанні з показником "Фінансова стійкість". Якщо фінансовий ресурс є значним (понад 7 балів), а відповідність ринку низькою (менше 5 балів), це активує відповідне правило, оскільки така комбінація неминуче призводить до неефективного витрачання бюджету.

Аналогічним чином реалізовано виявлення інших загроз. Ризик "Technology Push" діагностується у випадку значного розриву між технічною зрілістю продукту та підтвердженим попитом, що характерно для інженерних команд, які ігнорують етап Customer Development. Ризик "Zombie Team" фіксується, коли потенціал ринку оцінюється високо, але компетенції команди є недостатніми для його захоплення.

Для кожного виявленого ризику система формує текстовий опис проблеми та конкретну рекомендацію щодо її усунення, які додаються до списку інсайтів.

Фрагмент програмного коду, що реалізує цю логіку, наведено нижче (див. рис. 4.5).

```
@staticmethod
    def _analyze_imbalances(user_scores: np.ndarray) -> List[str]:
        insights = []
        tech, mvp, pmf, tam, team, ops, fin, legal = user_scores

        # 1. Technology Push
        if tech > 7 and pmf < 4:
            insights.append("⚡ Ризик 'Technology Push': Технологія без попиту.")

        # 2. Premature Scaling
        if fin > 7 and pmf < 5:
            insights.append("🔥 Ризик 'Premature Scaling': Бюджет витрачається даремно.")

        # 3. Growth Trap
        if pmf > 7 and fin < 4:
            insights.append("💸 Ризик 'Growth Trap': Є попит, але немає ресурсів.")

        # 4. Zombie Team
        if tam > 7 and team < 5:
            insights.append("🕵️ Ризик виконавця: Ринок більший за можливості команди.")

        # 5. Operational Chaos
        if pmf > 6 and fin > 6 and ops < 4:
            insights.append("🌪️ Ризик 'Operational Chaos': Хаос при масштабуванні.")

        return insights
```

Рисунок 4.5 – Фрагмент реалізації бази знань експертної системи

Перелік основних правил, закладених у базу знань системи, наведено у Таблиці 4.1.

Таблиця 4.1 – Правила виявлення структурних ризиків стартапу

| **Логічна умова (Trigger)** | **Діагноз системи** | **Стратегічна рекомендація** |
|---|---|---|
| Tech > 7 AND Market < 4 | Ризик «Пастка Інженера» (Technology Push) | Виявлено значний перекіс у бік розробки без підтвердженого попиту. Рекомендовано тимчасово зупинити розробку (Code Freeze) та перенаправити ресурси на дослідження клієнтів (CustDev). |
| Finance > 7 AND Ops < 5 | Ризик «Масштабування Хаосу» (Scaling Chaos) | Наявність значного бюджету при неналагоджених операційних процесах призведе до неефективного витрачання коштів. Необхідно терміново описати стандартні операційні процедури (SOP) та впровадити CRM/ERP системи. |
| BizModel > 8 AND PMF < 3 | Ризик «Галюцинація» (Fantasy Startup) | Існує теоретично привабливa бізнес-модель, яка не підкріплена реальним інтересом ринку до продукту. Необхідно повернутися на етап валідації проблеми клієнта. |
| Team > 8 AND PMF < 4 | Ризик демотивації команди | Висококваліфікована команда працює над продуктом, який не має ринкового успіху. Існує висока ймовірність втрати ключових співробітників. Рекомендовано розглянути можливість зміни бізнес-моделі (Pivot). |
| BizModel > 8 AND Legal < 4 | Юридична вразливість | Успішна бізнес-модель не має належного правового захисту. Критично важливо провести аудит інтелектуальної власності (IP) та оформити договори. |

Для забезпечення наочності результатів система генерує радарну діаграму (Radar Chart), яка дозволяє миттєво оцінити збалансованість проєкту. За цю функцію відповідає метод generate_radar_chart.Процес побудови графіка починається з математичної підготовки даних. Оскільки бібліотека Matplotlib будує полярні графіки на основі кутових координат, вхідний масив оцінок спершу

конвертується у відповідний формат. Кути для осей розраховуються шляхом поділу повного кола на кількість критеріїв.

Важливою деталлю є необхідність "замкнути" контур графіка, для чого до масиву значень та кутів програмно додається перший елемент у кінець списку. Без цього кроку лінія графіка залишилася б розірваною між останньою та першою віссю.

Особливу увагу при розробці було приділено оптимізації роботи з пам'яттю. Оскільки система працює у серверному оточенні, запис кожного згенерованого зображення на жорсткий диск був би вкрай неефективним рішенням, що сповільнювало б роботу при великій кількості користувачів. Тому у коді використано клас io.BytesIO зі стандартної бібліотеки Python. Згенероване зображення зберігається безпосередньо в буфер оперативної пам'яті у форматі PNG, звідки воно миттєво передається до API Telegram. Також для коректної роботи на сервері без графічного інтерфейсу (headless environment) бібліотека Matplotlib примусово перемикається на використання бекенду Agg, що запобігає помилкам, пов'язаним зі спробою відкрити вікно перегляду зображення.

Лістинг коду, що відповідає за візуалізацію, наведено нижче (див. рис. 4.6).

```python
@staticmethod
    def generate_radar_chart(categories: List[str], user_scores: np.ndarray) -> io.BytesIO:
        num_vars = len(categories)
        angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

        values = np.concatenate((user_scores, [user_scores[0]]))
        angles += angles[:1]
        ideal = np.full(num_vars + 1, 10.0)

        fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'))

        ax.plot(angles, ideal, color='#2ECC71', linewidth=1, linestyle='--')
        ax.plot(angles, values, color='#2980B9', linewidth=2.5)
        ax.fill(angles, values, color='#3498DB', alpha=0.35)

        ax.set_xticks(angles[:-1])
        ax.set_xticklabels(categories, size=10, weight='bold')

        buf = io.BytesIO()
        plt.savefig(buf, format='png', dpi=150, bbox_inches='tight')
        buf.seek(0)
        plt.close()
        return buf
```

Рисунок 4.6 – Метод генерації радарної діаграми

Завершальним етапом роботи модуля аналітики є агрегація всіх отриманих даних у єдиний структурований звіт. Метод format_readiness_message приймає на вхід розрахований інтегральний показник TOPSIS, список виявлених "вузьких місць", згенеровані інсайти та налаштування стратегії.

На першому кроці відбувається категоризація результату. Залежно від значення фінального показника (який знаходиться в діапазоні від 0 до 1), стартапу присвоюється один з трьох статусів: "READY TO SCALE" (Готовий до масштабування), "NEEDS OPTIMIZATION" (Потребує оптимізації) або "TOO EARLY" (Занадто рано). Кожен статус супроводжується відповідним набором рекомендацій та візуальним індикатором (емодзі).

Далі алгоритм формує текстову візуалізацію прогресу (Progress Bar) з використанням символів Unicode. Це дозволяє користувачеві візуально оцінити свій рейтинг навіть без відкриття зображення. Наступним блоком до повідомлення додаються результати роботи евристичного аналізатора. Якщо критичних ризиків

не виявлено, система додає позитивне повідомлення про гармонійний розвиток проєкту. У випадку низьких оцінок система автоматично виділяє топ-3 найслабші метрики та виводить їх окремим списком, фокусуючи увагу засновника на пріоритетних зонах розвитку.

Код методу формування звіту представлено нижче (див. рис. 4.7).

```
@staticmethod
    def format_readiness_message(score: float, weaknesses, mode_name) -> str:
        if score >= 0.75:
            icon, status = "🚀", "READY TO SCALE"
        elif score >= 0.50:
            icon, status = "⚠️", "NEEDS OPTIMIZATION"
        else:
            icon, status = "🔴", "TOO EARLY"

        bar = "▓" * int(score * 20) + "░" * (20 - int(score * 20))

        msg = f"""
        <b>📊 ЗВІТ ГОТОВНОСТІ СТАРТАПУ</b>
        ━━━━━━━━━━━━━━━━━━━━
        <b>Статус:</b> {icon} {status}
        <b>Індекс:</b> {score:.1%} <code>[{bar}]</code>
        <b>Стратегія:</b> {mode_name}
        """
        return msg
```

Рисунок 4.7 – Логіка формування текстового звіту

Така модульна структура забезпечує гнучкість системи: логіку прийняття рішень, візуалізацію та текстове представлення рознесено по окремих функціональних блоках, що дозволяє вдосконалювати кожен з них незалежно один від одного.

### 4.3 Архітектура контролера та взаємодія з Telegram API

Центральним елементом серверної частини системи є модуль контролера bot.py, який виступає посередником між інтерфейсом користувача (Telegram) та

обчислювальним ядром системи. Його головна задача – керувати потоком діалогу, зберігати проміжні стани сесії та маршрутизувати запити до відповідних обробників.

Як технологічну основу для реалізації контролера обрано бібліотеку python-telegram-bot (версія 20+), яка є обгорткою над Telegram Bot API. Вибір саме цієї бібліотеки зумовлений її повною підтримкою асинхронного програмування (asyncio), що є критично важливим для забезпечення масштабованості сервісу.

Взаємодія з ботом не є лінійною: вона залежить від контексту (наприклад, повідомлення "10" може означати оцінку за критерієм, а може – вибір пункту меню). Для вирішення цієї проблеми архітектура контролера побудована на патерні «Кінцевий автомат» (Finite State Machine, FSM).

У програмному коді цей патерн реалізовано через клас ConversationHandler. Цей клас відстежує поточний стан кожного користувача (user_id) і визначає, який саме набір функцій-обробників має реагувати на наступне повідомлення.

Визначено наступні стани автомата:

- CHOOSE_MODE: стан очікування вибору стратегії (користувач обирає між ручним режимом AHP та шаблоном);
- AHP_COMPARING: циклічний стан, у якому користувачеві послідовно пропонуються пари критеріїв для порівняння;
- TOPSIS_RATING: стан збору кількісних оцінок за шкалою 1–10. Кожен запит супроводжується контекстною підказкою для підвищення об'єктивності оцінки;
- SHOW_RESULTS: фінальний стан, у якому генерується та надсилається звіт, після чого сесія завершується або перезапускається.

Ініціалізація ConversationHandler та реєстрація точок входу наведена нижче (див. рис. 4.8).

```python
def main():
    app = Application.builder().token(TOKEN).build()

    conv_handler = ConversationHandler(
        entry_points=[CommandHandler("start", bot.start)],
        states={
            CHOOSE_MODE: [
                CallbackQueryHandler(bot.handle_mode, pattern="^mode_"),
                CallbackQueryHandler(bot.process_profile, pattern="^profile_")
            ],
            AHP_COMPARING: [
                CallbackQueryHandler(bot.process_ahp, pattern="^cmp_"),
                CallbackQueryHandler(bot.show_pair, pattern="^next_pair$")
            ],
            TOPSIS_RATING: [
                CallbackQueryHandler(bot.process_rating, pattern="^rate_")
            ]
        },
        fallbacks=[CommandHandler("cancel", bot.cancel)]
    )

    app.add_handler(conv_handler)
    app.run_polling()
```

Рисунок 4.8 – Конфігурація кінцевого автомата

Для забезпечення високої продуктивності система використовує асинхронний цикл подій (Event Loop). Ключові функції контролера оголошені з ключовим словом async, що дозволяє інтерпретатору Python не блокувати виконання програми під час очікування I/O операцій (наприклад, запиту до бази даних або відправки фотографії в Telegram).

Метод отримання оновлень реалізовано через технологію Long Polling (app.run_polling()). Бот відкриває з'єднання з сервером Telegram і тримає його відкритим до появи нових повідомлень, що забезпечує миттєву реакцію на дії користувача без необхідності налаштування складних Webhook-серверів та SSL-сертифікатів на етапі розгортання.

Важливим аспектом є збереження контексту користувача (user_data). Оскільки HTTP-протокол (на якому базується Telegram API) є stateless (без

збереження стану), бот використовує словник user_storage для тимчасового зберігання даних сесії в оперативній пам'яті. Структура контексту включає:

– екземпляр калькулятора AHP;

– поточні індекси ітераторів (яку пару порівнюємо, який критерій оцінюємо);

– накопичені масиви оцінок (topsis_scores);

– обрану стратегію та ваги.

Фрагмент коду, що демонструє обробку вибору режиму та ініціалізацію контексту, наведено нижче (див. рис. 4.9).

```python
async def handle_mode(self, update: Update, context: ContextTypes.DEFAULT_TYPE) -> int:
    query = update.callback_query
    await query.answer()

    if query.data == "mode_manual":
        # Ініціалізація ручного режиму
        await query.edit_message_text(
            "📐 <b>Етап 1: AHP (Analytic Hierarchy Process)</b>\n"
            "Визначимо стратегічні пріоритети вручну.",
            reply_markup=InlineKeyboardMarkup([[
                InlineKeyboardButton("🚀 Почати", callback_data="next_pair")
            ]]),
            parse_mode='HTML'
        )
        return AHP_COMPARING
    else:
        # Завантаження меню шаблонів
        kb = []
        for k, v in ExpertProfiles.get_profiles().items():
            kb.append([InlineKeyboardButton(v['name'], callback_data=f"profile_{k}")])

        await query.edit_message_text(
            "📁 <b>Оберіть бізнес-модель:</b>",
            reply_markup=InlineKeyboardMarkup(kb),
            parse_mode='HTML'
        )
        return CHOOSE_MODE
```

Рисунок 4.9 – Асинхронний обробник вибору режиму роботи

Для забезпечення надійності роботи системи впроваджено механізм логування через стандартну бібліотеку logging. Усі критичні дії (старт бота, помилки бази даних, виключення при розрахунках) фіксуються у консолі та лог-файлі з часовими мітками. Це дозволяє адміністратору системи швидко діагностувати проблеми та відстежувати активність користувачів.

Така архітектура контролера забезпечує стабільну роботу додатку, чітке розділення логіки діалогу та бізнес-логіки, а також високу швидкість реакції інтерфейсу.

### 4.4 Організація збереження даних

Будь-яка інформаційна система підтримки прийняття рішень вимагає надійного механізму персистентності даних (Data Persistence) для збереження результатів роботи між сесіями користувача та перезапусками сервера. У даному проєкті за цей функціонал відповідає модуль database.py, який реалізує шар доступу до даних (Data Access Layer, DAL).

В якості системи керування базами даних (СУБД) було обрано SQLite. Це вбудована реляційна база даних, яка не потребує окремого сервера для роботи, оскільки двигун бази інтегрується безпосередньо у програму.

Вибір SQLite для архітектури Telegram-бота обумовлений наступними факторами:

– atomic commit: SQLite підтримує повну відповідність стандарту ACID (Atomicity, Consistency, Isolation, Durability), що гарантує цілісність даних навіть у випадку раптового збою живлення або помилки сервера.

– zero configuration: відсутність необхідності налаштування портів, користувачів та прав доступу значно спрощує розгортання системи (Deployment). База даних – це просто файл на диску (startup_assess.db).

– висока продуктивність: для навантажень, характерних для аналітичних ботів (тисячі запитів на день), SQLite працює швидше за клієнт-серверні бази

(MySQL/PostgreSQL) завдяки відсутності накладних витрат на мережеву передачу даних.

Клас Database спроєктовано з використанням патерну Singleton [32]. Це гарантує, що у програмі завжди існуватиме лише один екземпляр підключення до бази даних, що запобігає конфліктам при одночасному записі даних з різних потоків.

При ініціалізації системи автоматично виконується DDL-запит (Data Definition Language) CREATE TABLE IF NOT EXISTS, який формує структуру таблиці assessments. Ця таблиця діє як журнал подій, зберігаючи історію всіх оцінювань.

Схема даних включає наступні атрибути:

– id (INTEGER PRIMARY KEY): унікальний ідентифікатор запису (автоінкремент);

– user_id (INTEGER): ідентифікатор користувача Telegram для прив'язки даних;

– date (TIMESTAMP): часова мітка проведення оцінки;

– mode (TEXT): обрана стратегія (наприклад, "manual" або "template_b2b");

– final_score (REAL): інтегральний показник готовності (число з плаваючою крапкою);

Графічне представлення інформаційної моделі наведено нижче (див. рис. 4.10).

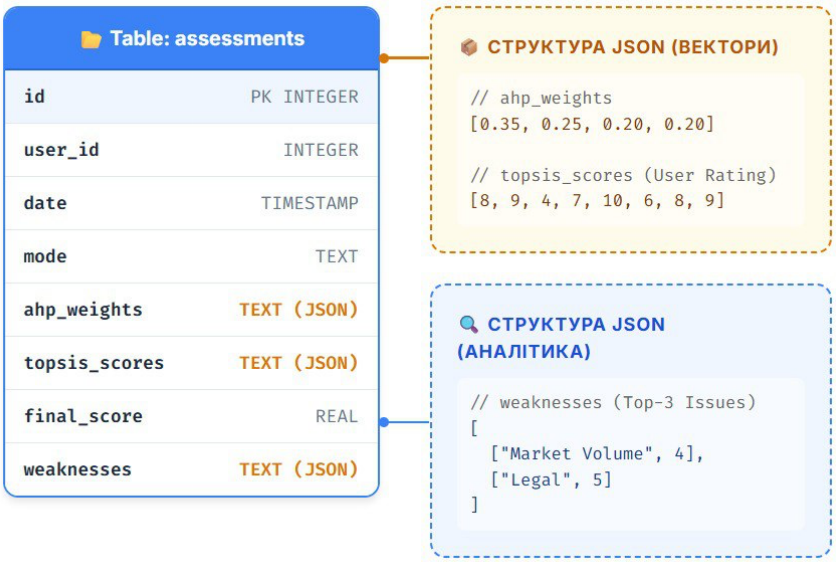


Рисунок 4.10 – Інформаційна модель бази даних системи

Однією з технічних проблем при роботі з реляційними базами даних є складність зберігання масивів або списків змінної довжини. Класична теорія баз даних вимагає створення окремих таблиць зв'язку (нормалізація), що ускладнює структуру та сповільнює запити.

У даній роботі застосовано сучасний гібридний підхід. Вектори вагових коефіцієнтів (AHP Weights), масиви оцінок користувача (TOPSIS Scores) та список виявлених ризиків зберігаються в таблиці як текстові поля у форматі JSON.

У методі save_assessment реалізовано механізм серіалізації: перед записом у БД об'єкти Python (списки, словники, масиви NumPy) конвертуються у JSON-рядок за допомогою бібліотеки json. Приклад конвертації:

– вхідні дані (NumPy array): np.array([0.3, 0.4, 0.3]);
– збережені дані (TEXT): "[0.3, 0.4, 0.3]".

Цей підхід дозволяє зберігати повний контекст сесії оцінювання в одному записі таблиці, що значно спрощує процедуру читання та аналізу історії. Відображення цього процесу у програмній реалізації показано нижче (див. рис. 4.11).

```
# Серіалізація масивів NumPy у JSON для зберігання в текстовому полі БД
weights_json = json.dumps(ahp_weights.tolist())
scores_json = json.dumps(topsis_scores.tolist())
```

Рисунок 4.11 – Відображення програмної реалізації системи

Така архітектура бази даних є гнучкою: якщо в майбутньому зміниться кількість критеріїв оцінки, не потрібно буде змінювати схему бази даних (виконувати складні міграції), оскільки JSON-поля автоматично адаптуються під нову структуру даних. Це створює надійну основу для подальшого розвитку системи та впровадження функціоналу аналізу динаміки розвитку стартапу.

### 4.5 Реалізація інтерфейсу та сценарій взаємодії з користувачем

Розробка користувацького інтерфейсу (UI) для систем підтримки прийняття рішень вимагає особливого підходу. Оскільки цільовою аудиторією системи є засновники стартапів та інвестори – люди з обмеженим часом, – інтерфейс було спроєктовано з урахуванням принципів мінімалізму, доступності та зниження когнітивного навантаження.

Використання платформи Telegram дозволило відмовитися від розробки окремого графічного фронтенду (Web/Mobile App) на користь нативного інтерфейсу месенджера. Це рішення забезпечує «безшовний» досвід (Seamless UX): користувачеві не потрібно встановлювати нове програмне забезпечення або проходити складну реєстрацію.

Взаємодія з системою реалізована у вигляді лінійного діалогу, керованого кінцевим автоматом. Сценарій роботи складається з трьох логічних етапів: налаштування профілю оцінювання, інтерактивна діагностика та візуалізація результатів.

Точкою входу в систему є стандартна команда /start. При її отриманні контролер ініціалізує нову сесію користувача в оперативній пам'яті та скидає попередні результати.

На цьому етапі критично важливим є правильне налаштування вагових коефіцієнтів, оскільки вони визначають математичну модель оцінки. Для забезпечення гнучкості система пропонує користувачеві два сценарії взаємодії (див. рис. 4.12):

– «Власна стратегія (AHP)»: призначена для досвідчених користувачів, які хочуть налаштувати систему під специфічні умови. Цей режим запускає процедуру попарних порівнянь за методом Сааті;

– «Галузевий шаблон»: реалізує концепцію «швидкого старту». Користувач може обрати один із попередньо налаштованих профілів (наприклад, «B2B SaaS», «DeepTech» або «Marketplace»), ваги яких базуються на агрегованих статистичних даних успішних стартапів.

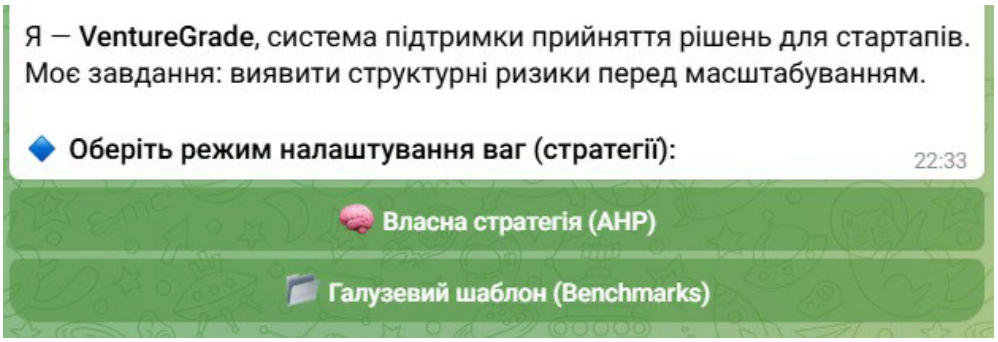


Рисунок 4.12 – Головне меню вибору стратегії оцінювання

Використання інтерактивних кнопок (InlineKeyboardMarkup) замість текстових команд дозволяє чітко структурувати шлях користувача та уникнути помилок введення на етапі навігації.

Збір даних реалізовано за патерном «Wizard» (Майстер). Система не вивалює на користувача довгу анкету, а послідовно, крок за кроком, запитує оцінку за кожним із 8 критеріїв. Це дозволяє утримати фокус уваги користувача на конкретному питанні.

Кожен екран діагностики складається з трьох елементів (див. рис. 4.13):

– заголовок критерію (наприклад, «Технологічна зрілість») із відповідним візуальним маркером (емодзі) для швидкої ідентифікації;

– контекстна підказка (Tooltip): це критично важливий елемент інтерфейсу, який пояснює суть метрики та надає "якірні значення" для шкали. Наприклад, підказка «1 = Тільки ідея, 10 = Стабільна версія» допомагає користувачеві відкалібрувати своє суб'єктивне відчуття та підвищує об'єктивність вхідних даних;

– клавіатура оцінювання: матриця кнопок від 1 до 10.

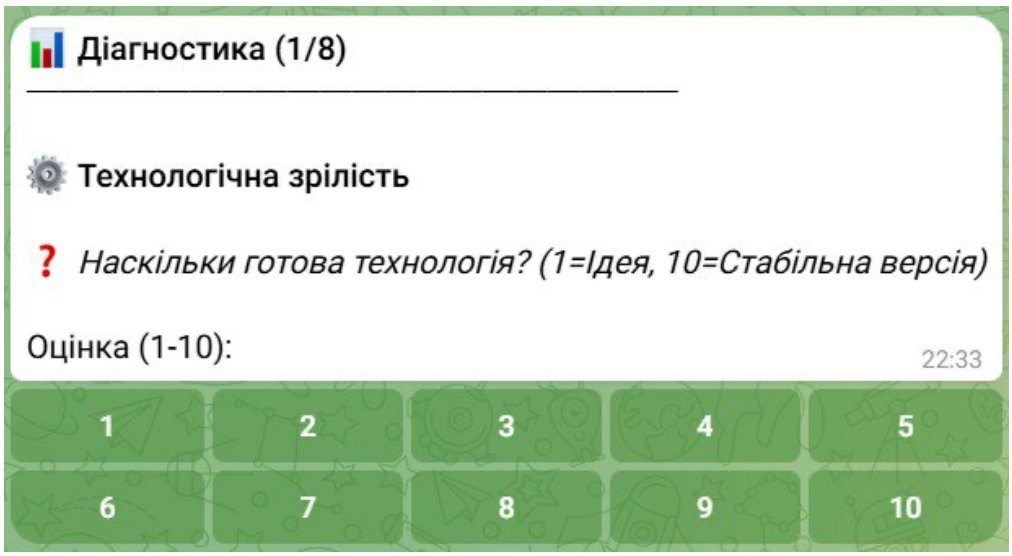


Рисунок 4.13 – Інтерфейс процесу оцінювання показників

Такий підхід має суттєву перевагу над введенням тексту: він жорстко обмежує простір можливих дій користувача (Input Validation constraints). Користувач фізично не може ввести некоректні дані (літери, спецсимволи або числа поза діапазоном), що спрощує логіку валідації на сервері та гарантує цілісність даних для математичного ядра.

Після отримання останньої оцінки система автоматично передає накопичений вектор даних у модуль розрахунків. Результат повертається користувачеві у вигляді комплексного звіту, який поєднує графічну, кількісну та текстову інформацію (див. рис. 4.14).

Структура фінального екрану спроєктована за принципом «від загального до конкретного»:

– візуалізація (Radar Chart): у верхній частині звіту розміщується згенерована бібліотекою Matplotlib радарна діаграма. Вона накладає профіль поточного проєкту (синя зона) на «ідеальний» профіль (зелений пунктир). Це дозволяє користувачеві за частку секунди візуально оцінити форму свого бізнесу та побачити диспропорції (наприклад, сильний перекіс у бік розробки при слабкому маркетингу);

– метрики та статус: нижче виводиться інтегральний індекс готовності (TOPSIS Score) у відсотках. Для кращого сприйняття числове значення дублюється візуальним прогрес-баром, сформованим із символів Unicode. Текстовий статус (наприклад, «TOO EARLY») слугує швидким вердиктом системи;

– стратегічні інсайти: найцінніша частина звіту. Блок, сформований модулем евристичного аналізу, надає конкретні, дієві рекомендації. Система не просто каже «все погано», а вказує на причину (наприклад, «Ризик Technology Push») і пропонує алгоритм дій («Зупинити розробку, почати CustDev»).

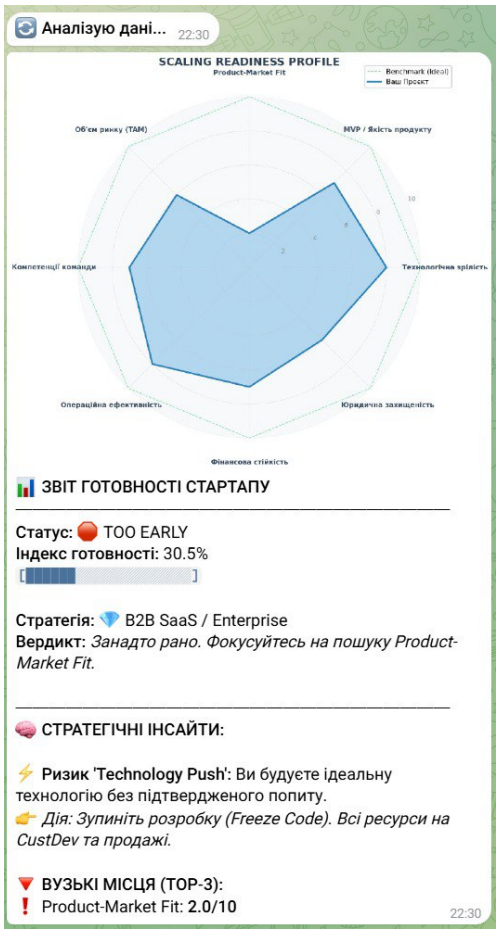


Рисунок 4.14 – Фінальний аналітичний звіт системи

Така трирівнева структура подачі інформації (Графіка -> Числа -> Текст) забезпечує високу інформативність та дозволяє користувачеві отримати комплексну експрес-діагностику проєкту всього за 2-3 хвилини, що повністю відповідає поставленій меті створення ефективної системи підтримки прийняття рішень.

## Висновки до розділу 4

У даному розділі детально описано процес програмної реалізації інформаційної системи, яка являє собою комплексне клієнт-серверне рішення для діагностики стартапів. Розробку виконано на основі мікромодульної архітектури з дотриманням принципів SOLID, що забезпечує гнучкість та масштабованість системи.

За результатами виконання розділу було досягнуто наступних результатів:

– реалізовано математичне ядро системи (модулі ahp.py, topsis.py), яке базується на векторизованих операціях бібліотеки NumPy. Це дозволило забезпечити миттєву обробку даних. Ключовою особливістю реалізації є впровадження некомпенсаторної логіки ("вето-порогів") у метод TOPSIS, що програмно унеможливлює отримання високого рейтингу проєктами з критичними недоліками;

– розроблено підсистему інтелектуальної аналітики (utils.py), яка включає базу знань експертних правил та механізм перехресного аналізу метрик (Cross-Impact Analysis). Це дозволило системі не просто розраховувати бали, а й генерувати змістовні рекомендації щодо усунення структурних ризиків (наприклад, "Technology Push" або "Premature Scaling");

– спроєктовано та реалізовано асинхронний контролер (bot.py) на базі архітектурного патерну «Кінцевий автомат» (Finite State Machine). Використання технології Long Polling та асинхронного циклу подій (Event Loop) забезпечує стабільну роботу системи під навантаженням та коректне управління контекстом користувача;

– впроваджено гібридну модель збереження даних (database.py). Використання СУБД SQLite у поєднанні з JSON-серіалізацією векторів дозволило вирішити проблему зберігання масивів змінної довжини без ускладнення реляційної схеми бази даних, що забезпечує високу гнучкість при зміні кількості критеріїв;

– реалізовано адаптивний користувацький інтерфейс на платформі Telegram. Застосування патерну «Wizard» (покроковий майстер) та захисних механізмів валідації вводу (Input Validation) дозволило створити інтуїтивно зрозумілий інструмент, що не потребує навчання користувача.

Таким чином, у четвертому розділі представлено повністю працездатний програмний продукт, який поєднує складну математичну логіку з сучасними підходами до проєктування високонавантажених систем та зручним UX.

# 5 ЕКСПЕРИМЕНТАЛЬНЕ ТЕСТУВАННЯ СИСТЕМИ

Для підтвердження адекватності розробленої математичної моделі та перевірки ефективності роботи програмного комплексу було проведено експериментальне дослідження. Метою експерименту є перевірка гіпотези про те, що впроваджена модифікація методу TOPSIS (некомпенсаторна логіка) здатна коректно ідентифікувати стартапи з високим ризиком провалу, навіть за наявності у них значних фінансових ресурсів.

## 5.1 Методика проведення експерименту

Валідація розробленої програмної системи оцінювання масштабування стартапів проводилася методом ретроспективного аналізу (Back-testing) [45]. Даний підхід є стандартом у перевірці прогностичних моделей і полягає у моделюванні процесу прийняття рішень на основі історичних даних, результат яких вже відомий на момент проведення дослідження.

Основною метою експерименту є перевірка адекватності роботи розроблених алгоритмів та їх здатності коректно ідентифікувати потенціал масштабування (або ризики провалу) стартапу, спираючись на неповні дані ранніх стадій розвитку.

Суть методу полягає у симуляції оцінки відомих стартапів станом на критичні моменти їхнього життєвого циклу – прийняття рішення про масштабування або залучення раундів інвестицій (Series A або Series B).

Важливим аспектом методики є принцип «ізоляції часу»: під час формування вхідних даних для системи використовувалася виключно та інформація, яка була публічно доступною або відомою внутрішнім стейкхолдерам на момент прийняття історичного рішення. Це дозволило уникнути помилки «упередження післязнання» (hindsight bias) та забезпечити чистоту експерименту.

Для демонстрації чутливості системи до ключових критеріїв успіху було обрано два контрастні сценарії (Case Studies), що базуються на реальних історичних прецедентах.

Сценарій №1: «Failure» (Помилкове масштабування):

– об'єкт моделювання: стартап, який на момент оцінки мав значне зовнішнє фінансування, сильну команду засновників та високу медійну активність, але згодом зазнав краху;

– ключова характеристика: невідповідність продукту ринку (відсутність Product-Market Fit) при високих показниках "марнославства" (vanity metrics);

– мета сценарію: перевірити, чи зможе система виявити фундаментальні проблеми (наприклад, негативну юніт-економіку або високий відтік клієнтів) за фасадом загального успіху та рекомендувати утриматися від масштабування.

Сценарій №2: «Success» (Успішне масштабування):

– об'єкт моделювання: стартап, який демонстрував стійке органічне зростання, високе утримання користувачів та ефективну бізнес-модель, що згодом дозволило йому стати "єдинорогом" або успішно вийти на IPO;

– ключова характеристика: збалансованість команди, технологічна перевага та підтверджений попит;

– мета сценарію: підтвердити здатність системи ідентифікувати прихований потенціал ("алмаз") та надати високу інтегральну оцінку пріоритетності масштабування.

Оскільки розроблена система оперує кількісними оцінками експертів, було проведено процедуру трансформації якісних історичних даних у кількісні показники.

Вхідні дані (вектори експертних оцінок за 10-бальною шкалою) для кожного критерію та підкритерію ієрархічної моделі були сформовані на основі аналізу таких джерел:

– публічні фінансові звіти компаній за відповідні періоди (S-1 filings, річні звіти) ;

– аналітичні статті та інтерв'ю засновників, датовані періодом до настання успіху чи краху;

– «Post-mortem» аналізи (детальні розбори причин провалу стартапів), з яких було виокремлено стан метрик на момент «точки неповернення».

Кожен історичний факт трансформувався у бал (від 1 до 10) згідно з розробленою шкалою лінгвістичних змінних, що імітувало роботу незалежної групи експертів.

### 5.2 Аналіз кейсу №1: Quibi (Сценарій провалу)

Як еталонний приклад негативного сценарію («Failure») для тестування системи було обрано американський стрімінговий сервіс Quibi [39, 50], запуск якого відбувся у квітні 2020 року. Цей кейс є хрестоматійним прикладом передчасного масштабування (Premature Scaling), коли наявність значних ресурсів не компенсує відсутність відповідності продукту ринку.

На момент запуску проєкт залучив рекордні для індустрії $1.75 млрд інвестицій до початку надання послуг. Команду очолювали Джеффрі Катценберг (екс-голова Disney Studios) та Мег Вітмен (екс-CEO HP), що створювало ілюзію неминучого успіху. Суть продукту полягала у створенні високобюджетного короткометражного відеоконтенту виключно для мобільних пристроїв. Однак, попри потужну маркетингову кампанію, сервіс закрився через 6 місяців після запуску.

На основі ретроспективних даних для системи було сформовано вхідний вектор оцінок. Експертні оцінки виставлялися, симулюючи ситуацію за 1 місяць до офіційного запуску (березень 2020), коли рішення про агресивне масштабування вже було прийнято.

Таблиця 5.1 – Вхідні значення критеріїв для кейсу Quibi

| **Група критеріїв** | **Оцінка (1-10)** | **Обґрунтування оцінки** |
|---|---|---|
| K1: фінансова стійкість | 10.0 | Аномально високий бюджет, повне покриття операційних витрат на 2 роки вперед. |
| K2: компетенції команди | 10.0 | Засновники з досвідом управління глобальними корпораціями (Disney, HP). |
| K3: технологічна зрілість | 8.0 | Якісна реалізація мобільного додатку, унікальна технологія «Turnstyle». |
| K4: ринкові умови | 5.0 | Висока конкуренція з безкоштовними аналогами (YouTube, TikTok), початок пандемії. |
| K5: Product-Market Fit | 2.0 | Гіпотези не валідовані на реальних користувачах, продукт не вирішує нагальної проблеми («Pain Point»). |

Метою експерименту було безпосереднє порівняння результатів оцінювання, отриманих за допомогою класичного методу TOPSIS, та результатів, згенерованих розробленою модифікацією із застосуванням механізму нелінійної штрафної фільтрації.

При використанні стандартного алгоритму, де інтегральна оцінка формується як середнє зважене значення, відбувся ефект математичної компенсації. Критично низький показник відповідності продукту ринку був перекритий максимальними балами за критеріями фінансової стійкості та компетентності команди. В результаті класичний метод показав високий рейтинг готовності стартапу (близько 85%), що інтерпретується як рекомендація до інвестування. Такий висновок є хибно-позитивним (False Positive), оскільки в реальності ігнорування проблем із продуктом призвело до значних фінансових втрат.

На відміну від класичного підходу, розроблена система під час обробки вхідних даних ідентифікувала критичне відхилення у блоці критеріїв, що відповідають за валідацію продукту. Алгоритм спрацював за наступною логікою:

– сканування обмежень: система зафіксувала, що показник Product-Market Fit знаходиться на рівні, нижчому за мінімально допустимий поріг життєздатності продукту;

– активація захисного механізму: виявлення «слабкої ланки» призвело до блокування стандартної процедури лінійної згортки. Замість цього було активовано механізм «вето», який наклав динамічний штраф на загальний рейтинг;

– коригування рейтингу: базовий високий рейтинг, сформований завдяки сильнім фінансам та команді, був примусово знижений шляхом застосування коефіцієнта жорсткості штрафу. Це дозволило скоригувати фінальну оцінку з потенційних 90% до реалістичних 54%.

На основі розрахованого індексу готовності система автоматично сформувала звіт із вердиктом «Потребує оптимізації» (Needs Optimization). Модуль аналітики діагностував патерн ризику «Ресурсна пастка».

Змістовна інтерпретація діагнозу вказує на небезпечний дисбаланс: наявність значного бюджету при відсутності підтвердженого попиту створює ризик неефективного використання капіталу. Система сформувала заборону на масштабування маркетингових активностей до моменту, поки показники утримання користувачів не досягнуть нормативних значень.

Експеримент підтвердив, що розроблена методика дозволяє уникнути помилки виживаного та блокує рекомендації до масштабування для проєктів, які мають гарні ресурсні показники, але фундаментальні проблеми з ціннісною пропозицією.

### 5.3 Аналіз кейсу №2: Grammarly (Сценарій успіху)

Як приклад позитивного сценарію («Success») для перевірки системи було обрано український стартап Grammarly [41]. Для моделювання було взято історичний період 2017 року – безпосередньо перед залученням першого великого раунду зовнішніх інвестицій (General Catalyst, $110 млн).

На відміну від попереднього кейсу, Grammarly на цьому етапі розвивалася за моделлю органічного зростання (bootstrapping). Компанія вже була прибутковою, мала чітку бізнес-модель та мільйони активних щоденних користувачів. Цей випадок є ідеальним для тестування здатності системи розпізнавати стартапи, які

готові до масштабування не через наявність надлишкового бюджету, а завдяки зрілості продукту.

Вхідний вектор оцінок відображає збалансований стан компанії, де ключовим драйвером є продукт, а не зовнішній капітал. Оцінка критеріїв:

– фінансова стійкість (7.0 балів): оцінка нижча за максимальну, оскільки компанія оперувала власними коштами, а не гігантськими інвестиційними траншами. Проте наявність стабільного позитивного грошового потоку (Cash Flow) забезпечила високий бал надійності;

– компетенції команди (8.0 балів): сильна технічна команда та ефективний менеджмент, орієнтований на дані, хоча й без гучних імен "зіркових" CEO на той момент;

– Product-Market Fit (9.0 балів): найвищий показник у векторі. Продукт вирішував чітку проблему користувачів, мав вірусний ефект розповсюдження та високі показники утримання клієнтів.

Процес обробки даних системою продемонстрував принципову відмінність від кейсу з проблемним стартапом:

– перевірка порогових значень: на етапі первинного сканування система перевірила всі критичні метрики (PMF, Unit-економіка, Технічна стабільність). Оскільки показник відповідності продукту ринку склав 9.0 балів (що значно перевищує критичний поріг у 3.0 бали), механізм блокування не було активовано;

– вибір алгоритму згортки: оскільки жоден із "червоних прапорців" не спрацював, система перейшла до стандартного алгоритму розрахунку. Було застосовано класичний метод TOPSIS, який розраховує близькість об'єкта до ідеального рішення на основі середньозважених відстаней;

– відсутність штрафів: штрафні коефіцієнти дорівнювали одиниці (тобто не змінювали результат), оскільки розвиток проєкту був гармонійним і не мав перекосів у бік "ресурсної бульбашки".

За результатами розрахунку інтегральний індекс готовності до масштабування склав 82.0%. Це високий показник, який базується на реальних досягненнях продукту, а не лише на ресурсах.

На основі отриманого значення система згенерувала фінальний звіт:

– статус: READY TO SCALE (Готовий до масштабування) ;

– діагноз модуля аналітики: «Структурних дисбалансів не виявлено. Розвиток проєкту гармонійний. Високий рівень PMF є надійним фундаментом для ефективного освоєння інвестицій. Ризик неефективного витрачання коштів мінімальний».

Експеримент підтвердив коректність роботи системи у позитивних сценаріях. Вона не занижує оцінки стартапам, які розвиваються органічно, і правильно ідентифікує момент, коли залучення великих інвестицій стане каталізатором росту, а не способом покриття збитків.

### 5.4 Висновки з експерименту

Узагальнені результати порівняльного аналізу роботи розробленої системи на двох полярних сценаріях наведено в Таблиці 5.1. Дані демонструють реакцію алгоритму на різні комбінації вхідних параметрів та ефективність застосування штрафних механізмів.

Таблиця 5.2 – Результати ретроспективної валідації системи

| **Параметр** | **Кейс 1: Quibi (Failure)** | **Кейс 2: Grammarly (Success)** |
|---|---|---|
| Вхідні дані (шкала 1-10) | Фінанси: 10, Команда: 10, PMF: 2 | Фінанси: 7, Команда: 8, PMF: 9 |
| Дія системи | Активація штрафу (Вето) | Розрахунок без штрафів |
| TOPSIS Score (Результат) | 54% | 82% |

Кінець таблиці 5.2

| **Параметр** | **Кейс 1: Quibi (Failure)** | **Кейс 2: Grammarly (Success)** |
|---|---|---|
| Статус системи | NEEDS OPTIMIZATION | READY TO SCALE |
| Висновок | Ризик ідентифіковано вірно | Прогноз підтверджено |

Проведений експеримент підтвердив, що розроблена математична модель та програмна реалізація методу TOPSIS ефективно вирішують фундаментальну проблему класичних методів багатокритеріального аналізу — проблему повної компенсаторності.

В ході експерименту було встановлено наступне:

– блокування помилкових позитивних рішень. У кейсі Quibi система продемонструвала здатність ігнорувати "шум" від надвисоких фінансових показників. Класичний підхід у такій ситуації видав би рекомендацію до інвестування, орієнтуючись на середнє арифметичне. Натомість, розроблений алгоритм коректно ідентифікував критичний ризик відсутності ринкового попиту (PMF) і примусово знизив рейтинг проєкту до рівня "ризиковий". Це підтверджує, що система виконує функцію запобіжника від передчасного масштабування;

– адекватність оцінки органічного зростання. У кейсі Grammarly система показала, що вона не дискримінує проєкти з помірним фінансуванням, якщо вони мають сильний продукт. Відсутність штрафних санкцій дозволила стартапу отримати високий рейтинг виключно за рахунок збалансованості показників та зрілості технології;

– практична цінність. Результати валідації свідчать про те, що запропонована система підтримки прийняття рішень (СППР) здатна суттєво знизити інвестиційні ризики. Вона діє як автоматизований аудитор, який фокусує увагу стейкхолдерів на «вузьких місцях» проєкту, які неможливо перекрити лише вливанням коштів.

Таким чином, розроблена методика довела свою придатність для використання в реальних умовах оцінювання стартапів ранніх стадій, забезпечуючи більш точні та безпечні рекомендації порівняно з традиційними лінійними методами.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було реалізовано інформаційну систему оцінки готовності стартапу до масштабування, яка поєднує методи багатокритеріального аналізу (AHP, TOPSIS), алгоритми евристичної генерації рекомендацій та технології інтерактивної взаємодії через чат-бот. Основною метою роботи було створення доступного та об’єктивного інструменту для діагностики зрілості бізнес-проєктів, що дозволяє засновникам та інвесторам знизити ризики передчасного витрачання ресурсів та підвищити обґрунтованість управлінських рішень.

У ході дослідження проаналізовано сучасний стан проблеми масштабування інноваційних компаній, визначено життєвий цикл стартапу та ключові фактори успіху. Це дозволило сформулювати вимоги до майбутньої системи та визначити оптимальний математичний апарат для вирішення поставленої задачі. Зокрема, було обґрунтовано вибір гібридної моделі, що поєднує метод аналізу ієрархій (AHP) для визначення вагових коефіцієнтів стратегічних пріоритетів та метод TOPSIS для розрахунку інтегрального індексу готовності на основі близькості до еталонного рішення.

Розроблено науково обґрунтовану математичну модель. Створено гібридну модель багатокритеріального оцінювання, що поєднує метод аналізу ієрархій (AHP) для експертного зважування критеріїв та метод TOPSIS для ранжування альтернатив. Наукова новизна роботи полягає у вдосконаленні методу TOPSIS шляхом введення штрафних коефіцієнтів (механізму некомпенсаторної логіки). Цей механізм дозволяє моделі блокувати високу загальну оцінку стартапу, якщо виявлено критичні провали в одному або кількох ключових критеріях, що значно підвищує її діагностичну точність.

У процесі розробки створено модульну архітектуру системи, що складається з математичного ядра (модулі ahp.py, topsis.py), підсистеми аналітики та візуалізації (utils.py), контролера взаємодії з Telegram API (bot.py) та шару

збереження даних (database.py). Це забезпечує гнучкість, масштабованість і простоту використання системи як на мобільних, так і на десктопних пристроях. Реалізацію виконано мовою Python з використанням бібліотек NumPy (для матричних обчислень), Matplotlib (для генерації радарних діаграм) та SQLite (для забезпечення персистентності даних).

Важливим елементом проєкту стало впровадження модулю стратегічного аналізу. Розроблена СППР надає не лише кількісний «Індекс готовності», але й містить модуль евристичного аналізу, який автоматично генерує стратегічні інсайти та рекомендації, спрямовані на усунення виявлених «вузьких місць».

Підтверджено практичну працездатність. Проведена ретроспективна валідація системи на реальних прикладах – успішному (Grammarly) та неуспішному (Quibi) стартапах – підтвердила, що модифікований алгоритм коректно ідентифікує високі ризики передчасного масштабування. Це доводить надійність системи як інструменту прогнозування ризиків.

Отже, результати дослідження підтвердили досягнення поставленої мети та виконання всіх завдань роботи. Розроблена інформаційна система може бути використана стартапами, бізнесами-початківцями та індивідуальними підприємцями як ефективний засіб експрес-діагностики перед залученням інвестицій. У подальшому доцільно розширити її функціонал шляхом інтеграції з API фінансових сервісів для автоматичного збору метрик, розширення бази знань експертних профілів та впровадження елементів машинного навчання для прогнозування успішності проєктів.

Таким чином, виконана робота робить суттєвий внесок у розвиток систем підтримки прийняття рішень в інноваційному менеджменті, поєднуючи суворі математичні методи з доступними програмними засобами реалізації.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Startup Genome. The Global Startup Ecosystem Report 2024. San Francisco : Startup Genome LLC, 2024. 350 p. URL: https://startupgenome.com/report/gser2024 (дата звернення: 20.10.2025).

2. CB Insights. The Top 12 Reasons Startups Fail. CB Insights Research. 2024. URL: https://www.cbinsights.com/research/startup-failure-reasons-top/ (дата звернення: 21.10.2025).

3. Marmer M., Herrmann B. L., Dogrultan E., Berman R. Startup Genome Report Extra on Premature Scaling: A deep dive into why most high growth startups fail. Startup Genome. 2011. Vol. 1, No. 1. P. 1–56.

4. Бланк С., Дорф Б. Стартап: настільна книга засновника. Київ : Наш Формат, 2018. 632 с.

5. Ries E. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York : Crown Currency, 2011. 336 p.

6. Osterwalder A., Pigneur Y. Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers. Hoboken : John Wiley & Sons, 2010. 288 p.

7. Maurya A. Running Lean: Iterate from Plan A to a Plan That Works. Sebastopol : O'Reilly Media, 2012. 240 p.

8. Thiel P., Masters B. Zero to One: Notes on Startups, or How to Build the Future. New York : Crown Business, 2014. 224 p.

9. Horowitz B. The Hard Thing About Hard Things: Building a Business When There Are No Easy Answers. New York : HarperBusiness, 2014. 304 p.

10. Christensen C. M. The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Boston : Harvard Business Review Press, 2016. 288 p.

11. Saaty T. L. The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. New York : McGraw-Hill, 1980. 287 p.

12. Saaty T. L. Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World. Pittsburgh : RWS Publications, 2008. 323 p.

13. Hwang C. L., Yoon K. Multiple Attribute Decision Making: Methods and Applications. Berlin : Springer-Verlag, 1981. 259 p.

14. Zadeh L. A. Fuzzy sets. Information and Control. 1965. Vol. 8, No. 3. P. 338–353.

15. Behzadian M., Otaghsara S. K., Yazdani M., Ignatius J. A state-of-the-art survey of TOPSIS applications. Expert Systems with Applications. 2012. Vol. 39, No. 17. P. 13051–13069.

16. Onder E., Sundus D. The combination of AHP and TOPSIS methods for project selection. Journal of Industrial Engineering and Operations Management. 2013. Vol. 1, No. 1. P. 14–25.

17. Kovryga O., Kamiński P. Comparative Analysis of Multi-Criteria Decision-Making Methods for Startup Evaluation. International Journal of Innovation and Technology Management. 2023. Vol. 20, No. 02. P. 45–62.

18. Гнатієнко Г. М., Снітюк В. Є. Експертні технології прийняття рішень : монографія. Київ : ТОВ «Маклаут», 2008. 444 с.

19. Петров Е. Г., Новожилова М. В., Гребеннік І. В. Методи і засоби прийняття рішень у соціально-економічних системах : навч. посіб. Київ : Техніка, 2004. 256 с.

20. Velasquez M., Hester P. T. An Analysis of Multi-Criteria Decision Making Methods. International Journal of Operations Research. 2013. Vol. 10, No. 2. P. 56–66.

21. Kahraman C., Cebeci U., Ulukan Z. Multi-criteria supplier selection using fuzzy AHP. Logistics Information Management. 2003. Vol. 16, No. 6. P. 382–394.

22. Chen S. J., Hwang C. L. Fuzzy Multiple Attribute Decision Making: Methods and Applications. Berlin : Springer-Verlag, 1992. 536 p.

23. Mankins J. C. Technology Readiness Levels: A White Paper. NASA, Office of Space Access and Technology. 1995. URL: https://www.nasa.gov/pdf/458490main_TRL_Definitions.pdf.

24. ISO 16290:2013. Space systems — Definition of the Technology Readiness Levels (TRLs) and their criteria of assessment. Geneva : ISO, 2013. 18 p.

25. Cloudwatch HUB. Market Readiness Levels (MRL). URL: https://www.cloud-watchhub.eu/exploitation/market-readiness-levels-mrl (дата звернення: 22.10.2025).

26. Python Software Foundation. Python 3.12.7 Documentation. URL: https://docs.python.org/3/ (дата звернення: 25.10.2025).

27. Telegram API. Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api (дата звернення: 26.10.2025).

28. Python-telegram-bot. Documentation for python-telegram-bot v21.x. URL: https://docs.python-telegram-bot.org/en/v21.6/ (дата звернення: 26.10.2025).

29. NumPy Developers. NumPy User Guide. Release 1.26.0. URL: https://numpy.org/doc/stable/user/index.html (дата звернення: 27.10.2025).

30. Matplotlib Developers. Matplotlib: Visualization with Python. URL: https://matplotlib.org/stable/users/index.html (дата звернення: 27.10.2025).

31. Owens M., Allen G. The Definitive Guide to SQLite. Berkeley : Apress, 2010. 368 p.

32. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

33. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

34. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816 p.

35. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 3rd ed. Sebastopol : O'Reilly Media, 2022. 550 p.

36. Richardson L., Amundsen M. RESTful Web APIs. Sebastopol : O'Reilly Media, 2013. 404 p.

37. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Boston : Addison-Wesley, 2012. 640 p. (SEI Series in Software Engineering).

38. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.

39. Auletta K. Frenemies: The Epic Disruption of the Ad Business (and Everything Else). New York : Penguin Press, 2018. 384 p. (Джерело про медіа-бізнес, релевантне для Quibi).

40. Swisher K. Burn Book: A Tech Love Story. New York : Simon & Schuster, 2024. (Згадки про сучасні техно-провали).

41. Grammarly. Grammarly Engineering Blog. URL: https://www.grammarly.com/blog/engineering/ (дата звернення: 28.10.2025).

42. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 26 с.

43. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмного забезпечення. Вимоги до якості систем і програмного забезпечення та оцінювання (SQuaRE). Моделі якості систем і програмного забезпечення. Київ : ДП «УкрНДНЦ», 2016.

44. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 17 с.

45. ISO/IEC/IEEE 29119-1:2013. Software and systems engineering — Software testing — Part 1: Concepts and definitions. Geneva : ISO, 2013.

46. Yadav V. et al. A Hybrid AHP-TOPSIS Approach for Supplier Selection in Manufacturing Industry. International Journal of Engineering Research & Technology. 2019. Vol. 8, No. 10.

47. Taha R. A., Daim T. Multi-Criteria Applications in Renewable Energy Analysis, a Literature Review. Research Evaluation and Selection. 2013. P. 17–30.

48. Broman G., Robert K. H. A framework for strategic sustainable development. Journal of Cleaner Production. 2017. Vol. 140. P. 17–31.

49. Blank S. Why the Lean Start-Up Changes Everything. Harvard Business Review. 2013. Vol. 91, No. 5. P. 63–72.

50. Eisenmann T. R. Why Startups Fail. New York : Currency, 2021. 368 p.

# ДОДАТОК А

## Код файлу ahp.py

```
import numpy as np
from typing import List, Dict

class AHPCalculator:
    RANDOM_INDEX = {
        1: 0.00, 2: 0.00, 3: 0.58, 4: 0.90, 5: 1.12,
        6: 1.24, 7: 1.32, 8: 1.41, 9: 1.45, 10: 1.49,
        11: 1.51, 12: 1.48, 13: 1.56, 14: 1.57, 15: 1.59
    }

    def __init__(self, criteria: List[str]):
        self.criteria = criteria
        self.n = len(criteria)
        if self.n > 15:
            raise ValueError("Кількість критеріїв перевищує межі точності
методу AHP (max 15).")
        self.comparison_matrix = np.ones((self.n, self.n))

    def set_comparison(self, i: int, j: int, value: float):
        self.comparison_matrix[i, j] = value
        self.comparison_matrix[j, i] = 1 / value

    def calculate_weights(self) -> np.ndarray:
        eigenvalues, eigenvectors = np.linalg.eig(self.comparison_matrix)
        max_idx = np.argmax(eigenvalues.real)
        self.lambda_max = eigenvalues[max_idx].real

        principal_eigenvector = eigenvectors[:, max_idx]
        principal_eigenvector = np.abs(np.real(principal_eigenvector))
        self.weights = principal_eigenvector / principal_eigenvector.sum()

        return self.weights

    def calculate_consistency_ratio(self) -> float:
        if self.n <= 2: return 0.0

        ci = (self.lambda_max - self.n) / (self.n - 1)
        ri = self.RANDOM_INDEX.get(self.n, 1.59)

        return max(0.0, ci / ri) if ri != 0 else 0.0

    def reset(self):
        self.comparison_matrix = np.ones((self.n, self.n))

class ExpertProfiles:
    @staticmethod
    def get_profiles() -> Dict[str, Dict]:
```

```
        return {
            "b2b_saas": {
                "name": "💎 B2B SaaS / Enterprise",
                "desc": "Стратегія: Акцент на продажах (Sales) та
утриманні (Re-tention).",
                "weights": [0.30, 0.40, 0.20, 0.10]
            },
            "marketplace": {
                "name": "🌐 Marketplace / Platform",
                "desc": "Стратегія: Ефект мережі та масштабування опе-
рацій.",
                "weights": [0.15, 0.45, 0.25, 0.15]
            },
            "deep_tech": {
                "name": "🚀 DeepTech / R&D",
                "desc": "Стратегія: Унікальна технологія та захист IP.",
                "weights": [0.50, 0.15, 0.25, 0.10]
            },
            "service": {
                "name": "🤝 Service Business",
                "desc": "Стратегія: Репутація бренду та команда.",
                "weights": [0.10, 0.30, 0.50, 0.10]
            }
        }

def create_ahp_for_criteria():
    criteria = ["Продукт & Технології", "Ринок & Клієнти", "Команда & Опе-
рації", "Фінанси & Право"]

    descriptions = [
        "Якість рішення, інноваційність, стадія готовності (MVP/Scale).",
        "Обсяг ринку (TAM), вартість залучення (CAC), попит.",
        "Досвід фаундерів, ефективність бізнес-процесів.",
        "Прибутковість (P&L), грошовий потік, юридичний захист."
    ]

    pairs = [(i, j) for i in range(len(criteria)) for j in range(i + 1,
len(criteria))]
    return criteria, pairs, descriptions
```

# ДОДАТОК В

## Код файлу bot.py

```
import logging
import os
import numpy as np
from dotenv import load_dotenv

from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import (
    Application, CommandHandler, CallbackQueryHandler,
    ConversationHandler, ContextTypes
)

from ahp import AHPCalculator, create_ahp_for_criteria, ExpertProfiles
from topsis import TOPSISCalculator, create_topsis_subcriteria
from utils import ReportGenerator, create_saaty_scale_keyboard, cre-
ate_rating_keyboard
from database import Database

load_dotenv()
TOKEN = os.getenv("TELEGRAM_TOKEN")

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s', level=logging.INFO)
logger = logging.getLogger(__name__)

CHOOSE_MODE, AHP_COMPARING, AHP_CONSISTENCY, TOPSIS_RATING, SHOW_RESULTS =
range(5)

db = Database()
user_storage = {}

criteria_names, ahp_pairs, criteria_descriptions = create_ahp_for_crite-
ria()
subcriteria, sub_groups, emojis, sub_help_texts = create_topsis_subcrite-
ria()

class StartupBot:

    async def start(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        user = update.effective_user
        user_storage[user.id] = {
            'ahp': AHPCalculator(criteria_names),
            'pair_idx': 0,
            'ahp_attempts': 0,
            'topsis_scores': [],
```

```
            'sub_idx': 0,
            'mode': 'manual',
            'mode_name': 'Custom Strategy',
            'weights': None
        }

        txt = (
            f"👋 <b>Вітаю, {user.first_name}!</b>\n\n"
            "Я — <b>VentureGrade AI</b>, система підтримки прийняття
рішень для стартапів.\n"
            "Моє завдання: виявити структурні ризики перед масштабуван-
ням.\n\n"
            "🔷 <b>Оберіть режим налаштування ваг (стратегії):</b>"
        )
        kb = [
            [InlineKeyboardButton("🧠 Власна стратегія (AHP)",
callback_data="mode_manual")],
            [InlineKeyboardButton("📂 Галузевий шаблон (Benchmarks)",
callback_data="mode_template")]
        ]
        await update.message.reply_text(txt, re-ply_markup=InlineKeyboard-
Markup(kb), parse_mode='HTML')
        return CHOOSE_MODE

    async def handle_mode(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        query = update.callback_query
        await query.answer()

        if query.data == "mode_manual":
            await query.edit_message_text(
                "📐 <b>Етап 1: AHP (Analytic Hierarchy Process)</b>\n\n"
                "Визначимо стратегічні пріоритети. Я буду показувати пари
кри-теріїв, "
                "а ви обирайте, що є більш критичним для успіху.",
                reply_markup=InlineKeyboardMarkup([[InlineKeyboardBut-
ton("🚀 Почати", callback_data="next_pair")]]),
                parse_mode='HTML'
            )
            return AHP_COMPARING
        else:
            kb = []
            for k, v in ExpertProfiles.get_profiles().items():
                kb.append([InlineKeyboardButton(v['name'],
callback_data=f"profile_{k}")])
            await query.edit_message_text(
                "📂 <b>Оберіть бізнес-модель:</b>\n"
                "Система використає ваги, що базуються на статистиці
успішних ком-паній.",
                reply_markup=InlineKeyboardMarkup(kb), parse_mode='HTML'
            )
```

```
            return CHOOSE_MODE

    async def process_profile(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        query = update.callback_query
        await query.answer()
        pkey = query.data.split("_", 1)[1]
        profile = ExpertProfiles.get_profiles()[pkey]
        uid = update.effective_user.id

        user_storage[uid]['weights'] = np.array(profile['weights'])
        user_storage[uid]['mode'] = f"template_{pkey}"
        user_storage[uid]['mode_name'] = profile['name']

        await query.edit_message_text(
            f"✅ <b>Профіль: {profile['name']}</b>\n"
            f"<i>{profile['desc']}</i>\n\n"
            "Переходимо до діагностики метрик.",
            reply_markup=InlineKeyboardMarkup(
                [[InlineKeyboardButton("📊 Почати оцінку",
callback_data="start_topsis")]]),
            parse_mode='HTML'
        )
        return TOPSIS_RATING

    async def show_pair(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        query = update.callback_query
        uid = update.effective_user.id
        data = user_storage[uid]

        if data['pair_idx'] >= len(ahp_pairs):
            return await self.check_consistency(update, context)

        i, j = ahp_pairs[data['pair_idx']]
        kb = []
        for row in create_saaty_scale_keyboard():
            kb.append([InlineKeyboardButton(t,
callback_data=f"cmp_{i}_{j}_{d}") for t, d in row])

        msg = (
            f"⚖️ <b>Пріоритети ({data['pair_idx'] +
1}/{len(ahp_pairs)})</b>\n"
            f"━━━━━━━━━━━━━━━━━━━━\n\n"
            f"🔵 <b>A: {criteria_names[i]}</b>\n"
            f"<i>└ {criteria_descriptions[i]}</i>\n\n"
            f"VS\n\n"
            f"🟠 <b>Б: {criteria_names[j]}</b>\n"
            f"<i>└ {criteria_descriptions[j]}</i>"
        )
```

```
        await query.edit_message_text(msg, reply_markup=InlineKeyboard-
Markup(kb), parse_mode='HTML')
        return AHP_COMPARING

    async def process_ahp(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        query = update.callback_query
        data = user_storage[update.effective_user.id]
        parts = query.data.split("_")
        val = int(parts[3])
        res = val if len(parts) > 4 and parts[4] == "left" else 1 / val if
len(parts) > 4 else 1.0
        data['ahp'].set_comparison(int(parts[1]), int(parts[2]), res)
        data['pair_idx'] += 1
        return await self.show_pair(update, context)

    async def check_consistency(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        uid = update.effective_user.id
        data = user_storage[uid]
        weights = data['ahp'].calculate_weights()
        cr = data['ahp'].calculate_consistency_ratio()
        data['weights'] = weights

        if cr > 0.1 and data['ahp_attempts'] < 1:
            data['ahp_attempts'] += 1
            data['pair_idx'] = 0
            data['ahp'].reset()
            await update.callback_query.edit_message_text(
                f"⚠️ <b>Протиріччя (CR={cr:.2f})</b>\n\n"
                "Ваші відповіді нелогічні (A > B, B > C, але C > A).
Спробуйте ще раз.",
                reply_markup=InlineKeyboardMarkup([[InlineKeyboardBut-
ton("🔄 Повторити", callback_data="next_pair")]]),
                parse_mode='HTML'
            )
            return AHP_COMPARING

        await update.callback_query.edit_message_text(
            f"✅ <b>Стратегію збережено (CR={cr:.2f})</b>\n"
            "Переходимо до оцінки.",
            reply_markup=InlineKeyboardMarkup([[InlineKeyboardButton("📊 До
метрик", callback_data="start_topsis")]]),
            parse_mode='HTML'
        )
        return TOPSIS_RATING

    async def show_rating(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        query = update.callback_query
        data = user_storage[update.effective_user.id]
```

```
        if data['sub_idx'] >= len(subcriteria):
            return await self.calculate_final(update, context)

        kb = []
        for row in create_rating_keyboard():
            kb.append([InlineKeyboardButton(t, callback_data=f"rate_{d}")
for t, d in row])

        idx = data['sub_idx']
        msg = (
            f"📊 <b>Діагностика ({idx + 1}/{len(subcriteria)})</b>\n"
            f"━━━━━━━━━━━━━━━━━━━━\n\n"
            f"{emojis[idx]} <b>{subcriteria[idx]}</b>\n\n"
            f"❓ <i>{sub_help_texts[idx]}</i>\n\n"
            f"Оцінка (1–10):"
        )
        await query.edit_message_text(msg, reply_markup=InlineKeyboard-
Markup(kb), parse_mode='HTML')
        return TOPSIS_RATING

    async def process_rating(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        val = int(update.callback_query.data.split("_")[1])
        user_storage[update.effective_user.id]['topsis_scores'].ap-
pend(val)
        user_storage[update.effective_user.id]['sub_idx'] += 1
        return await self.show_rating(update, context)

    async def calculate_final(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        uid = update.effective_user.id
        data = user_storage[uid]
        await update.callback_query.edit_message_text("🔄 <b>Аналізую
дані...</b>", parse_mode='HTML')

        user_scores = np.array(data['topsis_scores'])
        weights = data['weights']

        # --- DEBUG INFO ---
        print(f"DEBUG: User Scores: {user_scores}")
        print(f"DEBUG: Weights: {weights}")

        topsis = TOPSISCalculator(subcriteria, sub_groups)
        score, _, _ = topsis.calculate_topsis_score(user_scores, weights)
        weaknesses = topsis.analyze_weaknesses(user_scores)

        print(f"DEBUG: Final Score: {score}")

        db.save_assessment(uid, update.effective_user.first_name,
data['mode'], weights, user_scores, score, weaknesses)
```

```
        report = ReportGenerator.format_readiness_message(
            score, weaknesses, weights, criteria_names, user_scores, da-
ta['mode_name']
        )
        img = ReportGenerator.generate_radar_chart(subcriteria,
user_scores)

        await context.bot.send_photo(chat_id=update.effective_chat.id,
photo=img, caption=report, parse_mode='HTML')
        await context.bot.send_message(chat_id=update.effective_chat.id,
text="🏁 /start для нової оцінки",
                                       reply_markup=None)
        return ConversationHandler.END

    async def cancel(self, update: Update, context: Con-
textTypes.DEFAULT_TYPE) -> int:
        await update.message.reply_text("❌ Скасовано.")
        return ConversationHandler.END

def main():
    if not TOKEN:
        print("ERROR: TELEGRAM_TOKEN not found!")
        return
    bot = StartupBot()
    app = Application.builder().token(TOKEN).build()
    conv = ConversationHandler(
        entry_points=[CommandHandler("start", bot.start)],
        states={
            CHOOSE_MODE: [CallbackQueryHandler(bot.handle_mode, pat-
tern="^mode_"),
                          CallbackQueryHandler(bot.process_profile, pat-
tern="^profile_")],
            AHP_COMPARING: [CallbackQueryHandler(bot.process_ahp, pat-
tern="^cmp_"),
                            CallbackQueryHandler(bot.show_pair, pat-
tern="^next_pair$")],
            AHP_CONSISTENCY: [CallbackQueryHandler(bot.show_pair, pat-
tern="^next_pair$")],
            TOPSIS_RATING: [CallbackQueryHandler(bot.show_rating, pat-
tern="^start_topsis$"),
                            CallbackQueryHandler(bot.process_rating, pat-
tern="^rate_")]
        },
        fallbacks=[CommandHandler("cancel", bot.cancel)]
    )
    app.add_handler(conv)
    print("Bot is running...")
    app.run_polling()

if __name__ == "__main__":
    main()
```

# ДОДАТОК С

## Код файлу database.py

```
import sqlite3
import json
import numpy as np
from datetime import datetime
import logging

logger = logging.getLogger(__name__)

class Database:
    def __init__(self, db_name="startup_assess.db"):
        self.db_name = db_name
        self.create_tables()

    def create_tables(self):
        try:
            with sqlite3.connect(self.db_name) as conn:
                conn.execute("""
                CREATE TABLE IF NOT EXISTS assessments (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    user_id INTEGER,
                    user_name TEXT,
                    date TIMESTAMP,
                    mode TEXT,
                    ahp_weights TEXT,
                    topsis_scores TEXT,
                    final_score REAL,
                    weaknesses TEXT
                )
                """)
                conn.commit()
        except Exception as e:
            logger.error(f"DB Init Error: {e}")

    def save_assessment(self, user_id, user_name, mode, ahp_weights, top-
sis_scores, final_score, weaknesses):
        try:
            weights_json = json.dumps(ahp_weights.tolist() if isin-
stance(ahp_weights, np.ndarray) else ahp_weights)
            scores_json = json.dumps(topsis_scores.tolist() if isin-
stance(topsis_scores, np.ndarray) else topsis_scores)
            weaknesses_json = json.dumps(weaknesses)

            with sqlite3.connect(self.db_name) as conn:
                query = """
                INSERT INTO assessments
                (user_id, user_name, date, mode, ahp_weights, top-
sis_scores, fi-nal_score, weaknesses)
```

```
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)
                """
                conn.execute(query, (
                    user_id, user_name, datetime.now(), mode,
                    weights_json, scores_json, final_score, weak-
nesses_json
                ))
                conn.commit()
            logger.info(f"Success: Saved assessment for {user_id}")
        except Exception as e:
            logger.error(f"DB Save Error: {e}")
```

# ДОДАТОК D

## Код файлу topsis.py

```
import numpy as np
from typing import List, Tuple

class TOPSISCalculator:
    def __init__(self, subcriteria: List[str], criteria_groups:
List[int]):
        self.subcriteria = subcriteria
        self.criteria_groups = criteria_groups
        self.n = len(subcriteria)

    def calculate_topsis_score(self, user_scores: np.ndarray, crite-
ria_weights: np.ndarray):
        normalized_scores = user_scores / 10.0

        ideal_norm = np.ones(self.n)
        anti_ideal_norm = np.full(self.n, 0.1)

        local_weights = np.zeros(self.n)
        group_counts = np.bincount(self.criteria_groups)

        for i, group_idx in enumerate(self.criteria_groups):
            local_weights[i] = criteria_weights[group_idx] /
group_counts[group_idx]

        weighted_user = normalized_scores * local_weights
        weighted_ideal = ideal_norm * local_weights
        weighted_anti_ideal = anti_ideal_norm * local_weights

        d_plus = np.sqrt(np.sum((weighted_user - weighted_ideal) ** 2))
        d_minus = np.sqrt(np.sum((weighted_user - weighted_anti_ideal) **
2))

        if (d_plus + d_minus) == 0:
            score = 0
        else:
            score = d_minus / (d_plus + d_minus)

        pmf_index = 2
        if user_scores[pmf_index] < 3.0:
            score *= 0.6

        return score, normalized_scores, {}

    def analyze_weaknesses(self, user_scores: np.ndarray, threshold=6.0):
        weaknesses = []
        for i, score in enumerate(user_scores):
            if score < threshold:
```

```
                weaknesses.append((self.subcriteria[i], score))
        return sorted(weaknesses, key=lambda x: x[1])

def create_topsis_subcriteria():
    subs = [
        "Технологічна зрілість", "MVP / Якість продукту",
        "Product-Market Fit", "Об'єм ринку (TAM)",
        "Компетенції команди", "Операційна ефективність",
        "Фінансова стійкість", "Юридична захищеність"
    ]
    groups = [0, 0, 1, 1, 2, 2, 3, 3]

    emojis = ["⚙️", "💎", "🎯", "📈", "👥", "🔁", "💵", "⚖️"]

    help_texts = [
        "Наскільки готова технологія? (1=Ідея, 10=Стабільна версія)",
        "Якість реалізації продукту? (1=Прототип, 10=Готовий продукт)",
        "Чи хоче ринок ваш продукт? (1=Ні, 10=Черга клієнтів)",
        "Розмір ринку? (1=Вузька ніша, 10=Глобальний ринок)",
        "Досвід команди? (1=Студенти, 10=Серійні підприємці)",
        "Чи налагоджені процеси? (1=Хаос, 10=Чіткі інструкції)",
        "Грошовий запас? (1=На місяць, 10=Прибутковий бізнес)",
        "Захист IP та договори? (1=Відсутні, 10=Повний пакет)"
    ]

    return subs, groups, emojis, help_texts
```

# ДОДАТОК Е

## Код файлу utils.py

```
import io
import numpy as np
import matplotlib.pyplot as plt
from typing import List
import matplotlib

matplotlib.use('Agg')

class ReportGenerator:

    @staticmethod
    def generate_radar_chart(categories: List[str], user_scores: np.ndar-
ray) -> io.BytesIO:
        labels = categories
        num_vars = len(labels)
        angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).to-
list()

        values = np.concatenate((user_scores, [user_scores[0]]))
        angles += angles[:1]

        ideal = np.full(num_vars + 1, 10.0)

        fig, ax = plt.subplots(figsize=(10, 10), sub-plot_kw=dict(projec-
tion='polar'))

        fig.patch.set_facecolor('white')
        ax.set_facecolor('#F8F9FA')
        ax.grid(color='#BDC3C7', linestyle='--', linewidth=0.5, alpha=0.7)
        ax.spines['polar'].set_visible(False)

        ax.plot(angles, ideal, color='#2ECC71', linewidth=1, linestyle='--
', la-bel='Benchmark (Ideal)', alpha=0.6)

        ax.plot(angles, values, color='#2980B9', linewidth=2.5, lin-
estyle='solid', label='Ваш Проєкт')
        ax.fill(angles, values, color='#3498DB', alpha=0.35)

        ax.set_xticks(angles[:-1])
        ax.set_xticklabels(labels, size=10, weight='bold',
color='#34495E')
        ax.set_ylim(0, 10)
        ax.set_yticks([2, 4, 6, 8, 10])
        ax.set_yticklabels(['2', '4', '6', '8', '10'], color='#7F8C8D',
size=9)
```

```
        ax.tick_params(pad=25)

        plt.title("SCALING READINESS PROFILE", size=16, weight='bold',
col-or='#2C3E50', y=1.08)
        legend = ax.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1),
frameon=True)
        legend.get_frame().set_edgecolor('#BDC3C7')

        buf = io.BytesIO()
        plt.tight_layout()
        plt.savefig(buf, format='png', dpi=150, bbox_inches='tight', face-
col-or='white')
        buf.seek(0)
        plt.close()
        return buf

    @staticmethod
    def _analyze_imbalances(user_scores: np.ndarray) -> List[str]:
        insights = []
        tech, mvp, pmf, tam, team, ops, fin, legal = user_scores

        if tech > 7 and pmf < 4:
            insights.append(
                "⚡ <b>Ризик 'Technology Push':</b> Ви будуєте ідеальну
технологію без підтвердженого попиту.\n"
                "👉 <i>Дія: Зупиніть розробку (Freeze Code). Всі ресурси на
CustDev та продажі.</i>"
            )

        if tam > 8 and mvp < 4:
            insights.append(
                "🦄 <b>Ризик 'Fantasy Startup':</b> Ви орієнтуєтесь на ве-
личезний ринок, але продукт ще занадто сирий.\n"
                "👉 <i>Дія: Звузьте нішу (Niching down). Доведіть цінність
на ма-лому сегменті.</i>"
            )

        if fin > 7 and pmf < 5:
            insights.append(
                "🔥 <b>Ризик 'Premature Scaling':</b> Ви маєте бюджет, але
продукт не готовий до ринку.\n"
                "👉 <i>Дія: Не масштабуйте маркетинг! Витрачайте гроші
тільки на експерименти.</i>"
            )

        if pmf > 7 and fin < 4:
            insights.append(
                "💸 <b>Ризик 'Growth Trap':</b> Ринок вимагає ваш продукт,
але у вас закінчуються ресурси.\n"
                "👉 <i>Дія: Терміново піднімайте інвестиції (Series A) під
метрики росту.</i>"
```

```
            )

        if tam > 7 and team < 5:
            insights.append(
                "🥷 <b>Ризик виконавця:</b> Потенціал ринку перевищує можливості поточної команди.\n"
                "👉 <i>Дія: Найм сильних C-level менеджерів (CTO/CMO) або адвайзерів.</i>"
            )

        if pmf > 6 and fin > 6 and ops < 4:
            insights.append(
                "🌪️ <b>Ризик 'Operational Chaos':</b> Виросте навантаження — система впаде.\n"
                "👉 <i>Дія: Впровадьте CRM, опишіть SOP та автоматизуйте рутину.</i>"
            )

        if (pmf > 7 or fin > 7) and legal < 4:
            insights.append(
                "⚖️ <b>Юридична вразливість:</b> Успішний бізнес без захисту активів.\n"
                "👉 <i>Дія: Аудит IP, оформлення договорів з командою та клієнта-ми.</i>"
            )

        if pmf > 7 and tech < 4:
            insights.append(
                "🏗️ <b>Критичний техборг:</b> Продукт продається, але архітектура не витримає навантаження.\n"
                "👉 <i>Дія: Рефакторинг. Плануйте перехід від MVP до Scalable Ar-chitecture.</i>"
            )

        if not insights and max(user_scores) < 6:
            insights.append(
                "🐢 <b>Ризик стагнації:</b> Проєкт розвивається надто повільно або команда втратила фокус.\n"
                "👉 <i>Дія: Потрібен радикальний перегляд стратегії або Pivot.</i>"
            )

        return insights

    @staticmethod
    def format_readiness_message(score: float, weaknesses, weights, criteria_names, user_scores, mode_name) -> str:
        if score >= 0.75:
            status_icon, status_text = "🚀", "READY TO SCALE"
```

```
            desc = "Системні показники в нормі. Проєкт готовий до Series
A."
        elif score >= 0.50:
            status_icon, status_text = "⚠️", "NEEDS OPTIMIZATION"
            desc = "Масштабування ризиковане. Є критичні дисбаланси."
        else:
            status_icon, status_text = "🛑", "TOO EARLY"
            desc = "Занадто рано. Фокусуйтесь на пошуку Product-Market
Fit."

        bar = "▓" * int(score * 20) + "░" * (20 - int(score * 20))

        msg = f"""
<b>📊 ЗВІТ ГОТОВНОСТІ СТАРТАПУ</b>
═══════════════════════
<b>Статус:</b> {status_icon} {status_text}
<b>Індекс готовності:</b> {score:.1%}
<code>[{bar}]</code>

<b>Стратегія:</b> {mode_name}
<b>Вердикт:</b> <i>{desc}</i>

═══════════════════════
🧠 <b>СТРАТЕГІЧНІ ІНСАЙТИ (AI Analysis):</b>
"""
        insights = ReportGenerator._analyze_imbalances(user_scores)
        if insights:
            for insight in insights:
                msg += f"\n{insight}\n"
        else:
            if score < 0.6:
                msg += "\n💡 <b>Порада:</b> <i>Критичних диспропорцій
немає, але загальний рівень низький. Потрібен рівномірний ріст.</i>\n"
            else:
                msg += "\n✅ <i>Структурних ризиків не виявлено. Розвиток
гармонійний.</i>\n"

        if weaknesses:
            msg += "\n🔻 <b>ВУЗЬКІ МІСЦЯ (TOP-3):</b>\n"
            for name, val in weaknesses[:3]:
                msg += f"❗ {name}: <b>{val:.1f}/10</b>\n"

        return msg

def create_saaty_scale_keyboard():
    return [
        [("Left Stronger (+9)", "9_left"), ("(+5)", "5_left"), ("(+3)",
"3_left")],
        [("⚡ EQUAL (1) ⚡", "1")],
        [("(+3)", "3_right"), ("(+5)", "5_right"), ("Right Stronger (+9)",
"9_right")]
```

```
    ]

def create_rating_keyboard():
    return [
        [("1", "1"), ("2", "2"), ("3", "3"), ("4", "4"), ("5", "5")],
        [("6", "6"), ("7", "7"), ("8", "8"), ("9", "9"), ("10", "10")]
    ]
```