**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методики оптимізації вибору оптимальних інвестиційних портфелів за допомогою аналізу часових рядів»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
*(код, найменування спеціальності)*
освітньо-професійної програми «Інженерія програмного забезпечення»
*(назва)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Денис СУМІН
*(підпис)*

Виконав: здобувач вищої освіти групи ПДМ-63
Денис СУМІН

Керівник: Ірина ЩЕРБИНА
*к.т.н., доцент*

Рецензент: _____
*науковий ступінь, вчене звання* *Ім'я, ПРІЗВИЩЕ*

**Київ 2024**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Суміну Денису Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка методики оптимізації вибору оптимальних інвестиційних портфелів за допомогою аналізу часових рядів»

керівник кваліфікаційної роботи Ірина ЩЕРБИНА к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література,параметри інвестиційних портфелів, вимоги до інвесттиційних портфелів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1.Вивчення сучасних методів управління портфелями
2.Аналіз платформ вибору інвестиційних портфелів з точки зору технологічних рішень
3.Розробка системи вибору оптимальних інвестиційних портфелів

5. Перелік графічного матеріалу: *презентація*

1.Аналіз існуючих інструментів вибору інвестиційних портфелів
2.Порівняльний аналіз платформ
3.Основні формули для аналізу інвестиційного портфелю
4.Алгоритм часових рядів для оптимізації вибору інвестиційних портфелів
5.Результати моделювання

6. Дата видачі завдання «19» жовтня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз наявної науково-технічної літератури | 19.10-05.11.23 | |
| 2 | Вивчення матеріалів для аналізу вибору інвестиційних портфелів | 06.11-12.11.23 | |
| 3 | Дослідження моделей вибору інвестиційних портфелів | 13.11-19.11.23 | |
| 4 | Аналіз платформ вибору інвестиційних портфелів з точки зору технологічних рішень | 20.11-26.11.23 | |
| 5 | Проведення процедур валідації та верифікації розробленої моделі | 27.11-03.12.23 | |
| 6 | Оцінювання та інтерпретація отриманих даних з врахуванням аспектів оптимізації | 04.12-10.12.23 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 11.12-20.12.23 | |
| 8 | Розробка демонстраційних матеріалів | 21.12-29.12.23 | |

Здобувач вищої освіти _____ Денис СУМІН
*(підпис)*

Керівник
кваліфікаційної роботи _____ Ірина ЩЕРБИНА
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 97 стор., 13 табл., 36 рис., 30 джерел.

*Мета роботи* – реалізація системи для оптимізації вибору інвестиційних портфелів на базі сучасних технологічних рішень.

*Об'єкт дослідження* – процес вибору інвестиційних портфелів в умовах фінансових ринків.

*Предмет дослідження* – реалізація алгоритму часових рядів для оптимізації вибору інвестиційних портфелів, який включає в себе методики визначення ризику, очікуваної доходності кожного інвестиційного активу та коефіцієнту кореляції між активами.

*Короткий зміст роботи*: В межах виконаного проекту здійснено глибокий огляд наукових публікацій, зокрема аналіз методів портфельної оптимізації, ключових фінансових інструментів та ринків. Сформовано методичний підхід до дослідження, який ґрунтується на аналізі вторинних фінансових даних, включаючи акції, облігації та ринок нерухомості. Була сконструйована математична модель, яка використовує часові ряди для оптимізації вибору портфелю.

КЛЮЧОВІ СЛОВА: ІНВЕСТИЦІЙНІ ПОРТФЕЛІ, ЧАСОВІ РЯДИ.

## ABSTRACT

Text part of the master's qualification work: 97 pages, 36 pictures, 13 table, 30 sources.

The purpose of the work – development of a system to enhance the selection of investment portfolios through the utilization of contemporary technological solutions

Object of research – the act of selecting investment portfolios within the context of financial markets.

Subject of research – deploying a time series algorithm to enhance the optimization of investment portfolio selection, encompassing methodologies for assessing risk, estimating the expected return for each investment asset, and determining the correlation coefficient among assets.

Summary of the work: As part of the concluded project, a thorough examination of scholarly works was undertaken, specifically delving into the analysis of methods for optimizing portfolios, crucial financial instruments, and markets. A systematic research approach has been developed, relying on the analysis of secondary financial data, encompassing stocks, bonds, and the real estate market. A mathematical model was formulated, employing time series techniques to refine the selection of portfolios.



KEYWORDS: INVESTMENT PORTFOLIOS, TIME SERIES

**ЗМІСТ**

# ВСТУП

На сучасному етапі економічного розвитку України та світу, вибір оптимальних інвестиційних портфелів є одним з найбільш актуальних і складних завдань у сфері фінансового менеджменту та економічної стратегії. Формування інвестиційних портфелів зазнає безперервної взаємодії з численними зовнішніми та внутрішніми чинниками, які інтенсивно розвиваються у зв'язку з глобалізацією, технологічним прогресом та соціально-економічними трансформаціями.

Підстави та вихідні дані для розробки даної теми базуються на численних теоретичних моделях, емпіричних дослідженнях, а також на потребах реальної економіки, які вимагають все більшої точності, оперативності та адаптивності в управлінні інвестиційними процесами.

При аналізі академічних та практичних джерел у сфері управління інвестиційними портфелями виявляється ряд обмежень. Ці недоліки стають критичними, коли мова йде про інтеграцію авангардних технологій, таких як алгоритми часових рядів, машинне навчання, аналіз великих даних та інші. Стандартні моделі, такі як модель Марковіца, хоч і є фундаментальними, але часто базуються на історичних даних та статичних принципах. Це може бути недостатнім для адекватної реакції на динамічно змінювані умови ринку.

Актуальність даного дослідження виражається у декількох ключових аспектах, які є важливими для економічної стабільності та розвитку не тільки на рівні окремих організацій, але й для України в цілому:

– волатильність ринку. Сучасні ринки характеризуються високою волатильністю, що зумовлено геополітичними напруженнями, економічними санкціями та іншими незрозумілими факторами. У цьому контексті, використання сучасних технологій для оптимізації інвестиційних портфелів стає не просто актуальним, але і критично необхідним;

– глобалізація інвестицій. Зростання міжнародних інвестицій потребує нових методологій та технологічних рішень, здатних адаптуватися до різних юрисдикцій, ринків та валют;

– технологічний прогрес. Розвиток технологій надає нові можливості для ефективного управління активами, але також вимагає перегляду існуючих методологій;

– національна безпека. Ефективне управління інвестиційними портфелями є важливим і для забезпечення економічної безпеки держави, особливо в періоди економічних та політичних криз;

– академічний інтерес. Тема також актуальна з точки зору наукових досліджень, оскільки існуючі теорії та методики часто не враховують нові технологічні можливості, що може обмежувати їх ефективність.

Мета даного дослідження полягає в реалізації системи для оптимізації вибору інвестиційних портфелів на базі сучасних технологічних рішень.

З метою досягнення цієї мети можна виокремити основні задачі:

– систематизація та аналіз існуючих наукових підходів та методологій в області вибору та оптимізації інвестиційних портфелів. Передбачає аналітичний огляд літературних джерел та вивчення сучасних методів, інструментів та платформ, які застосовуються в даній сфері;

– розробка критеріїв для оцінювання ефективності інвестиційних стратегій та їх оптимізація. Включає в себе створення та оптимізацію математичної моделі для вибору інвестиційних портфелів, а також проведення валідації та верифікації розробленої моделі;

– визначення технологічних аспектів та архітектурних рішень для розробки програмного продукту. Включає оцінку та вибір технологічних компонентів, аналітичний огляд вимог, специфікацію розробки бази даних та проектування архітектури програмного забезпечення;

– проведення процесу розробки, адаптації та верифікації програмних компонентів з використанням оптимізаційних технік. Задача передбачає

конструювання та оптимізацію програмних компонентів, їх тестування, а також організацію та виконання експериментальних випробувань.

Об'єктом дослідження є процес вибору інвестиційних портфелів в умовах фінансових ринків.

Предметом дослідження є реалізація алгоритму часових рядів для оптимізації вибору інвестиційних портфелів, який включає в себе методики визначення ризику, очікуваної доходності кожного інвестиційного активу та коефіцієнту кореляції між активами.

Наукова новизна одержаних результатів дослідження полягає в розробці та впровадженні алгоритму часових рядів для оптимізації вибору інвестиційних портфелів. Цей алгоритм представляє собою якісний внесок в теорію і практику управління інвестиційними ресурсами з декількох ключових позицій:

– використання часових рядів як базового математичного апарату для аналізу ринкових даних є відносно новим підходом в контексті інвестиційних портфелів. Такий метод дозволяє з більшою точністю прогнозувати динаміку активів, що на практиці може призвести до зниження ризиків та підвищення потенційної доходності;

– алгоритм інтегрує в себе розрахунок критичних параметрів інвестиційного портфеля: ризику, очікуваної доходності кожного активу, а також коефіцієнту кореляції між активами. В сукупності це дозволяє досягти оптимального співвідношення між активами в портфелі;

– відмінність даного алгоритму від існуючих методів полягає у його спроможності адаптуватися до швидко змінюваних ринкових умов. Практична цінність розробленого в рамках роботи алгоритму на базі часових рядів може бути використана в різних сферах фінансової діяльності, зокрема в управлінні інвестиційними портфелями. Алгоритм спрямований на оптимізацію портфеля шляхом кількісного визначення ризику, очікуваної доходності та коефіцієнта кореляції між активами, що уможливлює більш прогнозоване та ефективне управління капіталом.

# 1 АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ З ВИБОРУ ПОРТФЕЛІВ

## 1.1 Огляд наукової літератури та існуючих підходів до вибору інвестиційних портфелів

Інвестиційні портфелі представляють собою критичний механізм для оптимізації капітальних розміщень. Ці комплексні структури можуть включати в себе різноманітність фінансових інструментів, від акцій до облігацій, і від валют до товарних деривативів. Академічна література в області фінансів засвідчує значну кількість емпіричних та теоретичних досліджень, спрямованих на аналіз ефективності та ризикопрофілю інвестиційних портфелів[1].

Одним з ключових параметрів у управлінні інвестиційними портфелями є диверсифікація, що є важливою змінною у зниженні загального рівня ризику та потенційного збільшення доходності. Це підтверджено в науковому дослідженні «Diversification Across Time», де автори розглядають динаміку довготермінової диверсифікації як інструмент підвищення рентабельності портфеля [2].

Центральною гіпотезою в даній області є принцип, згідно якого розподіл інвестицій між різними класами активів (акції, облігації, нерухомість, валютні інструменти та ін.) або географічними зонами може функціонувати як механізм демпфування ризиків. Така диверсифікація може впливати на редукцію волатильності портфеля через компенсаційний ефект між позитивними та негативними коливаннями ринкових цін різних активів. Тобто, невдачі в одному секторі або географічному регіоні часто балансуються успіхами в інших, що веде до зниження можливості кумулятивних втрат.

В реаліях фінансової науки, диверсифікація активів в інвестиційному портфелі актуалізує себе як окрема субдисципліна, взаємодія якої з часовими періодами та ринковими секторами може суттєво вплинути на довгострокову прибутковість. Диверсифікація в цьому контексті функціонує як інструментарій

для капталізації на динаміці різних ринків і секторів, тим самим дозволяючи інвестору максимізувати доходи незалежно від ринкових флуктуацій.

Цей процес не тільки характеризується вибором диверсифікованих активів, але також темпоральними вимірами стратегії. Довгострокове дотримання диверсифікаційних принципів може ампліфікувати ефекти компаундування та забезпечити можливість для адаптивної корекції портфеля відповідно до ринкових індикаторів.

Паралельно, управління ризиком у інвестиційних портфелях піддавалось академічному аналізу, як це видно з дослідження «Risk Management in Investment Portfolios», автори якого, Ендрю Енгелі та Майкл Форд, підкреслюють значущість активного ризикового управління для мінімізації можливих втрат, особливо під час економічних криз [3].

Такий фокус на ризику робить його управління невіддільним компонентом оптимізації інвестиційного портфеля. Стратегії такого управління, засновані на наукових принципах, мають за ціль зниження можливих втрат, що можуть мати місце в періоди фінансової нестабільності, та збереження капіталу для подальшого використання в інвестиційних можливостях.

Процедурний ландшафт управління ризиком ініціюється з фаз ідентифікації та квантитативної оцінки ризикових факторів. Цей етап вимагає детального аналізу ринкової кон'юнктури, включаючи волатильність інвестиційних активів та їхню корелятивну взаємодію з іншими активами в портфелі. Академічні дослідження в цій області здійснюються за допомогою застосування різноманітних статистичних методів і моделей, таких як дисперсія доходу, стандартне відхилення, коефіцієнт бета та показник Value at Risk [4].

По завершенні фази ідентифікації та оцінки, науковці розробляють стратегії для мінімізації виявлених ризиків. Ці стратегії можуть охоплювати такі напрямки, як диверсифікація активів портфеля, використання фінансових деривативів для хеджування ризиків, балансування структури портфеля, а також систематичний моніторинг і переоцінка активів.

Емпіричні дослідження, проведені в цій сфері, підтверджують, що активне управління ризиком може суттєво амортизувати фінансові втрати під час економічних рецесій та сприяти збереженню капіталу для подальших інвестиційних операцій. Ці висновки базуються на статистичному аналізі динаміки інвестиційних портфелів в різні хронологічні періоди, включаючи етапи системних фінансових криз.

Додатково, академічна література взаємодіє з питаннями оптимізації стратегічних підходів до управління інвестиційним портфелем. У праці «Strategies for Investment Portfolios», автори Вільям Шарп та Джеффрі Янг аргументують, що змішане використання різних інвестиційних стратегій, таких як пасивне та активне інвестування, а також технічний аналіз, може сприяти ефективності портфеля та диверсифікації ризиків. Їхні дослідження підтверджують, що комплексний підхід до інвестиційної стратегії може кумулятивно покращити фінансові показники портфеля [5].

У проведенні згаданих досліджень використовувалася методологія, яка об'єднує історичний аналіз інвестиційних портфелів, сформованих на основі різних стратегій управління активами. Статистичні інструменти, зокрема регресійний аналіз, кореляційна математика та тестування наукових гіпотез, були пристосовані для кількісного вимірювання ефективності комбінованих інвестиційних стратегій.

Дослідники встановили, що інвестиційні портфелі, які базуються на синтезі пасивних, активних та технічних методологій, виявляють збільшену доходність та зменшену волатильність в порівнянні з портфелями, що орієнтуються на єдиний метод управління активами.

Подальше розширення даного підходу було зроблене у дослідженні «Factor Investing and Asset Allocation», авторами якого є Андреас Готцман та Армін Хегедюс. Вони демонструють, що інтеграція різноманітних факторів, таких як капіталізація компаній, стиль інвестиційної активності, географічний фокус та інші, може привести до оптимізації показників портфеля та до мінімізації асоційованих ризиків.

Таким чином, згідно з результатами проведених досліджень, інвестиційні портфелі можуть бути налаштовані з урахуванням цілого спектру критеріїв та факторів, включаючи не лише стратегічні прийоми управління, але й корпоративні характеристики, стилістику інвестиційної поведінки та регіональну алокацію активів. Використання факторного аналізу, який включає в себе параметри, такі як вартість, моментум та низька волатильність, виявляється корисним у підвищенні ефективності портфеля та у зменшенні інвестиційних ризиків [6].

У науковому дослідженні була приділена увага аналізу історичних показників доходності акцій, а також асоційованих характеристик, таких як цінові мультиплікатори, варіабельність цінових коливань та ліквідність акцій. Для дослідження комплексних взаємозв'язків між цими параметрами було застосовано методологію факторного аналізу, що дозволяє розкласти даний датасет на компоненти, що найкраще характеризують змінні.

Завдяки факторному аналізу, ідентифіковано групи акцій із схожими характеристиками, що, у свою чергу, дозволило оцінити їх колективний вплив на показники доходності інвестиційного портфеля.

Емпіричні результати демонструють, що портфелі, структуровані з урахуванням вищезгаданих факторів, відзначаються підвищеною доходністю та зниженою волатильністю у порівнянні з конвенційними інвестиційними стратегіями. Ці висновки були піддані статистичній верифікації через застосування t-тесту для оцінки статистичної значущості розбіжностей, а також F-тесту для аналізу адекватності використаної моделі даним.

В науковому опусі «Portfolio Optimization with High Frequency Data», авторство якого належить Карен Селім та Пьер Кохен, проводиться аналіз можливості використання високочастотних даних для підвищення точності прогнозування і оптимізації структури інвестиційних портфелів. Термін «високочастотні дані» реферує до інформації, яка реєструється з великою тимпоральною резолюцією, такою як щосекундні цінові зміни акцій чи щохвилинні об'єми торгів.

Методологія дослідження передбачала використання високочастотних фінансових метрик, зокрема цінових котирувань, торговельних об'ємів, індексів біржового торгування тощо. Для аналітичної обробки даних були застосовані алгоритми машинного навчання, у тому числі архітектури нейронних мереж та методики глибокого навчання. Ці сучасні аналітичні інструменти дозволяють ідентифікувати латентні кореляції та неочевидні взаємозв'язки у великих датасетах.

Отримані результати свідчать, що моделі, які були навчені на основі високочастотних даних, демонструють вищу точність прогнозування порівняно з моделями, заснованими на денних агрегованих даних. Інтерпретація результатів також виявила, що портфелі, оптимізовані відповідно до таких високоточних прогнозів, характеризуються збільшеною доходністю та зниженою волатильністю.

Валідація емпіричних знахідок здійснювалася шляхом порівняння дослідних портфелів, сформованих на основі різних прогностичних моделей. Для статистичної оцінки значущості отриманих результатів були використані статистичні критерії, такі як t-тест і Шарпів коефіцієнт. Таким чином, дане дослідження підкреслює важливість інтеграції високочастотних даних в сучасні методики управління інвестиційними портфелями.

На підставі аналізу наукових робіт у сфері управління інвестиційними портфелями можна констатувати, що цей інструмент є ключовим для капіталовкладень. Успішна диверсифікація активів та ретельне управління ризиками можуть значно підвищити рентабельність портфелів, в той же час мінімізуючи потенційні ризики. Більш того, застосування різноманітних інвестиційних стратегій та інкорпорація різних факторних моделей можуть додатково оптимізувати доходність і знизити рівень ризику.

Зокрема, вчені дослідження підкреслюють переваги використання високочастотних даних у процесі моделювання і оптимізації інвестиційних портфелів. Цей метод дозволяє забезпечити вищу точність прогнозів, що, в свою чергу, може призвести до ефективніших інвестиційних рішень.

Проте, варто зазначити, що фінансові ринки залишаються вкрай комплексними та непередбачуваними структурами. Відтак, жодна методика чи стратегія не може гарантувати абсолютного успіху в інвестиційній діяльності. Тому використання науково-обґрунтованих методів та стратегій повинно супроводжуватися критичним аналізом та постійним моніторингом, з метою адаптації до змінюваних умов ринку [7].

## 1.2 Дослідження методології оптимізації портфелів та їх технологічної реалізації

Фінансові інструменти функціонують як механізми для управління капіталом, що включають інвестування, збереження та інші операції капіталообігу, такі як кредитування. До цієї широкої категорії входять акції, облігації, фондові індекси, валютні пари та ряд інших активів.

В аспекті акцій, вони є долею власності в корпорації і емітуються з метою залучення зовнішнього фінансування. Акції поділяються на два основних типи: звичайні та привілейовані. Звичайні акції надають право на частку в прибутках компанії та дають можливість участі в прийнятті стратегічних рішень через голосування на загальних зборах акціонерів. Привілейовані акції, в свою чергу, надають своїм власникам додаткові фінансові переваги, такі як пріоритет в виплаті дивідендів або при ліквідації активів, але зазвичай не надають права голосу в управлінських питаннях.

Облігації служать зобов'язаннями фінансового характеру, емітованими корпоративними або державними структурами з прямою метою мобілізації капітальних ресурсів. Ці фінансові інструменти мають різну терміновість – від короткострокових до довгострокових – та, як правило, пропонують вищий рівень доходності порівняно з традиційними депозитними вкладами. Втім, потрібно зауважити, що інвестування в облігації не звільнене від різноманітних фінансових ризиків, включаючи можливість дефолту по виплатам.

В контексті міжнародних валютних операцій, валютні пари представляють собою засоби торгівлі, що включають базову та котирувальну валюту. Вартість одиниці базової валюти вимірюється у термінах котирувальної валюти, формуючи тим самим курс обміну.

Щодо інших фінансових інструментів, до них відносяться, між іншим, ф'ючерсні контракти, опціони та додаткові інструменти, які зазвичай є доступними лише для інвесторів з високим рівнем кваліфікації. Ці інструменти можуть забезпечити високу доходність, але їх характеризує високий рівень ризику, особливо з урахуванням значних коливань їхніх ринкових цін в короткостроковій перспективі.

Кожен фінансовий інструмент має свою сукупність переваг та обмежень, отже, для ефективного використання конкретного інструменту імперативно провести глибокий аналіз його характеристик і потенційних ризиків. Оптимальний підхід передбачає сукупну оцінку доступних фінансових інструментів для вибору найбільш адекватного у конкретному економічному контексті. Науково обгрунтована стратегія інвестування та її консеквентна реалізація є ключовими факторами успіху. Окрім того, не менш важливою є постійна моніторингова діяльність щодо ринкових умов, яка дозволить інвестору оперативно реагувати на динамічно змінювані обставини.

Торговельні платформи фінансових інструментів відзначаються інтраструктурною різноманітністю, що включає в себе:

– ринок акцій. Сфера, де корпорації мобілізують капітал через реалізацію акціонерних пакетів. Інвестиційна активність на цьому ринку базується на купівлі та продажу акцій з метою отримання капіталізаційних прибутків та дивідендних виплат;

– борговий ринок. Цей сектор дозволяє урядовим структурам та промисловим комплексам залучати фінансування за допомогою випуску облігацій. Такі інструменти, як правило, є довгостроковими та генерують доход у вигляді процентних виплат;

– товарний ринок. Тут проводяться операції з різноманітними сировинними активами – від дорогоцінних металів до сільськогосподарської продукції. Інвестори взаємодіють на цьому ринку з метою капіталізації на динаміці цін, а також для спекулятивних цілей;

– валютний ринок. Платформа для взаємних операцій з національними валютами. Інвестиційна привабливість цього сектора полягає у можливості отримання прибутку від коливань валютних курсів, а також спекулятивних транзакцій.

Для кожного з цих ринків характерна своя сукупність переваг та потенційних ризиків. Отже, інвесторам не лише рекомендується, але і є обов'язковим проведення глибокого аналізу кожного сектора з метою визначення оптимального напряму інвестування, адаптованого до конкретних фінансово-економічних умов [8].

## 1.3 Вивчення сучасних методів управління портфелями та їх характеристик

У сфері інвестиційної діяльності функціонує численні методології структурування портфелів, зорієнтовані на адаптацію до конкретних інвестиційних цілей, рівня ризику та інших критеріїв інвестора.

Однією з найвидатніших є Марковіцька модель оптимізації портфелю, що є базовим методологічним інструментом у сучасному портфельному управлінні та корпоративних фінансах. За розробку цього підходу Гаррі Марковіц було присуджено Нобелівську премію з економіки у 1990 році.

Інноваційна суть Марковіцького підходу полягає в концепції «ефективного фронту». Ця концепція означає вибір оптимальної комбінації активів, яка або максимізує дохідність за фіксованого рівня ризику, або мінімізує ризик за заданої очікуваної дохідності [9].

Для реалізації цієї методики ключовими є три основні параметри:

– очікувана дохідність. Цей показник відображає математичне очікування можливої дохідності портфелю та розраховується як вагове середнє дохідності будь-якого з активів портфелю;

– волатильність (стандартне відхилення). Цей критерій є мірою ризику портфелю. Його обчислення здійснюється на основі стандартних відхилень дохідності активів, що входять у портфель, з урахуванням їхньої взаємозалежності;

– кореляція між активами. Показник кореляції відображає ступінь лінійної залежності між дохідністю різних активів портфелю і відіграє значущу роль у визначенні волатильності портфелю.

Застосування Марковіцівської моделі передбачає глибокий аналітичний огляд цих трьох параметрів для кожного активу, що входить у портфель. Це забезпечує можливість побудови портфелю, який належним чином корелює з інвестиційними цілями і ризиковим профілем інвестора.

До переваг методики Марковіца можна відзначити:

– мінімізація фінансового ризику. Алгоритми методики дозволяють знизити варіативність портфелю, що веде до стабільної дохідності в довготривалій перспективі;

– оптимальна алокація капіталу. Методика забезпечує можливість вибору активів з максимальною очікуваною дохідністю на одиницю ризику, що сприяє досягненню вищої загальної дохідності портфелю;

– фундаментальна диверсифікація. Принцип диверсифікації в активах з різними характеристиками ризику та доходності дозволяє знизити взаємозалежність дохідності компонент портфелю, що робить доходність менш чутливою до ринкових коливань.

Однак, існують декілька значущих недоліків:

– обчислювальна складність. Розрахунки, пов'язані з оптимізацією портфелю, можуть бути ресурсомісткими і вимагати спеціалізованого програмного забезпечення;

– прогнозна невизначеність. Модель ґрунтується на прогнозах дохідності та волатильності, які можуть не завжди відображати реальну динаміку ринку;

– проблема кореляційної залежності між активами. Методика припускає незалежність ризикових профілів активів, що може бути не завжди вірною передумовою;

– фінансова доступність. Не всі інвестори можуть мати доступ до ресурсів для повноцінного застосування методики через її високу вартість.

З урахуванням вищезгаданих факторів, методика Марковіца залишається вагомим інструментом для інвесторів, орієнтованих на раціональну алокацію ресурсів із метою мінімізації ризику та максимізації доходу [10].

Методологія оцінки портфелю, представлена Марком Кархартом у 1997 році, є субстанційним розширенням попередньої трьохфакторної моделі Фама-Френча [11]. Ключовою інновацією цієї методології є інтеграція різнорідних факторів для деталізації доходності портфелю акцій:

– ринковий портфель. Цей фактор репрезентує загальну доходність ринку та її вплив на індивідуальний портфель. Таким чином, можна відслідкувати кореляцію портфелю з глобальними ринковими коливаннями;

– SMB (Small Minus Big). Фактор SMB кількісно оцінює взаємодію між доходностями малих та великих компаній. Спостереження, що малі компанії часто показують більшу доходність, інтегровано в модель для отримання більш диференційованого розуміння ринкових тенденцій [12];

– HML (High Minus Low). Фактор HML базується на аналізі акцій з високим та низьким коефіцієнтом book-to-market. Це дозволяє ідентифікувати акції, що імовірно продемонструють високу («value» акції) або низьку («growth» акції) доходність в залежності від їх фундаментальних характеристик [13];

– MOM (Momentum). Четвертий фактор, який був введений Кархартом, вимірює імпульс доходності акцій, базуючись на їх досягненнях у минулому місяці. Це дає можливість антиципувати майбутню поведінку акцій на основі історичних показників [14].

Методологія Кархарта відзначається високою адаптивністю та можливістю комплексного аналізу, завдяки чому вона знайшла широке застосування у сфері управління портфелями та аналітичних дослідженнях. Втім, важливо зазначити, що використання даної моделі вимагає деталізованого розуміння кожного з чотирьох факторів, а також специфіки ринку, на якому проводиться аналіз.

Основні переваги методики Кархарта:

– мінімізація ризиків. Методологія Кархарта пропонує інструментарій для оцінки та оптимізації ризикованості інвестиційного портфеля, забезпечуючи можливість вибору активів із мінімальним ризиком при фіксованому рівні доходності;

– максимізація доходів. За допомогою кількісного аналізу факторів доходності, дана методика дозволяє ідентифікувати активи, які мають потенціал до максимальної доходності при заданому рівні ризику;

– врахування коефіцієнта кореляції. Методика інтегрує в себе розрахунок кореляційних взаємозв'язків між активами, що дозволяє досягти оптимальної диверсифікації і, відповідно, зменшення загального ризику портфеля;

– застосування статистичних методів. Використання статистичних методологій і математичного моделювання дозволяє отримати емпірично обґрунтовані та кількісно точні результати.

Основні недоліки методики Кархарта:

– висока складність. Методика вимагає глибокого математичного та статистичного підготовлення, що може стати проблемою для інвесторів з недостатнім рівнем кваліфікації;

– вимоги до вхідних даних. Для ефективної роботи моделі необхідний доступ до якісних, точних і повних даних щодо доходності та ризиків активів, що може бути непростою задачею;

– неврахування неліквідності. Методика не інкорпорує аналіз можливої неліквідності активів, що може суттєво вплинути на фактичну доходність та ризикованість портфеля;

– обмежена універсальність. Хоча методика Кархарта визнана ефективною для певних типів портфелів, її не можна вважати універсальним рішенням для всіх видів інвестиційних стратегій.

Отже, методологія Кархарта представляє собою високоефективний методичний підхід для конструювання інвестиційного портфеля, який відзначається здатністю до мінімізації ризикованих факторів та максимізації потенційної доходності при стабільному рівні ризику. Проте, інструментарій цієї методології включає в себе серйозні математичні та статистичні моделі, що робить її надзвичайно складною для осіб без спеціалізованої освіти в цих дисциплінах. Також, методика поставляє строгі вимоги до якості та повноти вхідних даних, зокрема з погляду ризиків та доходності активів. У зв'язку з цим, рекомендується, що перед імплементацією методики Кархарта інвестори повинні провести детальний аналітичний огляд та критичну оцінку її придатності у контексті власних інвестиційних стратегій та фінансових цілей.

Методологія, розроблена вченим, відомим як Білд, та що поширено відома під назвою «бета-коефіцієнт портфеля», представляє собою інтелектуальний інструмент для квантитативної оцінки систематичного ризику портфеля цінних паперів.

В арсеналі фінансової математики бета-коефіцієнт служить числовим індикатором, який діагностує ступінь кореляції дохідності інвестиційного портфелю із фінансовою динамікою загального ринку. Відзначимо, що цей коефіцієнт може приймати позитивні, нульові або негативні значення, і в якості такого він характеризує напрям і амплітуду реакції портфеля на ринкові флуктуації [15].

Цей індекс визначає декілька ключових аспектів:

– динаміка цін акцій портфеля в контексті ринкових коливань. Так, бета-коефіцієнт зі значенням 1 свідчить про синхронізацію портфеля з загальним ринком. Значення бета, що перевищує 1, вказує на вищу волатильність портфеля відносно ринку, тоді як значення менше 1 сигналізує про нижчу волатильність;

– ступінь ризикованості інвестиційного портфеля відносно базового ризику ринку. Цей параметр вказує на додаткові ризики чи можливості для зменшення ризиків, які інвестор приймає, обравши певний портфель;

– вплив на диверсифікацію загального інвестиційного портфеля. Бета-коефіцієнт може слугувати компасом для інвестора щодо того, як конкретний портфель вплине на загальний рівень ризику його глобального портфеля.

Осмислене використання методології, базованої на бета-коефіцієнті, забезпечує інвесторам засоби для деталізованого аналізу потенційної волатильності та систематичного ризику інвестиційного портфеля в контексті ринкових умов. Це, у свою чергу, є важливим етапом у прийнятті обґрунтованих інвестиційних рішень.

Переваги методології:

– оперативна простота. Методика, розроблена Білдом, характеризується високою ступенем зрозумілості та легкістю застосування, що робить її доступною для широкого спектру інвесторів. Для квантитативної оцінки потрібні лише дані про доходи та ризики активів в портфелі;

– доступність інформаційних ресурсів. Відкритий доступ до фінансових показників, включаючи ціни акцій, сприяє точному визначенню бета-коефіцієнтів для різноманітних активів та портфелів;

– ідентифікація систематичних ризиків. Коефіцієнт бета адекватно відображає систематичні ризики портфеля, порівнюючи їх з загальноринковими індексами.

Обмеження методології:

– залежність від ринкової волатильності. Методика заснована на премісі стабільності ринкових зв'язків, що може бути порушено в умовах високої нестабільності, внаслідок чого точність розрахунків може бути компрометована;

– залежність від вибору порівняльного індексу. Точність оцінки бета-коефіцієнта суттєво залежить від коректного вибору базового індексу, який відображає динаміку ринку, на якому розміщені активи;

– недоліки в урахуванні індивідуальних характеристик. Методика не враховує такі фактори як ліквідність, дивіденди, якість корпоративного управління та інші, що може призвести до неточних результатів;

– неврахування несистематичних ризиків. Методика ігнорує ризики, що не мають прямого відношення до ринкової динаміки, наприклад, політичні ризики або ризики валютних коливань.

У сукупності, методика Білда представляє собою корисний інструмент для оцінки систематичних ризиків, але її використання вимагає критичного підходу і комплексного аналізу. Інвесторам рекомендується збалансувати переваги та обмеження даної методології для отримання найбільш точної та об'єктивної оцінки ризиків та доходності портфеля.

## 1.4 Аналіз платформ вибору інвестиційних портфелів з точки зору технологічних рішень

*Bloomberg Terminal*

Bloomberg Terminal, також відомий як Bloomberg Professional, є інтегрованою платформою для професійних учасників фінансового ринку. Цей застосунок забезпечує високої якості дані, аналітичні інструменти, а також засоби для ефективного комунікаційного взаємодії. Основна мета полягає у наданні широкого спектру інформаційних та аналітичних сервісів для прийняття обґрунтованих фінансових рішень.

Основні функції платформи включають:

моніторинг ринку. Отримання реального часу інформації про різні класи активів, включаючи акції, облігації, валюти та сировину;

аналіз портфелю. Інструменти для оптимізації інвестиційних портфелів, включаючи розрахунок різних фінансових показників та моделювання ризиків;

торгівельні системи. Функціональність для здійснення торгів, використовуючи різні стратегії та алгоритми.

доступ до новин та аналізу. Інтегровані новинні канали та глибокий аналіз ринкових трендів, стратегій та економічних показників.

На рис. 1.1 зображено інтерфейс користувача платформи «Bloomberg Terminal».


Рис 1.1 Інтерфейс користувача платформи «Bloomberg Terminal»

Переваги Bloomberg Terminal:

– інтегрованість ресурсів. Об'єднує величезну кількість фінансових даних, новин та аналітичних інструментів в єдиному інтерфейсі, що забезпечує зручність управління активами;

– висока якість даних. Гарантує надійність та точність фінансової інформації, що є критично важливим для прийняття інвестиційних рішень;

– розширені аналітичні інструменти. Забезпечує потужні можливості для аналізу ринків, оцінки ризиків та оптимізації портфелів;

– комунікаційні можливості. Містить вбудовані комунікаційні засоби для прямого обміну інформацією та співпраці між фахівцями фінансової галузі;

– глобальне охоплення. Дозволяє отримувати дані з фінансових ринків всього світу, що є незамінним для глобальних інвестиційних стратегій.

Недоліки Bloomberg Terminal:

– висока вартість. Підписка на сервіс є досить коштовною, що може бути недоступно для невеликих інвестиційних компаній або індивідуальних інвесторів;

– складність інтерфейсу. Незважаючи на потужні можливості, інтерфейс може бути незручним та складним для користувачів без спеціалізованої підготовки;

– технічні вимоги. Для ефективної роботи потребує високоякісного інтернет-з'єднання та спеціалізованого обладнання, що збільшує загальні операційні витрати;

– лімітована персоналізація. Незважаючи на широкий спектр функціональних можливостей, платформа може не завжди надавати інструменти для дуже специфічних або нішових інвестиційних потреб.

– залежність від одного провайдера. Висока залежність від даного сервісу може стати ризикованим фактором у випадку технічних проблем або змін у політиці компанії.

В сукупності Bloomberg Terminal представляє собою потужний інструмент для аналізу фінансових ринків, що інтегрує широкий спектр даних та аналітичних засобів. Його переваги полягають у високій якості інформації, глобальному охопленні та розширених комунікаційних можливостях. Проте, висока вартість підписки, складність інтерфейсу та технічні вимоги роблять його менш доступним для невеликих та некваліфікованих інвесторів. З огляду на ці фактори, ефективність використання Bloomberg Terminal в значній мірі залежить від інвестиційних потреб та ресурсів конкретного користувача [16].

*Quantopian*

Quantopian є специфізованою онлайн-платформою, створеною для розробки, тестування та імплементації алгоритмічних торгівельних стратегій (рис. 1.2). Основна місія сервісу полягає у наданні інвесторам та дослідникам

інструментарію для експериментування та впровадження власних ідеї в реальному торгівельному середовищі.

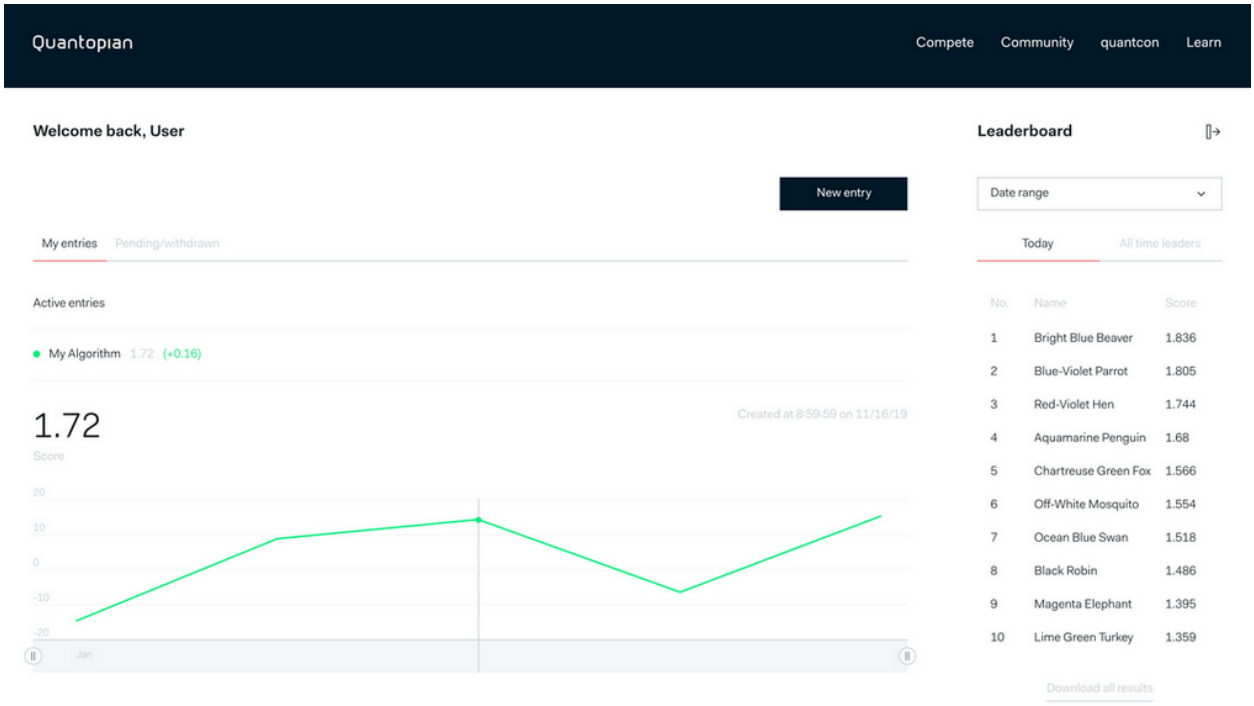



Рис 1.2 Інтерфейс користувача онлайн-платформи «Quantopian»

Основні функції:

– бібліотека даних. Доступ до широкого спектру фінансових даних, які можна використовувати для аналізу і тестування стратегій;

– Backtesting. Засоби для історичного тестування торгівельних алгоритмів на реальних даних.

– інтеграція з Python. Можливість використовувати мову програмування Python для розробки стратегій, що дозволяє використовувати розширений аналітичний інструментарій;

– ком'юніті. Велика спільнота дослідників та програмістів для обміну досвідом та знаннями.

Переваги Quantopian:

– відкритий доступ. Платформа надає безкоштовний доступ до багатого набору фінансових даних та аналітичних інструментів;

– гнучкість. Високий рівень кастомізації, дозволяючи користувачам розробляти дуже специфічні алгоритмічні стратегії;

– колаборативні інструменти. Quantopian пропонує зручні засоби для спільної роботи та обміну стратегіями і даними між користувачами, що підвищує колективний інтелект та сприяє накопиченню знань;

– обширна база даних. Платформа має велику кількість відкритих фінансових даних, доступ до яких може бути корисний для аналітики та розробки стратегій;

– масштабованість. Платформа розроблена з урахуванням можливості легкої масштабованості, що дозволяє користувачам розширювати та модифікувати свої стратегії відповідно до змін на ринку.

Недоліки Quantopian:

– обмеженість ресурсів. Незважаючи на велику кількість доступних даних, набори даних можуть бути обмежені та не завжди актуальні;

– комплексність. Для ефективного використання платформи потрібен високий рівень знань з програмування та фінансової аналітики;

– ризик переоптимізації. Існує ризик переоптимізації алгоритмів, коли стратегія, яка виглядає ефективною на історичних даних, може не показувати себе такою в реальних умовах;

– комерційні обмеження. Деякі функції або набори даних можуть бути доступні лише в комерційних пакетах, що обмежує доступ для індивідуальних користувачів.

Отже, Quantopian є авансованою платформою для розробки алгоритмічних торгівельних стратегій, яка забезпечує користувачам розширені аналітичні можливості та доступ до значущих фінансових даних. Вона пропонує велику гнучкість і кастомізацію, що робить її привабливою для досвідчених інвесторів і аналітиків. Однак, платформа не є бездоганною, зокрема через свою комплексність і потенційний ризик переоптимізації. Таким чином, Quantopian представляє великий інтерес для фахівців в області фінансів, але вимагає від користувачів високого рівня експертизи для ефективного використання [17].

*Wealthfront*

Wealthfront є автоматизованою платформою для управління інвестиційними портфелями (робот-консультантом), розробленою з метою максимізації доходів і мінімізації ризиків для інвесторів різного рівня знань та досвіду.

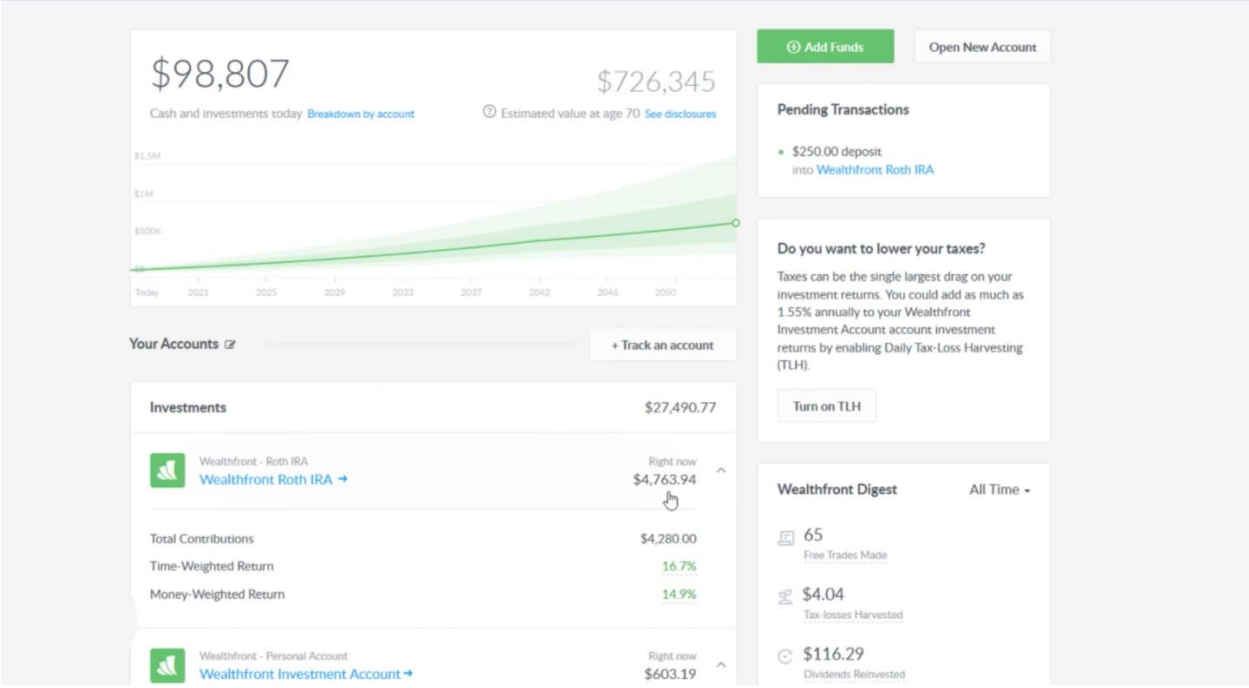



Рис 1.3 Інтерфейс користувача платформи «Wealthfront»

Основні функції системи:

– автоматизований відбір інвестиційних інструментів. Платформа використовує математичні алгоритми для вибору акцій, облігацій та інших фінансових інструментів, що відповідають індивідуальним інвестиційним стратегіям;

– перебалансування портфеля. Wealthfront автоматично перебалансує портфель інвестора з метою забезпечення оптимального співвідношення ризику та доходності;

– податкова оптимізація. Здійснює тактики податкового лосювання (Tax Loss Harvesting), щоб мінімізувати податкові зобов'язання інвестора;

– вкладення у ретайрмент. Платформа також надає можливість управління пенсійними рахунками, такими як IRA та 401(k).

Переваги:

– автоматизація процесів. Платформа дозволяє автоматизувати більшість аспектів інвестиційної діяльності, що сприяє ефективності та зниженню людського фактору у прийнятті рішень;

– податкова оптимізація. Використання стратегій податкового лосювання може призвести до економії коштів на податках, що є значущим для збільшення кінцевої доходності;

– гнучкість. Wealthfront пропонує ряд інвестиційних продуктів, включаючи індивідуальні пенсійні рахунки, що робить платформу відносно універсальною;

– інтуїтивний інтерфейс. Платформа має користувацький інтерфейс, який є доступним навіть для недосвідчених інвесторів.

Недоліки:

– обмеження у виборі активів. Платформа може не надавати доступу до деяких видів інвестиційних інструментів, що може бути недостатнім для досвідчених інвесторів;

– вартість послуг. Хоча робот-консультанти зазвичай здешевлені в порівнянні з традиційними фінансовими консультантами, вони все ж таки не є безкоштовними;

– ризик автоматизації. Залежність від алгоритмічних моделей може внести додатковий ризик через можливі програмні помилки або недосконалості в алгоритмах.

Отже, Wealthfront представляє собою високотехнологічну платформу, орієнтовану на автоматизацію та оптимізацію інвестиційних процесів. Завдяки зручному користувацькому інтерфейсу, можливостям податкової оптимізації та ряду інвестиційних продуктів, дана платформа стає атрактивною для широкого кругу інвесторів. Однак потенційні користувачі повинні бути свідомі вартості послуг, обмеженого спектру доступних активів та ризиків, які можуть виникнути внаслідок автоматизації. В цілому, Wealthfront є цінним інструментом для тих, хто

прагне систематизувати та оптимізувати свою інвестиційну діяльність, не залучаючи великі людські та фінансові ресурси [18].

Здійснивши аналітичний огляд наявних платформ для управління інвестиційними портфелями, варто також систематизувати та зіставити їхні функціональні характеристики. Такий методологічний підхід надасть можливість докладніше проникнути в спектр інвестиційних опцій, які ці системи пропонують (табл. 1.1).

Таблиця 1.1

Порівняльний аналіз платформ

| Характеристика | Bloomberg Terminal | Quantopian | Wealthfront |
|---|---|---|---|
| Функціональні можливості | Доступ до глобальних фінансових даних, новин, аналітики, торгівлі на біржі | Backtesting, торговельні стратегії, бібліотеки для роботи з фінансовими даними | Роботизоване інвестування, податкова оптимізація, автоматична ребалансировка, фінансове планування |
| Цінова політика | Висока: від $20,000 до $25,000 на рік за користувача | Безкоштовно для користувачів, але з обмеженим доступом до даних | Зазвичай 0.25% від загальної суми активів на рахунку |
| Доступність | Обмежена (потребує спеціалізованого обладнання або ПЗ) | Веб-базований доступ, можливість локального запуску коду | Веб-платформа та мобільні додатки |
| Точність аналізу | Висока, завдяки обширній базі даних | Залежить від якості користувацького коду та доступних даних | Автоматизована, але залежна від вибраних алгоритмів |
| Персоналізація | Висока: користувач може налаштувати інтерфейс та показники за власними потребами | Висока: користувач може писати власні алгоритми | Обмежена: користувач може вибирати лише з наявних опцій інвестування |

У контексті проведеного аналізу можна встановити, що кожна з розглянутих платформ (Bloomberg Terminal, Quantopian, Wealthfront) має свої специфічні переваги та обмеження. Bloomberg Terminal вирізняється обширністю доступних фінансових даних та аналітичних інструментів, але є дороговизною та складною у користуванні. Quantopian надає гнучкість у розробці торговельних стратегій, проте вимагає від користувачів високого рівня технічної компетентності. Wealthfront пропонує роботизовані рішення для масового користувача, але має обмежені можливості персоналізації.

Очевидно, що існуючі платформи не можуть задовольнити всі потреби інвесторів, особливо тих, хто шукає збалансоване рішення між функціональністю, вартістю та простотою користування. Тому, існує виправдана необхідність у розробці нового технологічного рішення, яке б комбінувало найкращі характеристики вищезгаданих систем з можливістю адаптації до специфічних потреб користувачів. Це не тільки збільшить ефективність управління інвестиційними портфелями, але і розширить доступ до високоякісних інвестиційних інструментів для широкого спектру інвесторів.

## 1.5 Постановка задачі

Розробити комплексну систему, що використовує алгоритми часових рядів для оптимізації вибору інвестиційних портфелів. Система повинна мати високий ступінь адаптивності до змінних ринкових умов та забезпечувати ефективне управління капіталом.

Завдання:

– аналіз вимог. Визначити функціональні та нефункціональні вимоги до системи;

– розробка архітектури системи. Вибрати та обґрунтувати використання певних технологій та підходів для реалізації системи;

– реалізація алгоритму. Розробити алгоритм на базі часових рядів, що буде інтегрований в систему;

– проектування бази даних. Розробити структуру бази даних для збереження інформації про інвестиційні портфелі, активи, часові ряди та інші релевантні параметри;

– розробка інтерфейсу користувача. Реалізувати інтуїтивно зрозумілий та зручний для користувача інтерфейс;

– тестування та валідація. Провести комплексні тести системи для перевірки її надійності, ефективності та згідності з вимогами;

– документація. Підготувати технічну документацію, що включає специфікації, сценарії використання та інструкції для користувачів;

– оцінка ефективності. Провести аналіз ефективності системи на реальних або модельованих даних.

Система для оптимізації вибору інвестиційних портфелів повинна бути заснована на передових технологічних рішеннях. Зокрема, для програмної реалізації залучена мова програмування C#, яка забезпечує високу продуктивність, надійність та масштабованість. Щодо організації зберігання даних, у роботі застосована система управління базами даних MS SQL Server, яка демонструє високий рівень безпеки та ефективності в управлінні великими обсягами інформації.

Архітектура системи побудована за модульним принципом, що надає можливість швидкої адаптації до змінних ринкових умов. Кожен модуль відповідальний за конкретний аспект функціональності (наприклад, аналіз часових рядів, розрахунок ризиків, інтерфейс користувача тощо), та може бути модифікований або замінений незалежно від інших модулів. Така модульна структура сприяє ефективній інтеграції нових алгоритмічних рішень та технологій без необхідності глобальної перебудови всієї системи.

**Висновок до розділу**

В першому розділі роботи вивчено та проаналізовано наукову літературу та існуючі методики в області вибору і оптимізації інвестиційних портфелів. Цей етап став важливим для отримання комплексного розуміння динаміки фінансових ринків, особливостей різних класів активів, а також методів їх аналізу та вибору. Дослідження охоплює широкий спектр підходів — від класичних моделей, які визначили галузь на початкових етапах її розвитку, до сучасних інноваційних рішень, що використовують машинне навчання та інші методи аналізу даних.

Зокрема, було встановлено, що хоча модель Марковіца і залишається важливою в теорії портфельних інвестицій, її практичне використання обмежується рядом факторів, зокрема, нездатністю адекватно реагувати на швидкі зміни на ринку. Схожі обмеження спостерігаються в інших моделях та методиках, включаючи методику Кархарта та Білдом.

Це спонукало до дослідження сучасних технологічних платформ для управління інвестиційними портфелями, таких як Bloomberg Terminal, Quantopian та Wealthfront. Ці платформи використовують розширений набір алгоритмів для аналізу ринкових даних, але ні одна з них не забезпечує комплексного, адаптивного підходу до управління портфелем в динамічних ринкових умовах.

В результаті огляду була сформульована постановка задачі, яка спрямована на розробку системи для оптимізації вибору оптимальних інвестиційних портфелів з використанням сучасних технологій. Розроблена система планується до використання сучасних технологій, зокрема мови програмування C# та системи управління базами даних MS SQL Server, і буде мати модульну архітектуру для швидкої адаптації до змінних ринкових умов.

Отже, дана робота не лише агрегує існуючі знання в даній області, але і підкреслює нагальну потребу в розробці новітніх методів та технологічних рішень для оптимізації інвестиційних портфелів в контексті сучасних викликів та можливостей.

# 2 РОЗРОБКА ТА ОПТИМІЗАЦІЯ МЕТОДУ ВИБОРУ ІНВЕСТИЦІЙНИХ ПОРТФЕЛІВ

## 2.1 Встановлення критеріїв для оцінювання інвестиційних стратегій та їх оптимізація

При виборі оптимальних інвестиційних портфелів інструментарій аналітичного оцінювання включає в себе ряд критеріїв, що розглядають такі аспекти як дохідність, рівень ризику, кореляційні взаємозв'язки між активами та агреговані показники доходності портфелю. Ці критерії не лише являють собою ключові показники ефективності інвестицій, але й допомагають у формуванні когерентної стратегії розподілу капіталу.

Методологія аналізу часових рядів підтримує динамічний підхід до визначення тенденцій та взаємозв'язків між різними категоріями інвестиційних активів. Цей аналіз базується на історичних даних про ціни акцій, облігацій та інших фінансових інструментів. Інтеграція аналізу часових рядів у методику оцінки інвестиційних портфелів дозволяє не просто визначити статичні характеристики активів, але і спрогнозувати їх майбутню поведінку.

Зокрема, аналіз часових рядів відкриває можливість визначення кореляційних залежностей між активами, що дозволяє зменшити загальний рівень ризику портфелю через диверсифікацію. Також, цей метод допомагає оцінити ефективність вибраної інвестиційної стратегії, ідентифікувати потенційні ризики та можливості, а також скоригувати портфель у відповідності до змін на фінансових ринках.

У рамках розробки комплексної системи для вибору оптимальних інвестиційних портфелів із використанням сучасних технологій, ключова увага приділяється трем основним критеріям, які слугують показниками для аналізу та оцінки ефективності інвестиційних стратегій. Врахування цих критеріїв забезпечує ґрунтовний аналіз, що в свою чергу дозволяє максимізувати дохід,

контролювати ризики та забезпечити достатній рівень ліквідності портфеля. Основні критерії охоплюють:

– критерій ризику. Статистичні та аналітичні методи, як середній квадратичний ризик та Value at Risk, допомагають у квантитативній оцінці можливих фінансових ризиків, пов'язаних із конкретними інвестиційними інструментами;

– критерій рентабельності. Інструменти оцінки рентабельності, такі як середня дохідність чи кошторисний прибуток, дають змогу прогнозувати потенційну дохідність інвестиційних активів;

– критерій ліквідності: Фактори, як обсяг торгів на ринку або темпоральні характеристики купівлі-продажу, забезпечують інформаційну базу для оцінки ліквідності кожного інвестиційного інструменту.

Першим критерієм є дохідність, яка визначається як відношення прибутку до інвестиції. Для визначення дохідності буде використовуватись метод розрахунку середньої доходності, метод стандартного відхилення та інших показників, що вказують на величину та стабільність доходності.

Аналіз середньої дохідності є важливим для оцінки стабільності та прогнозованої ефективності інвестиційного портфеля. Формула для визначення середньої дохідності представлена у вигляді:

$$R_{avg} = \sum_{i=1}^{N}(R_i/N) \cdot 100, \quad (2.1)$$

де, $R_{avg}$– репрезентує середню дохідність портфеля;

$R_i$ – позначає $i$-ий окремий прибуток, отриманий від конкретного інвестиційного активу;

$N$ – вказує на кількість інвестиційних активів в портфелі.

Дана формула інтегрується у більш широкий алгоритм оптимізації портфеля, що може включати в себе не тільки класичні статистичні методи, але й сучасні комп'ютерні технології, такі як машинне навчання та штучний інтелект, для аналізу ринкових тенденцій та адаптації до них.

Для кількісної оцінки ризику використовується формула розрахунку стандартного відхилення. Математично, стандартне відхилення (σ) розраховується на основі виразу:

$$\sigma = \sqrt{\sum_{i=1}^{N}\left(R_i - R_{avg}\right)^2/(N-1)} \quad (2.2)$$

де $\sigma$ – стандартне відхилення, що слугує як метрика варіабельності або ризику, асоційованого з портфелем інвестицій;

$R_i$ – позначає $i$-ий показник доходності з індивідуального інвестиційного інструменту;

$R_{avg}$ – середня доходність, вирахувана на основі агрегованих даних з усіх взятих до аналізу інвестицій;

$N$ – загальна кількість інвестиційних інструментів у портфелі.

Важливість розрахунку коефіцієнта варіації (CV) полягає у тому, що цей показник дозволяє інвесторам провести нормовану оцінку ризику портфеля. Це стає можливим завдяки формулі:

$$CV = \left(\frac{\sigma}{R_{avg}}\right) * 100, \quad (2.3)$$

$CV$ – нормований показник, що має на меті вимірювання волатильності доходності портфеля;

$\sigma$ – стандартне відхилення, яке слугує підсумковою мірою ризику, базуючись на дисперсії доходності;

$R_{avg}$ – середньостатистична доходність, розрахована на основі агрегованих показників доходності активів, що входять до складу портфеля.

Застосування даного коефіцієнта відкриває можливості для якісного аналізу ризику через порівняння із іншими інвестиційними стратегіями або ринковими показниками. Так, інвестори зможуть не лише кількісно оцінити ризик, але й зробити його аналіз більш науково обґрунтованим.

Ризик являє собою ключовий параметр, що вимагає кількісної оцінки при формуванні інвестиційного портфеля. У цьому контексті пристосовуються засоби математичної статистики, зокрема такі індикатори як VaR та Expected Shortfall

(ES), що спрямовані на кількісну оцінку максимальних можливих втрат капіталу протягом дефінованого періоду із заданою ступіню вірогідності.

VaR представляє інструментарій, розроблений з метою оцінки амплітуди максимально допустимого відхилення вартості активів портфеля за визначений період і при певному рівні довіри. Вираз для обчислення VaR визначається наступним чином:

$$VaR = R_{avg} - (Z * \sigma), \quad (2.4)$$

де $VaR$ – показник максимальної можливої втрати з заданою ймовірністю;

$R_{avg}$ – арифметична середня доходностей активів портфеля, яка відображає експектовану норму прибутку;

$Z$ – шкала коефіцієнту зворотнього квантилю нормального розподілу, який відображає рівень довіри;

$\sigma$ – стандартне відхилення доходності портфеля, що слугує показником ризику.

Expected Shortfall (ES) представляє собою просунутий показник для оцінки ризику в контексті інвестиційного портфеля. По відношенню до Value-at-Risk (VaR), ES надає більш глибокий аналітичний зріз, оскільки концентрується не тільки на максимально можливій втраті з певним рівнем довіри, але і на характеристиці розподілу втрат, які можуть настати у випадку перевищення цього рівня.

Формула для розрахунку ES має наступний вигляд:

$$ES = R_{avg} - (Z * \sigma * \omega), \quad (2.5)$$

де $ES$ – показник, що характеризує очікувану втрату активів портфеля, яка може настати при перевищенні заданого порогового рівня (напр., 5% або 1%);

$R_{avg}$ – арифметична середня доходностей активів портфеля;

$Z$ – шкала коефіцієнта зворотного квантилю нормального розподілу;

$\sigma$ – стандартне відхилення доходності портфеля;

$\omega$ – ймовірність перевищення заданого порогового рівня втрат.

Основною перевагою Expected Shortfall є його здатність агрегувати інформацію не тільки про ймовірність настання втрат, але і про їх можливу амплітуду, що надає інвесторам більш комплексний інструмент для оцінки ризиків. Таким чином, ES є ефективним для використання у стратегічному управлінні ризиками інвестиційного портфеля.

В аналітичному портфелі управління інвестиціями поняття кореляції має критичне значення, зокрема для оптимізації ризик-дохідності. Кореляція як статистичний показник відображає ступінь взаємозалежності між фінансовими активами в інвестиційному портфелі. Осмислене використання кореляції дозволяє конструювати портфелі, що ефективно диверсифікують ризик.

Координоване добору активів з різними кореляційними характеристиками забезпечує можливість зниження загального рівня ризику портфеля. Це ґрунтується на принципі, що коливання вартості одного активу можуть бути нейтралізовані антагоністичними коливаннями іншого.

Математичний апарат для розрахунку кореляції між двома активами виражається наступною формулою:

$$\text{Correlation} = \frac{\text{Covariance}}{(\sigma_1 * \sigma_2)}, \tag{2.6}$$

де *Correlation* – коефіцієнт кореляції, що може вар'юватися від -1 до 1;

*Covariance* – коваріація, яка вимірює схильність двох активів змінюватися разом;

$\sigma_1$, $\sigma_2$ –відхилення першого та другого активів відповідно.

Оцінка кореляції повинна виконуватися з урахуванням динаміки ринкових умов та специфіки активів.

Також важливим завданням є формування оптимального портфелю активів, який би забезпечував найвищу доходність при заданому рівні ризику, або найменший ризик при фіксованому рівні доходності. Для цього вводиться ряд математичних показників та формул, серед яких важливе місце займає розрахунок ваг активів в портфелі. Вираз для визначення ваги і-го активу в оптимальному портфелі представлений наступним чином:

$$w_i = (R_i - R_f)/(\sigma_i * \lambda)\,, \qquad (2.7)$$

де $w_i$ – доля $i$-го активу в портфелі;

$R_i$ – очікувана дохідність даного активу;

$R_f$ – безризикова ставка, що служить відправною точкою для оцінки доходності ризикованих інвестицій;

$\sigma_i$ – стандартне відхилення доходності активу, як індикатор ризику;

λ – параметр, що відображає ступінь аверсії до ризику з боку інвестора.

Постійний моніторинг та оцінка параметрів інвестиційного портфелю відіграють критичну роль у забезпеченні ефективності інвестиційної стратегії. Це завдання включає в себе систематичний аналіз доходності та ризику, що вимагає застосування математичних моделей і статистичних методів. Специфічні алгоритми та інструменти, такі як фінансові похідні (опціони, ф'ючерси тощо) або методики диверсифікації, можуть бути застосовані для мінімізації ймовірності виникнення ризиків.

Оцінка дохідності та ризику конкретного портфелю має бути виконана на основі ретроспективного аналізу часових рядів фінансових показників, а також проекції цих показників на майбутнє. Проте, важливо розуміти, що попередні показники не гарантують майбутніх результатів, а лише служать базою для науково обґрунтованих припущень.

По завершенні цієї етапної оцінки, ключовим є внесення корективів у структуру портфелю з метою оптимізації його параметрів. Цей процес має бути динамічним і адаптивним, з огляду на швидкозмінний характер фінансових ринків.

Таким чином, створення та управління оптимальним інвестиційним портфелем — це ітераційний процес, який включає в себе першочерговий вибір активів на основі квантитативних показників, таких як доходність та ризик, а також подальший моніторинг та коригування цих параметрів для адаптації до змінюваних ринкових умов.

## 2.2 Створення та оптимізація математичної моделі

У цьому підрозділі представлена математична модель для ідентифікації оптимальних інвестиційних стратегій на фоні методології аналізу часових рядів. Схема передбачає оцінювання показників, таких як дохідність, волатильність та взаємозв'язок для кожного типу активу. За допомогою цих показників відбувається селекція найефективнішого портфелю. По завершенню формування портфеля проводяться подальші розрахунки його кумулятивної дохідності та ризикового профілю, які служать основою для кінцевого вибору оптимального портфеля.

Розглянемо $S_{i,j}$ як вартість $j$-го активу на $i$-й день. Змінна $j$ має діапазон значень від 1 до $n$, де $n$ є загальною кількістю інвестиційних активів у композиції портфеля. Змінна $i$ відображає число днів.

З метою оцінки кожного активу $j$, проводимо аналіз поточних ризикових параметрів та доходності. Позначимо r$i,j$ як дохідність $j$-го активу. Тоді середню доходність μ$j$ та стандартне відхилення s$j$ можна визначити за допомогою наступних математичних виразів:

$$\mu_j = \frac{1}{T}\sum_{i=1}^{T} r_{i,j} \tag{2.8}$$

$$s_j = \sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(r_{i,j} - \mu_j)^2} \tag{2.9}$$

Після виконання вищезгаданих розрахунків, процедура включає в себе визначення кореляції між двома активами, $j$ та $k$. Для цього використовується наступний математичний вираз:

$$corr_{j,k} = \frac{\sum_{i=1}^{T}(r_{i,j}-\mu_j)(r_{i,k}-\mu_k)}{\sqrt{\sum_{i=1}^{T}(r_{i,j}-\mu_j)^2}\sqrt{\sum_{i=1}^{T}(r_{i,k}-\mu_k)^2}} \tag{2.10}$$

У подальшому етапі розглядаються всі можливі варіанти злиття активів із заданими вагами для формування різноманітних інвестиційних портфелів. Для

кожного такого портфелю проводиться визначення його середньої дохідності та асоційованого ризику. Математичні вирази для цих метрик представлені нижче:

$$\mu_p = \sum_{j=1}^{n} w_j \mu_j \tag{2.10}$$

$$s_p = \sqrt{\sum_{j=1}^{n} \sum_{k-1}^{n} w_j w_k corr_{j,k} s_j s_k} \tag{2.11}$$

Тут $w_j$ - позначає вагу *j*-го у розглянутому портфелі.

Для відбору оптимального інвестиційного портфеля існує потреба в максимізації відношення доходності до асоційованого ризику. У цьому контексті ключовим показником служить коефіцієнт Шарпа, який формулюється наступним чином:

$$S = \frac{r_p - r_f}{\sigma_p} \tag{2.12}$$

де $r_p$ - проекція дохідності портфеля, $r_f$ - безризикова ставка відсотка, $\sigma_p$ - стандартне відхилення дохідності портфелю.

Ця метрика відображає рівень дохідності, досягнутий портфелем, у відношенні до понесених ризиків. Вищий коефіцієнт Шарпа індикаторний для портфелів із більш вигідним співвідношенням дохдності та ризику, та навпаки.

Також актуальним є введення поняття «ефективності портфеля», що характеризується як максимальне значення коефіцієнту Пірсона серед усіх можливих портфелів. Тобто, ефективні портфелі прагнуть до максимізації доходності при фіксованому, заданому рівні ризику.

В наступній частині аналізу здійснюється визначення коефіцієнтів кореляції між різними активами на основі попередньо розрахованих показників доходності та ризику. Кореляційний коефіцієнт між двома змінними, зокрема X та Y обчислюється як:

$$r(X,Y) = \frac{Cov(X,Y)}{Sd(X) * Sd(y)} \tag{2.13}$$

де *Cov(X,Y)* - є коваріаційною функцією між *X* та *Y*.

Коваріація між *X* та *Y* обчислюється як:

$$\mathrm{Cov}(\mathrm{X},\mathrm{Y}) = E[(X - X(X)(Y - E(Y)))], \quad (2.14)$$

де $E$ - математичне сподівання.

Стандартне відхилення для змінної $X$ та $Y$. обчислюється як:

$$Sd(x) = \sqrt{E[(X - E(X))^2]} \quad (2.15)$$

Коефіцієнт кореляції може варіюватися в межах від -1 до 1. Якщо значення даного коефіцієнта дорівнює -1, існує повна інверсна залежність між змінними; при значенні 0 — взаємозв'язок відсутній; при значенні 1 — існує повна пряма залежність між змінними.

Цей показник є важливим для оцінки спільного руху цін активів у портфелі та вибору оптимальної стратегії диверсифікації ризиків.

Субсеквентний етап аналізу передбачає відбір оптимального інвестиційного портфелю. В рамках цього процесу проводиться зіставлення прогнозованих доходностей та ризикових показників для кожного активу, а також оцінюється міра кореляції між ними. В сценарії, де доходність конкретного активу виявляється вищою за інші та характеризується нижчим рівнем ризику, цей актив рекомендується для інклюзії у портфель.

У ситуації, коли доходність активів схожа, але існує високий рівень кореляції між ними, активи інтегруються у портфель у пропорційних частках. Для більш точного відображення цього механізму нехай маємо N інвестиційних активів з прогнозованими доходностями $R_1$, $R_2$, ..., $R_n$ та відповідними ризиками $\sigma_1$, $\sigma_2$, ..., $\sigma_n$. Ваги активів у портфелі позначимо як $W_1$, $W_2$, ..., $W_n$, де сума ваг є рівною одиниці. Тоді прогнозована дохідність портфелю формулюється як:

$$R_p = W_1 * R_1 + W_2 * R_2 + \cdots + W_n * R_n \quad (2.16)$$

А для оцінки ризику портфелю використовується дисперсія дохідності, яка виражена як:

$$\sigma_p^2 = W_1^2 * \sigma_1^2 + W_2^2 * \sigma_2^2 + \cdots + W_n^2 * \sigma_n^2 + 2 * W_1 * W_2 * p_1^2 * \sigma_1 * \sigma_2 + 2 * W_1 * W_3 * p_1^3 * \sigma_1 * \sigma_3 + \cdots + 2 * W_{n-1} * W_n * p_{n-1,n} * \sigma_{n-1} * \sigma_n, \quad (2.17)$$

де $\rho_{i,j}$ - коефіцієнт кореляції між і-м та j-м активами.

У підсумку, активи, що вирізняються вищою дохідністю та меншим ризиком, рекомендуються для інтеграції у портфель. У випадку схожої дохідності, але високої кореляції, активи включаються в портфель у збалансованих пропорціях.

Після формування компонентного складу інвестиційного портфеля, другорядним завданням є визначення його сукупної ефективності та ступеню ризикованості. Цей процес передбачає використання математичних виразів, що базуються на індикаторах доходності та ризику окремих активів портфеля, а також на їх вагових коефіцієнтах у загальному портфелі.

Загальна оцінка ефективності портфеля враховує ваги окремих активів та їх очікувану доходність, що може бути розрахована за допомогою формули:

$$E(R_p) = W_1 * E(R_1) + W_2 * E(R_2) + \cdots + W_n * E(R_n), \quad (2.18)$$

де $E(R_p)$ - загальна очікувана доходність портфеля;

$E(R_1)$ - очікувана доходність *i*-го активу;

$W_1$ - його вага у портфелі;

n - кількість активів.

Ступінь ризику портфеля може бути визначений через розрахунок дисперсії його доходності, що розраховується наступним чином:

$$\sigma_p = \sqrt{W_1^2 * \sigma_1^2 + W_2^2 * \sigma_2^2 + \cdots + W_n^2 * \sigma_n^2 + 2W_1 * W_2 * \sigma_{1,2} + 2 * W_1 * W_3 * \sigma_{1,3}} + \sqrt{2 * W_{n-1} * W_n * p_{n-1,n} * \sigma_{n-1} * \sigma_n}, \quad (2.19)$$

де, $\rho_{i,j}$ - кореляційний коефіцієнт між *i*-м та *j*-м активами.

Якщо один актив виявляється більш ефективним за інші з мінімальним рівнем ризику, його вага в портфелі може бути підвищена. Якщо ж окремі активи характеризуються схожими показниками доходності, але мають високу взаємну

кореляцію, то їх вагові коефіцієнти в портфелі зазвичай робляться порівнюваними.

Після того, як компоненти інвестиційного портфеля визначені, наступна стадія це розрахунок його потенційної прибутковості та асоційованих ризиків. Для цього застосовують математичні рівняння, що враховують прибутковість та нестабільність кожного активу у портфелі, а також їх пропорційну частку в портфелі.

Для оцінки потенційної прибутковості портфеля, можна використовувати наступний вираз:

$$E(R_p) = W_1 * E(R_1) + W_2 * E(R_2) + \cdots + W_n * E(R_n), \quad (2.20)$$

де $E(R_p)$ - це проекція прибутковості всього портфелю;

$E(R_1)$ - проекція прибутковості *i*-го активу;

$W_1$ - частка цього активу в загальній вартості портфелю;

n - кількість активів.

Для оцінки ризику, або стандартного відхилення прибутковості портфеля, можна використовувати формулу:

$$\sigma p = \sqrt{W_1^2 * \sigma_1^2 + W_2^2 * \sigma_2^2 + \cdots + W_n^2 * \sigma_n^2} + \sqrt{2 * W_1 * W_2 * \sigma_{1,2} + 2 * W_1 * W_3 * \sigma_{1,3} + \cdots + 2 * W_{n-1} * \sigma_{n-1}} \quad (2.20)$$

де σp - коваріація між прибутковістю *i*-го та *j*-го активів.

Одержані показники потенційної прибутковості та ризику можна використовувати для порівняння з аналогічними характеристиками інших можливих портфелів. Це дозволить вибрати портфель, який пропонує найкраще співвідношення між очікуваною прибутковістю та ризиком.

### 2.3 Проведення процедур валідації та верифікації розробленої моделі

В рамках системи для відбору оптимальних інвестиційних портфелів, процеси валідації та верифікації включають кілька ключових етапів:

– аналіз автентичності джерел інформації. Перед аналітичною обробкою даних необхідно здійснити перевірку їх походження з метою забезпечення надійності та точності;

– оцінка алгоритмічного апарату. Слід забезпечити коректність алгоритмів, що застосовуються для розрахунку таких параметрів як ризики, потенційні доходи, кореляції між активами тощо;

– валідація модельного апарату. Модель для вибору інвестиційних портфелів повинна підтвердити свою ефективність на основі реальних, історично акумульованих даних, щоб визначити ступінь її точності у прогнозуванні;

– перевірка результативності. Необхідно аналізувати, наскільки виводи системи корелюють з реальними показниками ризику та доходності;

– тестова експлуатація. Перед введенням системи в комерційну експлуатацію слід провести тестування з метою переконання в її надійності та відсутності технічних недоліків;

– контроль роботи системи. Після запуску системи важливо забезпечити її неперервний моніторинг для того, щоб своєчасно ідентифікувати та усунути можливі дефіцити чи помилки в її функціонуванні.

У сфері вибору оптимальних інвестиційних портфелів важливість валідації та верифікації не можна переоцінити. Рішення, прийняті на основі аналізу системи, мають прямий вплив на економічну ефективність. Якщо система налаштована або калібрована неправильно, це може призвести до суттєвих фінансових втрат. Навпаки, точний і надійний аналіз може забезпечити високий рівень прибутковості.

Отже, перед реалізацією реальних інвестиційних стратегій за допомогою цієї системи, важливо забезпечити її точність і надійність. Це дозволить мінімізувати ризики та забезпечити оптимальну ефективність.

Валідація та верифікація були проведені з метою переконання в коректності функціонування системи та точності її результатів, що є критично важливим для зменшення ризиків та підвищення фінансової ефективності.

**Висновок до розділу**

В рамках даного розділу була проведена підготовча робота для розробки системи вибору оптимальних інвестиційних портфелів.

Визначено ключові критерії для оцінки інвестиційних портфелів, а саме: ризик, рентабельність, ліквідність, та середня доходність. Ці критерії викладені як фундаментальні параметри для майбутньої математичної моделі.

Проаналізовано можливість використання методу часових рядів та створено математичну модель, що буде здатна прогнозувати ринкові тренди та вибирати потенційно прибуткові активи.

Підготовлено концептуальні основи для реалізації математичної моделі в наступних розділах, що включатиме оптимізацію вибору інвестиційних портфелів.

Розроблені принципи та підходи в даному розділі можуть стати основою для подальшого наукового дослідження в цій області, та їх практичного впровадження після детальної розробки системи.

# 3 ВИЗНАЧЕННЯ ТЕХНОЛОГІЧНИХ АСПЕКТІВ ТА АРХІТЕКТУРНИХ РІШЕНЬ ПРОГРАМНОГО ПРОДУКТУ

## 3.1 Оцінка та вибір технологічних компонентів

*Мова програмування*

Вибір мови програмування є важливим етапом перед безпосередньою розробкою програмного забезпечення, особливо в контексті наукових досліджень, де кожен аспект від інструментарію до виконання має науково-обґрунтовану значимість. Цей вибір завдяки його стратегічній важливості вимагає всебічного аналізу ряду потенційно придатних мов програмування. Оцінка здійснюється за критеріями, які включають обчислювальну ефективність, доступність спеціалізованих бібліотек, зручність синтаксису для конкретних завдань, та інтегрованість з іншими технологічними рішеннями.

Мова програмування C# є об'єктно-орієнтованою мовою, розробленою компанією Microsoft в рамках платформи .NET. Ця мова комбінує в собі виразність синтаксису з потужними можливостями для розробки як консольних, так і графічних користувацьких додатків. Однією з ключових переваг C# є його гнучкість та масштабованість, що дозволяє використовувати мову в різних сферах: від розробки веб-сервісів і мобільних додатків до наукових досліджень і систем штучного інтелекту. Завдяки вбудованим засобам для роботи з мережами, базами даних і мультимедійними технологіями, а також широкому спектру доступних бібліотек і фреймворків, C# стає оптимальним вибором для комплексних і високонавантажених систем. Ця мова також характеризується високим рівнем безпеки типів, автоматичним керуванням пам'яттю та сучасними можливостями для паралельного програмування [19].

Мова програмування C++ є однією з найбільш широко використовуваних мов у сферах високопродуктивних обчислень, системного програмування, а також наукових досліджень. Ця мова є розвитком мови C з додаванням об'єктно-

орієнтованих конструкцій, а також шаблонів, які забезпечують більш виразні та гнучкі засоби абстракції. Основними перевагами C++ є його висока продуктивність, забезпечена низькорівневим доступом до комп'ютерних ресурсів, та широкий спектр бібліотек для наукових розрахунків, роботи з графікою, мережами та іншими сферами.

До особливостей C++ належить можливість детального керування пам'яттю та ресурсами, що є критично важливим у високонавантажених системах та обчислювально інтенсивних задачах. Також не можна ігнорувати велику кількість наявних фреймворків і бібліотек, які значно прискорюють процес розробки.

В контексті оптимізації вибору інвестиційних портфелів, C++ може бути використаний для реалізації високошвидкісних моделей та алгоритмів, де ефективність виконання є пріоритетним фактором. Однак, у порівнянні з C#, мова вимагає більш детального керування ресурсами і може бути менш зручною для реалізації високорівневих абстракцій [20-21].

Мова програмування Python відзначається своєю виразністю, читабельністю коду та широким спектром застосувань, що включає веб-розробку, аналіз даних, штучний інтелект, наукові дослідження та багато інших сфер. Однією з ключових переваг Python є богатий екосистемний ландшафт, що включає величезний арсенал бібліотек та фреймворків, спеціалізованих під конкретні задачі, включаючи фінансову аналітику та оптимізацію портфелів.

Python також відомий своєю гнучкістю, яка дозволяє легко інтегрувати його з іншими мовами програмування та системами. Це особливо корисно при роботі з великими об'ємами даних або високопродуктивними обчисленнями, де Python може використовуватися для прототипування, а більш ефективні мови, такі як C++ або C#, для фінальної реалізації.

У контексті оптимізації вибору інвестиційних портфелів, Python пропонує низку високорівневих бібліотек для математичних розрахунків, статистичного аналізу та машинного навчання, таких як NumPy, pandas, Scikit-learn, що може значно прискорити процес розробки. Проте, варто зазначити, що Python може

бути менш ефективним з точки зору виконання в порівнянні з компільованими мовами, такими як C++ або C# [22-23].

У табл. 3.1 проведено порівняння розглянутих мов програмування, що дозволить здійснити вибір найкращої.

Таблиця 3.1

Порівняльний аналіз мов програмування

| *Критерій* | *C#* | *C++* | *Python* |
|---|---|---|---|
| Обчислювальна ефективність | Висока | Висока | Помірна |
| Доступність спеціалізованих бібліотек | Висока | Висока | Дуже висока |
| Зручність синтаксису | Висока | Помірна | Висока |
| Безпека типів | Висока | Помірна | Помірна |
| Автоматичне керування пам'яттю | Є | Немає | Є |
| Підтримка паралельного програмування | Висока | Висока | Помірна |
| Вбудовані засоби для розробки UI | Висока (WinForms, WPF) | Помірна (Qt) | Помірна (Tkinter, PyQt) |
| Швидкість розробки | Висока | Помірна | Висока |
| Вбудовані засоби для роботи з мережами | Висока (WinForms, WPF) | Помірна (Qt) | Помірна (Tkinter, PyQt) |
| Доступність спеціалізованих бібліотек | Висока | Висока | Дуже висока |
| Зручність синтаксису | Висока | Помірна | Висока |

Виходячи з проведеного аналізу, можна стверджувати, що мова програмування C# представляє собою оптимальний вибір для розробки системи оптимізації вибору інвестиційних портфелів з використанням сучасних технологій. Декілька ключових аргументів в підтримку цього вибору висвітлено нижче:

– обчислювальна ефективність. C# демонструє високий рівень обчислювальної ефективності, який є критичним для фінансових розрахунків та оптимізації портфелів;

– інтегрованість з Microsoft технологіями. Оскільки ви вже використовуєте MS SQL Server, наявність глибокої інтеграції C# з цією та іншими технологіями Microsoft є великим плюсом;

– безпека типів. Сильна система типізації в C# забезпечує високий рівень надійності коду, що є важливим для фінансових систем, де помилка може призвести до серйозних наслідків;

– автоматичне керування пам'яттю. Збирач сміття в C# зменшує ризики, пов'язані з управлінням пам'яттю, що може бути корисним при роботі з великими об'ємами даних;

– підтримка паралельного програмування. Для задач, що вимагають високої пропускної здатності та обчислювальної міці, C# надає ефективні засоби реалізації паралельних та асинхронних операцій;

– швидкість розробки. Синтаксис C# та наявність зрілих бібліотек може значно прискорити процес розробки, що є важливим для комерційних проектів з обмеженими строками.

Враховуючи згадані переваги та вашу потребу в високопродуктивній, надійній та легко інтегрованій системі, C# видається найбільш доцільним варіантом для реалізації даного проекту.

Вибір правильної системи управління базами даних (СУБД) є критичним етапом в розробці будь-якої сучасної інформаційної системи. Це поняття відноситься і для систем оптимізації вибору інвестиційних портфелів, де важливі критерії включають не лише швидкість роботи та масштабованість, але й безпеку,

надійність та можливість глибокого аналізу даних. Додатково, інтегрованість СУБД з обраною мовою програмування може значною мірою полегшити процес розробки та подальшої експлуатації системи.

MS SQL Server є комерційною СУБД, розробленою компанією Microsoft. Однією з основних її характеристик є висока надійність та масштабованість, що робить її підходящою для великих корпоративних застосунків та систем, що обробляють великі об'єми даних. MS SQL Server підтримує широкий спектр функцій, включаючи транзакційну обробку, бізнес-інтелектуальний аналіз (BI), а також розподілене та хмарне зберігання [24].

Також, ця СУБД має ряд вбудованих засобів для оптимізації запитів та управління ресурсами, що полегшує задачу розробників та адміністраторів баз даних. Не менш важливим є те, що MS SQL Server інтегрується бездоганно з іншими продуктами та технологіями Microsoft, включаючи мову програмування C# [25].

Oracle Database є однією з найбільш популярних комерційних Систем Управління Базами Даних (СУБД), що використовується для управління великими та складними наборами даних в корпоративному середовищі. Ця СУБД розроблена компанією Oracle Corporation і відзначається високим рівнем надійності, масштабованості та безпеки.

Однією з ключових особливостей Oracle Database є її підтримка розподіленого зберігання даних, що дозволяє ефективно управляти великими об'ємами інформації. До того ж, Oracle має вбудовані засоби для бізнес-аналізу, обробки транзакцій, шифрування даних, а також ряд інших функцій, які спрощують роботу з корпоративними базами даних.

Oracle також має широкий асортимент додаткових модулів та інструментів, які можуть бути інтегровані для забезпечення специфічних бізнес-вимог. Однак, варто врахувати, що ліцензійна політика та вартість власності (TCO) для Oracle можуть бути досить високими, особливо при великих розмірах даних та потребах в розширеній функціональності [26].

MS Access є СУБД, розробленою компанією Microsoft, і є частиною пакету Microsoft Office. Ця СУБД призначена, в основному, для малих та середніх застосунків і проектів, де основні акценти зроблені на швидкість розробки та простоту використання. MS Access дозволяє швидко створювати прототипи баз даних, використовуючи зручний графічний інтерфейс, а також реалізовувати прості системи управління даними без великих витрат часу та ресурсів.

Однак, MS Access має обмеження з питань масштабованості, продуктивності та безпеки, що робить його менш підходящим для великих корпоративних систем або застосунків, що обробляють великі об'єми даних. Він не рекомендований для задач, що вимагають високої обчислювальної ефективності, розподіленого зберігання даних або високого рівня безпеки.

Також варто зазначити, що MS Access оптимізований для інтеграції з іншими продуктами Microsoft Office, такими як Excel та Outlook, що може бути корисним для деяких типів застосунків, де потрібна тісна взаємодія з офісними інструментами [27].

Порівняння основних характеристик MS SQL Server, Oracle та MS Access можна побачити в табл. 3.2.

Таблиця 3.2

Порівняльний аналіз СУБД

| Параметр | MS SQL Server | Oracle Database | MS Access |
|---|---|---|---|
| Швидкість роботи | Висока | Висока | Середня/Низька |
| Масштабованість | Висока | Висока | Обмежена |
| Безпека | Висока | Висока | Середня |
| Надійність | Висока | Висока | Середня |
| Можливість глибокого аналізу | Високий рівень (BI інтеграція) | Високий рівень (BI опції) | Обмежена |

Продовження таблиці 3.2

Порівняльний аналіз СУБД

| Параметр | MS SQL Server | Oracle Database | MS Access |
|---|---|---|---|
| Ліцензійна політика | Ліцензія на користувача | Ліцензія на ядро або користувача | Входить у MS Office |
| Спеціалізовані бібліотеки | Широкий асортимент | Широкий асортимент | Обмежена кількість |
| Комплексність налаштування | Середнє/Високе | Високе | Низьке |
| Підтримка хмарних рішень | Вбудована (Azure) | Доступна | Обмежена |

На підставі проведеного аналізу можна зробити висновок, що MS SQL Server є вибором, який забезпечує високий рівень продуктивності, безпеки та надійності, а також надає додаткові переваги в інтеграції з іншими Microsoft продуктами. Це робить його ідеально підходящим для проекту з оптимізації інвестиційних портфелів, особливо якщо розробка ведеться на мові програмування C#.

### 3.2 Аналітичний огляд вимог: діаграми прецедентів та основні сценарії

Перед розробкою проекту з оптимізації вибору інвестиційних портфелів за допомогою сучасних технологій, первинним кроком є вивчення системних вимог. Цей етап ґрунтується на потребах потенційних користувачів та специфікаціях, які система повинна задовольняти. Одним з методологічних інструментів для цього є діаграма прецедентів або «use-case» діаграма.

Діаграма прецедентів представляє взаємодії між користувачами та системою, а також перераховує ключові функціональні аспекти, які система

повинна виконувати. Вона складається з акторів (що може бути користувачами або іншими системами) та сценаріїв взаємодії, які можна назвати прецедентами.

Етап аналізу вимог охоплює збір, класифікацію та документацію всіх вимог, які можуть виникнути в процесі реалізації проекту. Це включає як функціональні, так і нефункціональні вимоги, що відображають функціональні можливості та обмеження, відповідно.

Ці вимоги можуть бути структуровані у вигляді таблиці чи іншого виду документації. Табл. 3.3 містить перелік вимог, які повинна задовольняти система для оптимізації інвестиційних портфелів.

Таблиця 3.3

Функціональні вимоги до системи

| **Вимоги** | **Опис** |
|---|---|
| REQ-1 | Можливість авторизації та аутентифікації користувачів |
| REQ-2 | Ведення аудиту та журналів дій користувачів |
| REQ-3 | Можливість аналізу фінансових часових рядів |
| REQ-4 | Можливість створення, редагування, та оптимізації інвестиційних портфелів |
| REQ-5 | Інтеграція алгоритмів для аналізу часових рядів |
| REQ-6 | Можливість генерування аналітичних звітів |
| REQ-7 | Можливість введення та збереження метаданих про активи, портфелі, та релевантні параметри |
| REQ-8 | Можливість візуалізації результатів аналізу |

Нефункціональні вимоги охоплюють широкий спектр параметрів, таких як продуктивність, безпека, зручність використання та інші характеристики, які сприяють ефективності, масштабованості та надійності системи. Їх важливо визначити на ранніх етапах розробки та взяти до уваги при проектуванні архітектури системи (табл. 3.5).

Таблиця 3.4

Нефункціональні вимоги до системи

| **Вимоги** | **Опис** |
|---|---|
| REQ-9 | Швидкодія системи при великих об'ємах даних |
| REQ-10 | Висока надійність та доступність системи |
| REQ-11 | Масштабованість системи |
| REQ-12 | Захист даних і конфіденційності інформації |
| REQ-13 | Інтуїтивність та зручність користувацького інтерфейсу |

В системі необхідно звернути особливу увагу на роль та відповідальність кожного з учасників взаємодії з системою. Актори в системі — це не лише люди, які взаємодіють з нею, а й інші системи або компоненти. Їх цілі та потреби можуть суттєво відрізнятися, і зрозуміння цих ролей є критично важливим для успішного проектування та реалізації системи.

Цілі акторів в системі слугують визначальним чинником для встановлення вимог до функціональності та якості системи. Вони формують контекст, в якому буде функціонувати система, та її ключові метрики ефективності.

В табл. 3.5 представлено опис акторів, їх цілей та взаємодії з системою.

Таблиця 3.5

Актори та цілі системи

| **Актори** | **Цілі** |
|---|---|
| Адміністратор | Забезпечити надійність та безпеку управління обліковими даними в системі |
| Користувач | Використовувати систему максимально ефективно та комфортно |
| База даних | Надійне збереження та структурована організація потрібних даних |

В контексті системної інженерії та аналізу вимог, варіанти використання системи (Use Cases) займають центральне місце у визначенні функціональних

можливостей та обмежень системи. Ці варіанти використання відображають конкретні сценарії, які описують, як система взаємодіє з зовнішніми акторами для досягнення певних цілей. Кожен варіант використання складається з послідовності дій, виконаних системою, а також можливих відгуків з боку акторів.

Варіанти використання є важливим інструментом для забезпечення глибокого розуміння всіх аспектів системи: від її функціональних можливостей до нефункціональних характеристик, таких як надійність, безпека та продуктивність. Вони дають можливість визначити граничні умови системи, яскраво ілюструють взаємодії між компонентами та акторами, і, таким чином, слугують відмінною основою для проектування системи та її подальшої реалізації.

В табл. 3.6 представлено перелік варіантів використання системи, що допоможе в уточненні вимог до неї і встановленні ключових параметрів для етапів розробки, тестування та валідації. Кожен варіант використання буде розглянутий через призму його цілей, основних та альтернативних сценаріїв, а також через взаємодію з іншими варіантами використання та акторами системи.

Таблиця 3.6

Опис варіантів використання системи

| **Варіант використання** | **Ім’я** | **Опис** |
|---|---|---|
| UC1 | Створення облікового запису | Дозволяє ініціацію користувачем процесу реєстрації в системі |
| UC2 | Підтвердження особистості | Дозволяє користувачу проходження процедури автентифікації для доступу до системи |

Продовження таблиці 3.6

Опис варіантів використання системи

| **Варіант використання** | **Ім'я** | **Опис** |
|---|---|---|
| UC3 | Перегляд списку користувачів | Надає можливість адміністратору системи отримати перелік зареєстрованих користувачів |
| UC4 | Додати користувача | Дає можливість додавання нових користувачів до системи |
| UC5 | Редагувати користувача | Дозволяє проводити модифікацію та оновлення інформації користувача |
| UC6 | Вивід каталогу фірм/компаній | Дає можливість користувачеві ознайомитись зі всіма фірм/компаній у системі |
| UC7 | Додати фірму/компанію | Дозволяє розширити фірм/компаній |
| UC8 | Редагувати фірму/компанію | Дозволяє змінювати та оновлювати інформацію про обрану фірму або компанію |
| UC9 | Формування цін на акції | Дозволяє користувачеві додавати/ змінювати інформацію про ціни на акції |
| UC10 | Формування цін на облігації | Дозволяє користувачеві додавати/ змінювати інформацію про ціни на облігації |
| UC11 | Формування цін на нерухомість | Дозволяє користувачеві додавати/ змінювати інформацію про ціни на нерухомість |
| UC12 | Оптимізація портфелю | Дозволяє користувачеві системи обрати найкращі інвестиційні портфелі |
| UC13 | Перегляд подій | Дозволяє переглядати події системи |

На підставі зібраних даних були розроблені діаграми прецедентів use-case для ролей користувача та адміністратора, які ілюстровані на фігурах 3.1 та 3.2 відповідно.

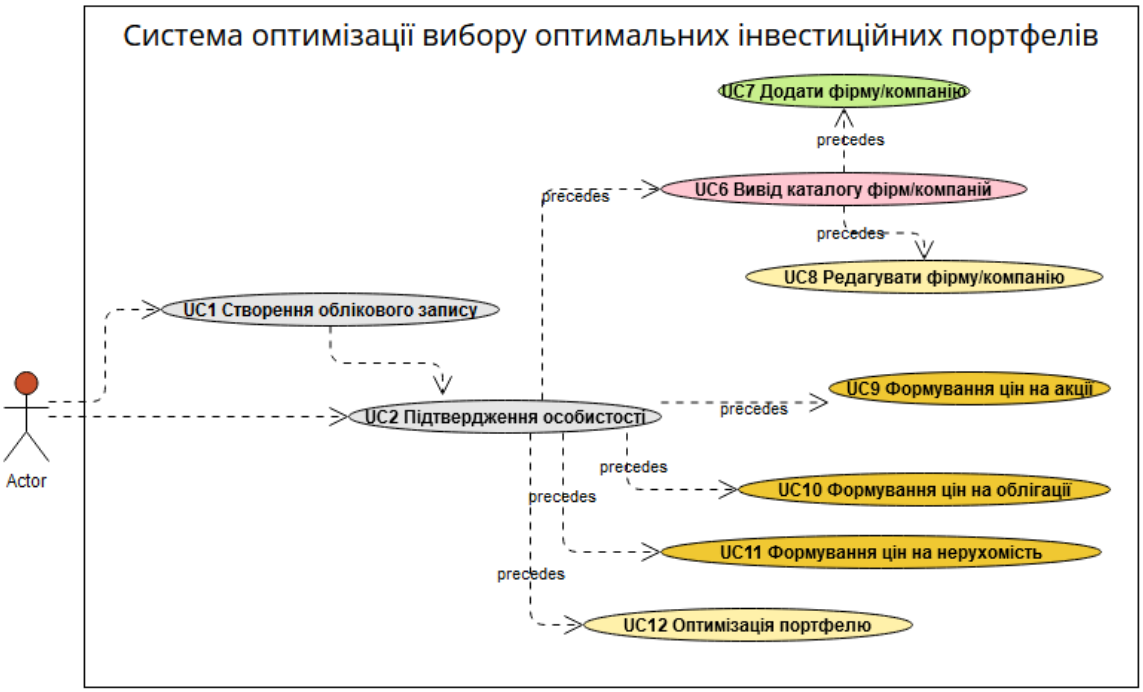



Рис 3.1 Діаграма use-case із роллю «користувач»


Рис 3.2 Діаграма use-case із роллю «адміністратор»

## 3.3 Специфікація розробки бази даних

У розробленій системі використовується реляційна база даних, яка включає в себе такі таблиці як:

– таблиця «Actions» призначена для збереження інформації про акції компаній. Кожний запис в таблиці містить інформацію про конкретну дію, її дату виконання, числові характеристики та посилання на відповідну компанію;

– таблиця «Bonds» фокусується на зберіганні даних про облігації, що випускаються компаніями. Записи включають інформацію про дату випуску, номінальну вартість та зв'язок з конкретною компанією;

– таблиця «Companys» служить для агрегації основних відомостей про компанії, включаючи їх назви та описи;

– таблиця «RealEstate» зберігає інформацію про активи в сфері нерухомості, які належать компаніям. Тут фіксуються дати транзакцій та оцінки вартості;

– таблиця «Users» концентрує дані про користувачів системи, включаючи їх імена, логіни, паролі та ролі в системі;

– таблиця «Logs» створена з метою зберігання журналів активності користувачів. Ця таблиця дає можливість відстежувати взаємодію користувачів з системою, їх дії, та може бути використана для аналізу та оптимізації роботи системи..

У розробленій реляційній базі даних для управління інвестиційними портфелями, взаємозв'язки між таблицями структуровані наступним чином:

– «Companys» (поле «CompanysId») – «Actions» (поле «CompanysId»): Вид зв'язку 1:N. Одна компанія може мати кілька різних акцій, але кожна акція асоційована лише з однією компанією;

– «Companys» (поле «CompanysId») – «Bonds» (поле «CompanysId»): Вид зв'язку 1:N. Одна компанія може випускати декілька видів облігацій, кожна з яких буде зв'язана з єдиною компанією;

– «Companys» (поле «CompanysId») – «RealEstate» (поле «CompanysId»): Вид зв'язку 1:N. Кожна компанія може володіти декількома об'єктами нерухомості, проте кожен об'єкт прив'язаний до однієї компанії;

– «Users» (поле «UsersId») – «Logs» (поле «UsersId»): Вид зв'язку 1:N. Для кожного користувача може бути зафіксована множина подій в журналі логів, але кожна подія є унікально прив'язана до одного користувача.

Всі ці зв'язки забезпечують інтегрованість та цілісність даних, що взаємодіють між різними сутностями бази. Вони допомагають в ефективному використанні засобів зберігання, організації та аналізу даних, що є важливим для досягнення цілей системи у контексті управління інвестиційними портфелями.

Для досягнення вищого рівня точності та структурованості даних в контексті оптимізації вибору інвестиційних портфелів з використанням сучасних технологій, будуть використовуватись вказані таблиці та відповідні їм атрибути:

Таблиця 3.7

Атрибути сутності «Actions»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | ActionsId | INT | Унікальний ідентифікатор дії пов'язаної з акціями компанії |
| 2 | ActionDate | DATETIME | Дата та час випуску акції |
| 3 | ActionValues | FLOAT | Цінність або вартість акцій на певний момент часу |
| 4 | CompanysId | INT | Ідентифікатор компанії, до якої відноситься дія |

Таблиця 3.8

Атрибути сутності «Logs»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | LogsId | INT | Унікальний ідентифікатор запису в журналі |
| 2 | UsersId | INT | Ідентифікатор користувача |
| 3 | EventNameShow | NVARCHAR(MAX) | Назва події |
| 4 | EvendDate | DATETIME | Дата та час події |
| 5 | UsersName | NVARCHAR(150) | Ім'я користувача, який ініціював подію |

Таблиця 3.9

Атрибути сутності «Bonds»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | BondsId | INT | Унікальний ідентифікатор облігації |
| 2 | BondsDate | DATETIME | Дата та час випуску облігації |
| 3 | BondsValues | FLOAT | Номінальна вартість облігації |
| 4 | CompanysId | INT | Ідентифікатор компанії, яка випустила облігацію |

Таблиця 3.10

Атрибути сутності « Users»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | UsersId | INT | Ідентифікатор користувача |
| 2 | FirstName | NVARCHAR(100) | Ім'я користувача |
| 3 | LastName | NVARCHAR(100) | Прізвище користувача |
| 4 | UsersName | NVARCHAR(100) | Логін користувача |
| 5 | UsersPassword | NVARCHAR(MAX) | Пароль користувача |

Продовження таблиці 3.10

Атрибути сутності « Users»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 6 | RoleId | INT | Ідентифікатор ролі користувача |
| 7 | Description | NVARCHAR(MAX) | Опис користувача |
| 8 | Email | NVARCHAR(MAX) | Електронна пошта користувача |
| 1 | UsersId | INT | Унікальний ідентифікатор користувача |

Таблиця 3.11

Атрибути сутності « RealEstate»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | RealEstateId | INT | Ідентифікатор нерухомості |
| 2 | RealEstateDate | DATETIME | Дата та час придбання нерухомості |
| 3 | RealEstateValues | FLOAT | Вартість нерухомості |
| 4 | CompanysId | INT | Ідентифікатор компанії, яка є власником нерухомості |

Таблиця 3.12

Атрибути сутності «Companys»

| №п/п | Найменування | Тип даних | Призначення |
|---|---|---|---|
| 1 | CompanysId | INT | Ідентифікатор компанії |
| 2 | CompanysName | NVARCHAR(250) | Назва компанії |
| 3 | Description | NVARCHAR(MAX) | Опис діяльності компанії |

Ці таблиці формують спільну інформаційну модель, що спрямована на ефективне зберігання, організацію та аналіз даних, що обробляються в рамках системи.

Фізична модель бази даних є завершальним етапом в процесі моделювання бази даних і включає в себе деталізацію структур даних, оптимізацію запитів, індексацію, а також визначення засобів збереження.

Для ефективної реалізації фізичної моделі бази даних важливою є етап кодування структур даних та обмежень цілісності. В рамках даної роботи, цей етап включає в себе розробку SQL-скриптів, що відповідають за створення таблиць, встановлення зв'язків між ними, а також формування індексів та тригерів. Ці скрипти деталізовано представлені у додатку А даної роботи.

На рис. 3.3 фізичну модель БД представлено у графічному вигляді.

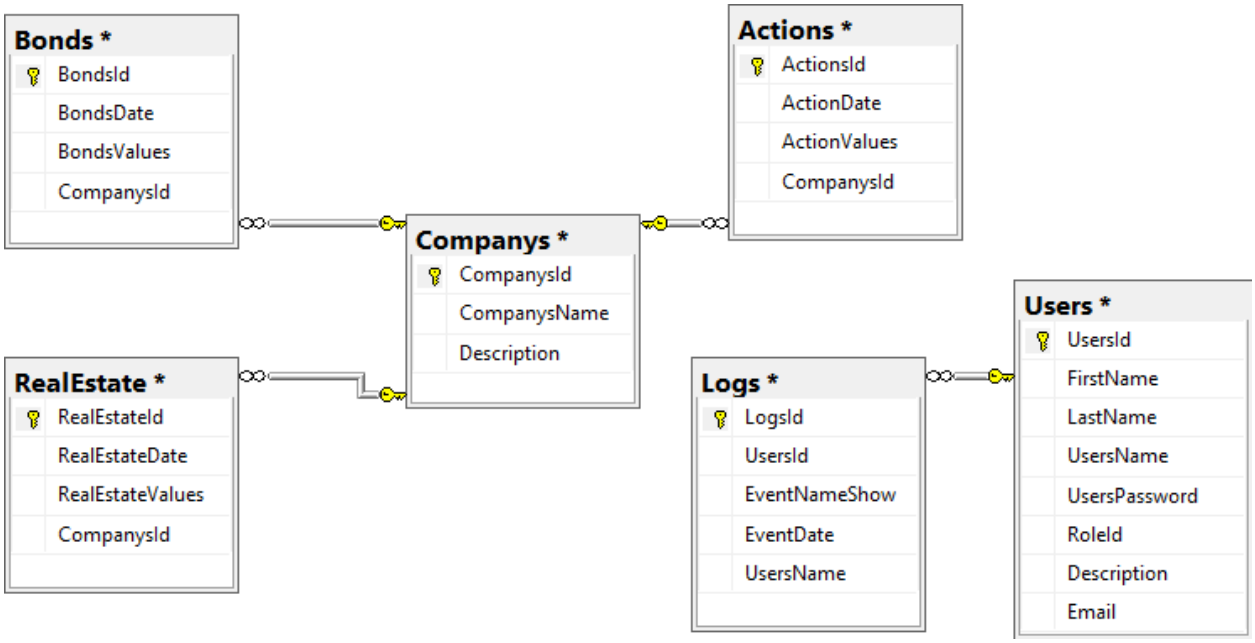



Рис 3.3 Фізична модель бази даних

### 3.4 Проектування архітектури програмного забезпечення

У рамках розробки системи вибрана модель тришарової архітектури, яка широко застосовується у сфері створення програмних рішень. Ця конструктивна схема організовує програмне середовище на три основні компоненти: візуальний інтерфейс (шар представлення), обробку бізнес-процесів (шар бізнес-логіки) та управління даними (шар доступу до даних).

Застосування тришарової архітектури сприяє ефективній декомпозиції функціоналу системи, що полегшує її масштабування та подальший розвиток. Специфічна конструкція дозволяє реалізувати інтерфейс користувача на основі веб-технологій, в той час як управління даними може бути ефективно виконане через реляційну базу даних. Це також полегшує інтеграційні процеси з іншими програмними системами. Одним із додаткових переваг є забезпечення інформаційної безпеки, оскільки доступ до бази даних регулюється на відділеному шарі.

### 3.4.1 Рівень бізнес-логіки

Шар бізнес-логіки (Business Logic Layer, або BLL) в системі слугує для реалізації бізнес-правил та оперативної обробки даних [28]. Цей компонент виконує наступні завдання:

– модифікація та адаптація даних отриманих з шару управління даними (Data Access Layer, або DAL), перед їхнім відображенням у користувацькому інтерфейсі (Presentation Layer, або PL); [29]

– аутентифікація та валідація даних, що подаються користувачем, перед їхнім зберіганням у даних;

– встановлення та моніторинг дотримання бізнес-правил у ході виконання операцій;

– обмін даними та результатами обробки з користувацьким інтерфейсом (PL).

Переваги використання BLL можна охарактеризувати наступним чином:

– гарантія цілісності та безпеки інформації;

– відсутність залежності бізнес-логіки від користувацького інтерфейсу;

– гнучкість у зміні джерел даних без перебудови бізнес-процесів;

– мінімізація дублювання коду і можливість його повторного використання.

Діаграму класів, що відображає структуру шару бізнес-логіки, можна переглянути на рис. 3.4.

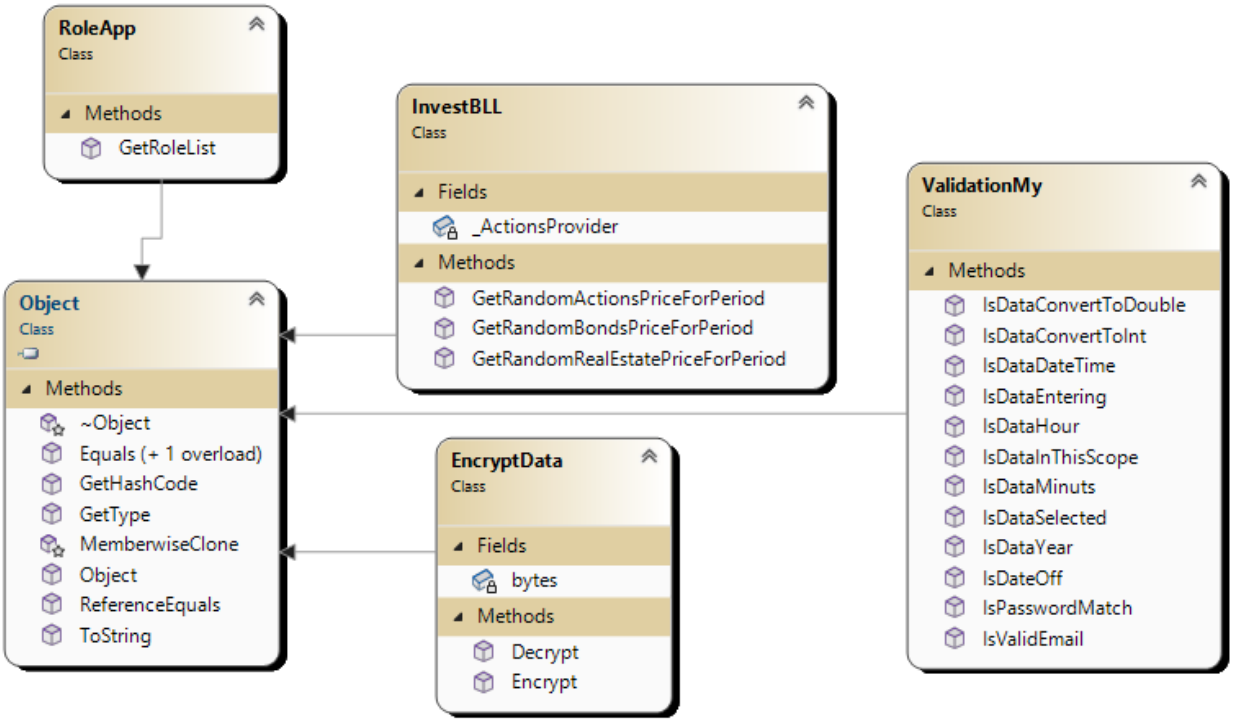



Рис 3.4 Діаграма класів бізнес-логіки системи

Діаграма рівня бізнес-логіки включає в себе наступні ключові класи:

– клас InvestBLL служить для управління інвестиційними портфелями. Він включає методи для розрахунку прибутковості, ризику, а також для оптимізації портфеля на основі математичної моделі;

– клас EncryptData відповідає за шифрування та дешифрування інформації в системі. Він реалізує методи для використання алгоритму шифрування, AES у відповідності до безпекових вимог програми;

– клас RoleApp управляє ролями та дозволами користувачів у додатку. Він може містити методи для перевірки дозволів, асоціації ролей із діями, а також для модифікації ролей в режимі реального часу;

– клас ValidationMy задіяний у процесах валідації даних, які надходять від користувача та інших системних компонентів. Містить методи для перевірки цілісності, відповідності форматам та іншим вимогам до даних.

### 3.4.2 Рівень користувацького інтерфейсу

Рівень UI (User Interface) служить для взаємодії користувача з програмою. Цей рівень відповідає за створення користувацького інтерфейсу, валідацію даних, а також за передачу інформації до рівня бізнес-логіки (BLL) [30]. Основні елементи UI, такі як кнопки та текстові поля, спрощують навігацію та взаємодію користувача з додатком. Діаграма класів цього рівня представлена на рис. 3.5.

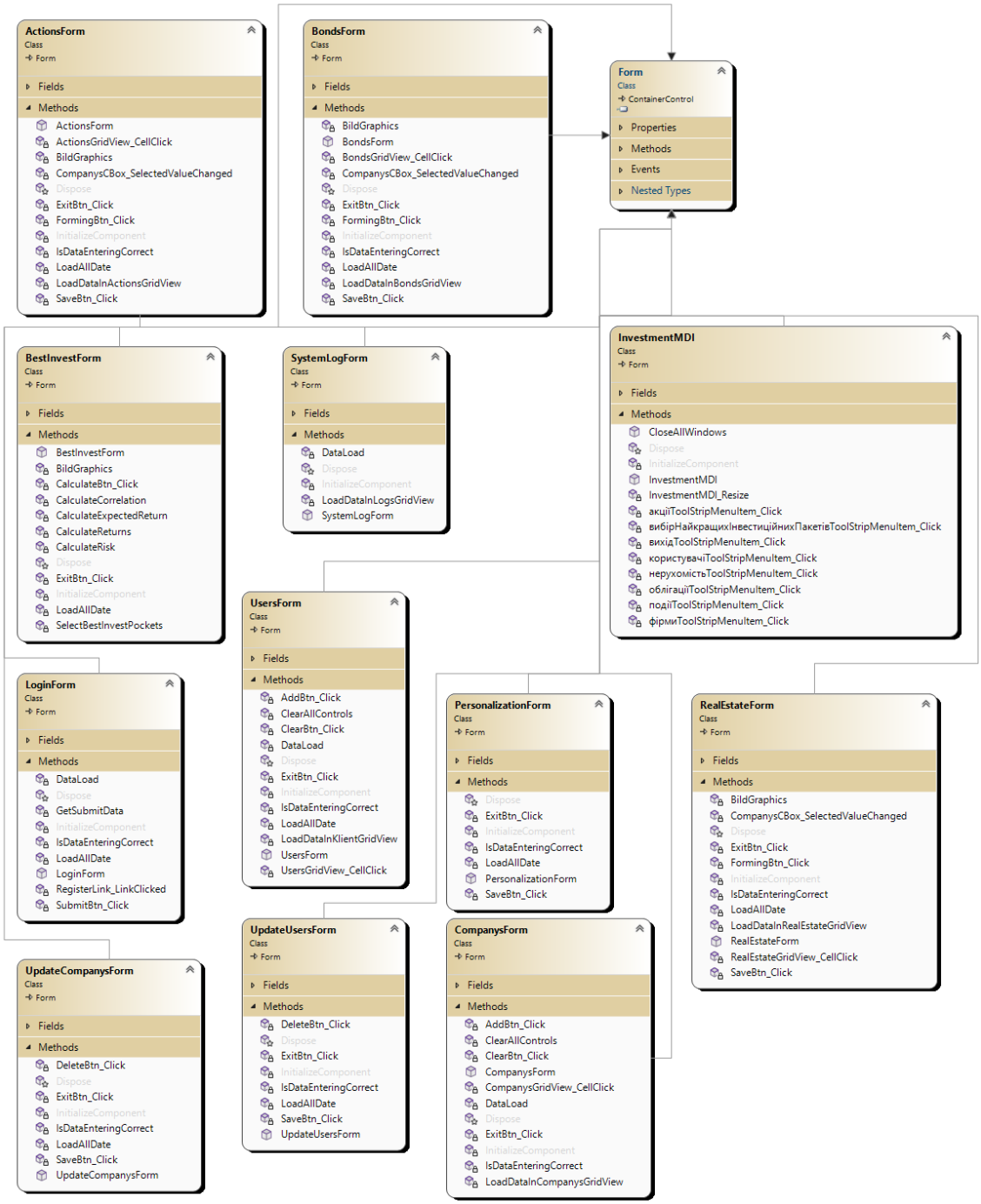



Рис 3.5 Діаграма класів рівня користувацького інтерфейсу

Рівень користувацького інтерфейсу містить одинадцять класів. Найбільш важливішими серед них є:

– клас InvestmentMDI служить як головна MDI (Multiple Document Interface) форма, в якій будуть відкриватися всі інші дочірні форми (наприклад, BestInvestForm, ActionsForm тощо);

– клас BestInvestForm призначений для аналізу та вибору оптимальних інвестиційних портфелів;

– клас ActionsForm призначений для роботи з акціями на ринку;

– клас BondsForm призначений для роботи з облігаціями;

– клас UpdateCompanysForm призначений для оновлення інформації про компанії, їх фінансові показники та стан на ринку;

– клас CompanysForm діє як інтерфейс для відображення та управління даними про компанії;

– клас RealEstateForm призначений для управління інвестиціями в нерухомість. Основні методи кключають: перегляд, додавання, та редагування інформації про нерухомість;

– клас LoginForm служить для аутентифікації користувачів. Основні методи складають: перевірка введених користувачем даних, авторизація або відхилення доступу;

– клас PersonalizationForm дозволяє користувачам налаштовувати індивідуальні параметри системи;

– клас SystemLogForm призначений для перегляду системних журналів активності;

– клас UpdateUsersForm призначений для оновлення даних про користувачів системи;

– клас UsersForm призначений для відображення та управління списком користувачів.

Кожен із зазначених класів виконує важливу роль у забезпеченні функціональності, безпеки та надійності системи оптимізації вибору інвестиційних портфелів.

### 3.4.3 Рівень доступу до даних

Рівень доступу до даних (DAL) служить механізмом для взаємодії з базою даних і виконання операцій над даними. У цьому рівні для кожної таблиці бази даних генерується специфічний клас, який надає методи для маніпуляцій з даними, таких як читання, додавання, модифікація та вилучення записів.

Крім того, для кожного такого класу формується відповідний інтерфейс, що визначає набір методів для доступу до даних. Ця конструкція сприяє гнучкості розробки, дозволяючи легко заміняти одну реалізацію DAL іншою без негативного впливу на решту архітектури програми, що полегшує етапи розробки та тестування.

Схематичне зображення класів даного рівня можна побачити на рис. 3.6.

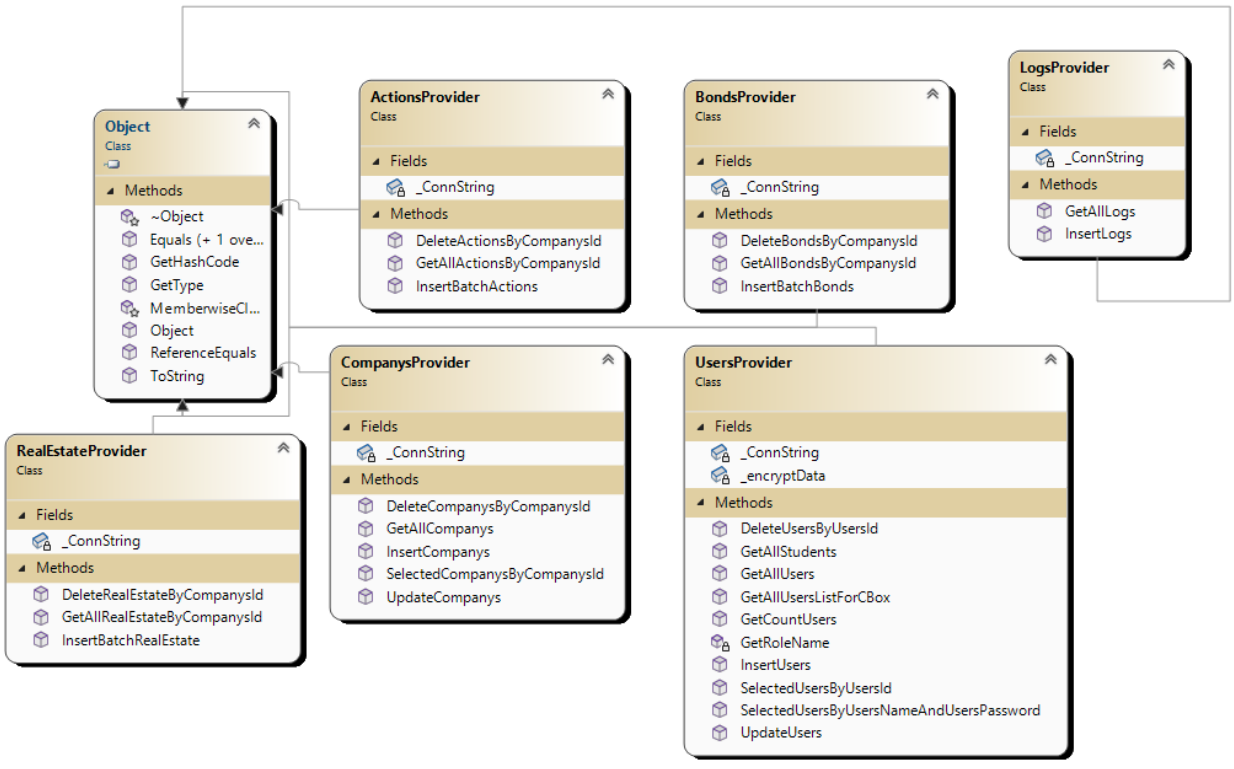



Рис 3.6 Діаграма класів рівня даних

Архітектура рівня доступу до даних (Data Access Layer, DAL) включає в себе набір класів, які служать для специфічної взаємодії з базою даних. Дозвольте мені описати основні класи цього рівня:

– клас ActionsProvider призначений для взаємодії з таблицею акцій в базі даних. Він містить методи для зчитування інформації про акції, їх додавання, редагування та видалення;

– клас BondsProvider аналогічно ActionsProvider, але специфічно орієнтований на облігації. Містить методи для управління записами в таблиці облігацій, такі як додавання нових облігацій, зміна існуючих або їх видалення;

– клас CompanysProvide призначений для взаємодії з даними про компанії. Методи цього класу включають: зчитування, додавання, оновлення і видалення інформації про компанії;

– клас LogsProvider забезпечує роботу з системними журналами. Через нього можливе зберігання, аналіз та відстеження різних типів системних подій, що сприяє забезпеченню безпеки та моніторингу;

– клас RealEstateProvider відповідальний за взаємодію з інформацією про нерухомість. Методи включають дії з додавання, редагування, видалення, а також зчитування записів про різні типи нерухомості;

– клас UsersProvider керує даними користувачів. Через нього реалізується доступ до даних про користувачів, їх роль у системі, авторизацію, а також може включати методи для аудиту та моніторингу дій користувачів.

Кожен з цих класів розроблений з урахуванням принципів інкапсуляції та абстракції, що забезпечує високий рівень безпеки та масштабованість системи.

**Висновок до розділу**

У даному розділі акцентовано увагу на ключових аспектах технологічної та архітектурної організації програмного продукту. Починаючи з вибору оптимальних технологічних компонентів, таких як мова програмування C# та система управління базами даних MS SQL Server, було покладено фундамент для подальшої реалізації проекту.

Аналітичний огляд вимог, що включав розробку діаграм прецедентів для різних ролей користувачів, надав детальний огляд очікуваних сценаріїв взаємодії

з системою. Специфікація розробки бази даних, у свою чергу, забезпечила ясність щодо структуризації та управління даними.

Архітектурна структура програмного забезпечення була спроектована з використанням трьохрівневої моделі. Це включає в себе рівень бізнес-логіки, який відповідає за обробку даних та бізнес-правил; рівень користувацького інтерфейсу, який забезпечує взаємодію з користувачем; та рівень доступу до даних, що управляє взаємодією з базою даних.

В цілому, розділ представляє цілісний і деталізований план технологічної та архітектурної організації системи, що є необхідним для забезпечення її функціональності, ефективності та масштабованості. Ці рішення створюють надійну основу для подальшої реалізації та впровадження проекту.

# 4 ПРОЦЕС РОЗРОБКИ, АДАПТАЦІЇ ТА ВЕРИФІКАЦІЇ ПРОГРАМНИХ КОМПОНЕНТІВ З ВИКОРИСТАННЯМ ОПТИМІЗАЦІЙНИХ ТЕХНІК

## 4.1 Конструювання та оптимізація програмних компонентів

Наступна стадія розробки після інтеграції бази даних в середовищі Visual Studio 2019 полягає у розробці коду для автоматизації та оптимізації вибору інвестиційних портфелів. Важливим моментом є використання алгоритмів аналізу часових рядів, які стали фундаментальною частиною використовуваної математичної моделі.

Першочергово, створюємо змінну «CONNECT» в конфігураційному файлі «App.config» для визначення параметрів з'єднання з базою даних, подробиці якого зображено на рис. 4.1.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECT" value="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\DB.mdf;Integrated Security=True" />
</appSettings>
```

Рис 4.1 Змінна  з налаштуваннями підключення до бази даних

Для забезпечення оптимізованої взаємодії з базою даних в системі використовується простір імен System.Data.SqlClient. Цей простір імен входить в стандартну бібліотеку платформи .NET і є спеціалізованим для роботи з базами даних, що працюють на SQL Server. Він містить класи, які реалізують ряд функцій: від встановлення з'єднання з базою даних до виконання SQL-запитів та збереження внесених користувачем змін. Це дозволяє системі взаємодіяти з базою даних в деталізований та безпечний спосіб, мінімізуючи ризики пов'язані з некоректною роботою з даними.

У контексті користувацького інтерфейсу важливим є введення компоненту menuStrip, інтегрованого у головне вікно програмного додатку. MenuStrip є стандартним елементом управління у бібліотеках для розробки графічного

інтерфейсу на платформі Windows. Цей компонент надає можливість динамічно формувати меню з опціями, що можна вибирати, тим самим забезпечуючи користувачам доступ до різноманітних функцій програми через графічний інтерфейс (рис. 4.2). Така структура сприяє інтуїтивному розумінню програми та забезпечує швидкий доступ до її основних функцій.

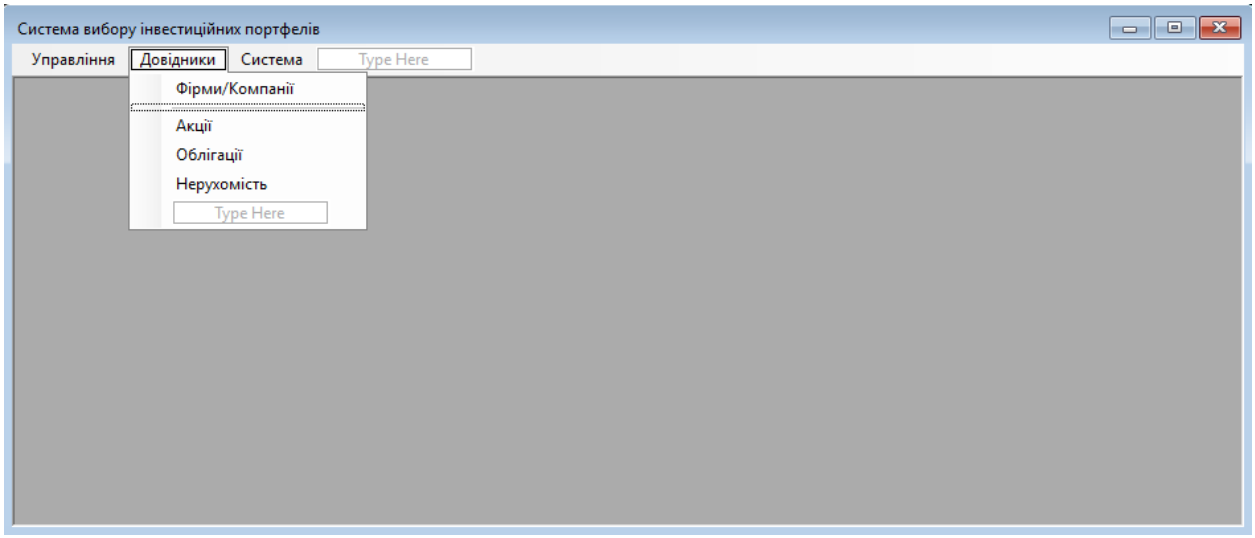



Рис 4.2 Додавання головного меню

Для управління вікнами форм в програмі інкапсульований код був прив'язаний до різних елементів меню. Цей код ініціалізує новий об'єкт форми та активує відповідне вікно при користувацькому виборі даного пункту меню. Додатково, перед активацією нового вікна, код автоматично деактивує і закриває попереднє вікно для уникнення потенційних проблем з відображенням та конфліктів в системі.

Ця методика реалізації була уніфікована для всіх елементів меню, що сприяє консистентності у функціоналі та взаємодії з різними формами програми. Приклад такої реалізації коду можна бачити на рис. 4.3.

```
1 reference
private void облігаціїToolStripMenuItem_Click(object sender, EventArgs e) {
  CloseAllWindows();
  BondsForm bondsForm = new BondsForm();
  bondsForm.MdiParent = this;
  bondsForm.WindowState = FormWindowState.Maximized;
  bondsForm.Show();
}
```

Рис 4.3 Код пунктів меню

Так архітектурна стратегія не тільки полегшує задачу програмування, забезпечуючи можливість повторного використання коду для різних пунктів меню та вікон форм, але також забезпечує плавність та правильність функціонування програми під час її використання.

Після вказаних етапів були сформовані класи, задачею яких є взаємодія з базою даних. Для кращого розуміння функціональності, надалі будуть наведені фрагменти методів цих класів з докладним поясненням. Як приклад, метод з ім'ям «InsertBatchBonds», призначений для поповнення бази даних даними про цінову динаміку облігацій. Вихідний код цього методу можна переглянути на рис. 4.4.

```
1 reference
public void InsertBatchBonds(List<Bonds> BondsList, int CompanysId) {
  string SqlString = "INSERT INTO Bonds (BondsDate, BondsValues, CompanysId) " +
    "VALUES (@BondsDate, @BondsValues, @CompanysId)";
  DeleteBondsByCompanysId(CompanysId);
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      conn.Open();
      for (int i = 0; i < BondsList.Count; i++) {
        cmd.Parameters.AddWithValue("@BondsDate", BondsList[i].BondsDate);
        cmd.Parameters.AddWithValue("@BondsValues", BondsList[i].BondsValues);
        cmd.Parameters.AddWithValue("@CompanysId", BondsList[i].CompanysId);
        cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
      }
      conn.Close();
    }
  }
}
```

Рис 4.4 Код методу «InsertBatchActions»

Метод InsertBatchBonds є елементом системи для масового внесення даних про облігації в базу даних. Цей метод ініціалізує SQL-запит, який використовує параметризовані значення, що значущо збільшує захист від можливих SQL-ін'єкцій. Перед внесенням нових даних метод видаляє всі існуючі записи, що відносяться до заданої компанії. Це здійснюється для уникнення дублювання та забезпечення консистентності бази даних.

Після ініціалізації SQL-запиту і видалення старих записів здійснюється відкриття нового підключення до бази даних. Тоді, в контексті цього підключення, створюється об'єкт для виконання SQL-команд. Застосування циклу

для перебору всіх елементів у переданому списку облігацій дозволяє забезпечити масову вставку даних. У кожній ітерації циклу параметри заповнюються значеннями з поточного елементу списку облігацій, після чого SQL-команда виконується, а її параметри очищуються для наступної ітерації.

Цей процес продовжується до того моменту, коли всі елементи списку не будуть оброблені. Заключним етапом є закриття підключення до бази даних, що завершує цикл життя даного методу. Таким чином, метод не тільки оптимізує роботу з базою даних, але і забезпечує високий рівень безпеки та консистентності даних.

Завдяки створенню методу «GetAllBondsByCompanysId», який ілюстровано на рис. 4.5, забезпечено можливість вилучення даних з файлу бази даних.

```
public List<Bonds> GetAllBondsByCompanysId(int CompanysId) {
  int i = 0;
  string SqlString = "SELECT * FROM Bonds WHERE CompanysId=@CompanysId";
  List<Bonds> listBonds = new List<Bonds>();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Bonds oneBonds = new Bonds();
          oneBonds.Number = ++i;
          oneBonds.BondsId = Convert.ToInt32(reader["BondsId"]);
          oneBonds.BondsDate = Convert.ToDateTime(reader["BondsDate"]);
          oneBonds.BondsValues = Convert.ToDouble(reader["BondsValues"]);
          oneBonds.CompanysId = Convert.ToInt32(reader["CompanysId"]);
          listBonds.Add(oneBonds);
        }
      }
      conn.Close();
    }
  }
  if (listBonds.Count == 0) {
    Bonds noBonds = new Bonds();
    noBonds.BondsId = 0;
    noBonds.Message = NamesMy.NoDataNames.NoDataInBonds;
    listBonds.Add(noBonds);
  }
  return listBonds;
}
```

Рис 4.5 Код методу «GetAllBondsByCompanysId»

Метод "GetAllBondsByCompanysId" призначений для вилучення даних про облігації конкретної компанії з бази даних. Цей метод приймає як аргумент ідентифікатор компанії, що зберігається в змінній "CompanysId", і повертає список об'єктів класу "Bonds", що містить деталізовану інформацію про облігації.

Під час виконання методу створюється з'єднання з базою даних через об'єкт "SqlConnection". Потім ініціалізується SQL-запит для вибірки всіх записів з таблиці "Bonds", які відповідають заданому ідентифікатору компанії.

Для виконання SQL-запиту використовується об'єкт "SqlCommand", який передає параметр "CompanysId" в SQL-запит. Після відкриття з'єднання з базою даних, метод читає повернуті рядки через об'єкт "SqlDataReader", створюючи нові екземпляри класу "Bonds" для кожного рядка та додаючи їх до раніше ініціалізованого списку "listBonds".

В кінці методу проводиться перевірка на те, чи були отримані які-небудь даних. Якщо список залишився порожнім, в нього додається об'єкт з повідомленням про відсутність даних.

Після завершенням конструювання усіх даних класів, були створені користувацькі інтерфейси для взаємодії з програмою. Однією з таких інтерфейсних форм є панель для вибору оптимальних інвестиційних планів, ілюстрація до якої приведена на рисунку 4.6.

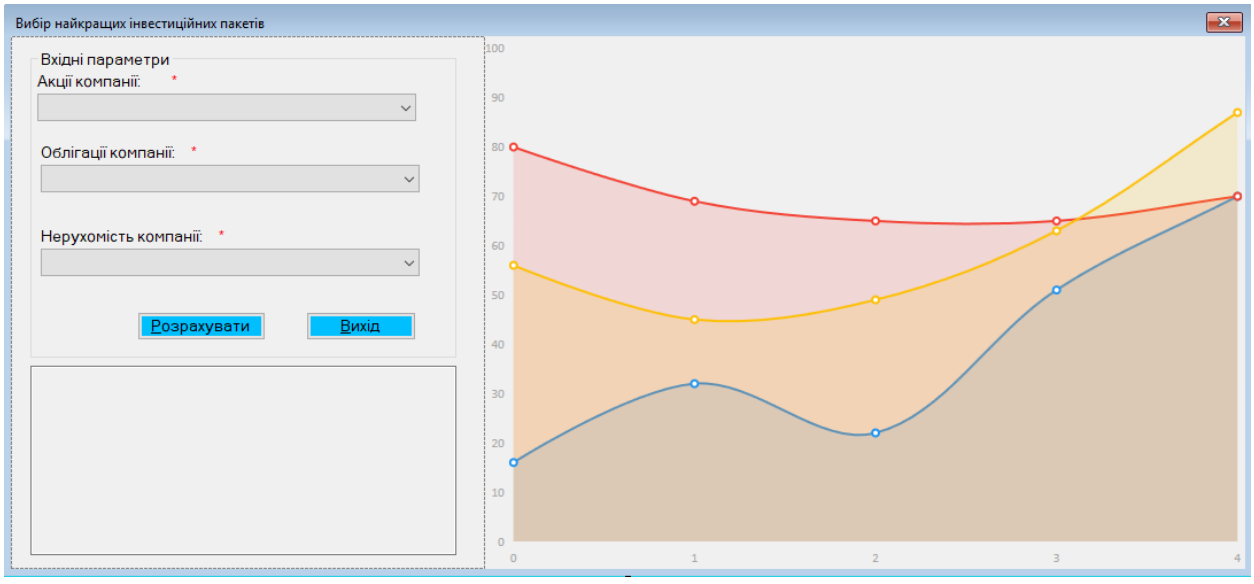



Рис 4.6 Форма вибору кращих інвестиційних пакетів

Під час ініціалізації зазначеної форми, в її конструкторі автоматично активується метод "LoadAllDate". Цей метод завдяки своїй роботі наповнює розкриваючі списки даними про різні види інвестицій — акції, облігації та нерухомість, що ілюстровано на рис. 4.7.

```
private void LoadAllDate() {
  _CompanysActionsList = _CompanysProvider.GetAllCompanys();
  ActionsCompanysCBox.DataSource = _CompanysActionsList;
  ActionsCompanysCBox.ValueMember = "CompanysId";
  ActionsCompanysCBox.DisplayMember = "CompanysName";

  _CompanysBondsList = _CompanysProvider.GetAllCompanys();
  BondsCompanysCBox.DataSource = _CompanysBondsList;
  BondsCompanysCBox.ValueMember = "CompanysId";
  BondsCompanysCBox.DisplayMember = "CompanysName";

  _CompanysRealEstateList = _CompanysProvider.GetAllCompanys();
  RealEstateCompanysCBox.DataSource = _CompanysRealEstateList;
  RealEstateCompanysCBox.ValueMember = "CompanysId";
  RealEstateCompanysCBox.DisplayMember = "CompanysName";
}
```

Рис 4.7 Код методу «LoadAllDate»

З метою керування процесами розрахунків була створена функція-реагування на подію під назвою «CalculateBtn_Click», програмний код якої зображено на рис. 4.8.

```
private void CalculateBtn_Click(object sender, EventArgs e) {
  _ActionsList = _ActionsProvider.GetAllActionsByCompanysId(Convert.ToInt32(ActionsCompanysCBox.SelectedValue));
  _BondsList = _BondsProvider.GetAllBondsByCompanysId(Convert.ToInt32(BondsCompanysCBox.SelectedValue));
  _RealEstateList = _RealEstateProvider.GetAllRealEstateByCompanysId(Convert.ToInt32(RealEstateCompanysCBox.SelectedValue));

  countElement = Math.Min(_ActionsList.Count, Math.Min(_BondsList.Count, _RealEstateList.Count));
  if (countElement > 1) {
    BildGraphics();
    SelectBestInvestPockets();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було проведено вибір найкращих " +
      "інвестиційних пакетів", DateTime.Now);
  } else {
    MessageBox.Show("Не достатньо інформаційї для знаходження " +
      "найкращих інвестиційних пакетів", "Увага!");
  }
}

1 reference
private void ExitBtn_Click(object sender, EventArgs e) {
  this.Close();
}
```

Рис 4.8 Код обробника події «CalculateBtn_Click»

Метод CalculateBtn_Click активується при натисненні кнопки "Calculate" в користувацькому інтерфейсі. Він здійснює ряд ключових дій для аналізу

інвестиційних портфелів. Початково метод збирає всі доступні дані з акцій, облігацій і нерухомості для вибраних компаній. Отримані дані зберігаються в трьох різних списках.

Після цього метод визначає мінімальну кількість елементів серед цих списків. Якщо ця мінімальна кількість перевищує одиницю, метод здійснює подальший аналіз. Він викликає функцію для побудови графічного представлення, функцію для вибору найкращих інвестиційних портфелів, а також записує інформацію про проведений аналіз у системний журнал.

Якщо ж мінімальна кількість елементів не перевищує одиницю, користувачу виводиться повідомлення про недостатність даних для проведення відповідного аналізу. Таким чином, метод є ключовим елементом у процесі вибору оптимальних інвестиційних рішень.

На рис. 4.9 проілюстровано програмний код методу, який розраховує денні прибутки з кожного типу інвестиційних ресурсів.

```
3 references
private double[] CalculateReturns(double[] prices) {
    double[] returns = new double[prices.Length - 1];
    for (int i = 0; i < prices.Length - 1; i++) {
        returns[i] = (prices[i + 1] - prices[i]) / prices[i];
    }
    return returns;
}
```

Рис 4.9 Код методу «CalcReturn»

Метод "CalcReturn" є частинною більшої системи та спрямований на обчислення щоденних доходів для масиву цін інвестиційного активу. Вхідний параметр методу — це масив "prices", який містить історичні ціни активу. Він повертає новий масив "returns", довжина якого на одиницю менша від довжини вхідного масиву.

Процес виконання методу можна розбити на наступні кроки:

– ініціалізація нового масиву "returns" довжиною "prices.Length: 1";
– виконання циклу "for", який ітерується від 0 до "prices.Length: 2";

– кожна ітерація циклу обчислюється величина доходу за формулою (prices[i + 1] - prices[i]) / prices[i] та зберігається у відповідному елементі масиву "returns".

По завершенню циклу метод повертає масив "returns", який містить обчислені щоденні доходи. Ці дані можна використовувати для подальшого аналізу ефективності інвестиційного портфеля.

Метод, призначений для визначення ризиків, було створено і представлено на ілюстрації 4.10.

```
private double CalcRisk(double[] returns) {
  double variance = 0;
  double averageReturn = 0;
  for (int i = 0; i < returns.Length; i++) {
    averageReturn += returns[i];
  }
  averageReturn /= returns.Length;
  for (int i = 0; i < returns.Length; i++) {
    variance += Math.Pow((returns[i] - averageReturn), 2);
  }
  variance /= (returns.Length - 1);
  double risk = Math.Sqrt(variance);
  return risk;
}
```

Рис 4.10 Код методу «CalcRisk»

Метод CalcRisk здійснює розрахунок ризику на основі масиву повернення інвестицій (returns). Він визначає середнє арифметичне зі всіх значень повернення інвестицій, додаючи кожне значення з масиву до загальної суми, яку потім ділить на кількість елементів у масиві. Далі знаходиться дисперсія цих значень. Знову ітеруючись через масив, метод розраховує квадрат відхилення кожного елементу від обчисленого середнього значення. Кожен з цих квадратів додається до загальної суми, яка потім ділиться на кількість елементів масиву зменшену на одиницю. Корінь квадратний від отриманої дисперсії являє собою ризик, який і стає результатом роботи методу. Ця метрика може бути використана для кількісного аналізу рівня ризику в інвестиційних портфелях.

На рис. 4.11 зображено програмний код методу, призначеного для визначення очікуваного прибутку.

```
3 references
private double CalcExpectedReturn(double[] returns) {
  double expectedReturn = 0;
  for (int i = 0; i < returns.Length; i++) {
    expectedReturn += returns[i];
  }
  expectedReturn /= returns.Length;
  return expectedReturn;
}
```

Рис 4.11 Метод розрахунку очікуваної дохідності

Метод CalcExpectedReturn реалізований для визначення очікуваної дохідності на основі переданих значень прибутків в масиві returns. Виконується ітераційний процес по всім елементам масиву, де кожний елемент додається до змінної expectedReturn. Після цього змінна expectedReturn ділиться на кількість елементів у масиві returns для знаходження середнього значення. Отримане середнє значення повертається як результат методу.

Таким чином, метод виконує вирахування середнього арифметичного з наданих значень прибутків, що в свою чергу вважається оцінкою очікуваної дохідності інвестиційного активу.

На рис. 4.12 зображено програмний код методу для розрахунку показника кореляції.

```
static double CalcCorrelation(double[] x, double[] y) {
  double sumX = 0;
  double sumY = 0;
  double sumXY = 0;
  double sumXSquare = 0;
  double sumYSquare = 0;
  int n = x.Length;
  for (int i = 0; i < n; i++) {
    sumX += x[i];
    sumY += y[i];
    sumXY += x[i] * y[i];
    sumXSquare += x[i] * x[i];
    sumYSquare += y[i] * y[i];
  }
  double correlation = (n * sumXY - sumX * sumY) /
    Math.Sqrt((n * sumXSquare - sumX * sumX) * (n * sumYSquare - sumY * sumY));
  return correlation;
}
```

Рис 4.12 Метод розрахунку коефіцієнт кореляції

Метод CalcCorrelation представляє собою процедуру для обчислення коефіцієнта кореляції між двома наборами даних x і y, представленими у вигляді масивів. В алгоритмі методу використовується формула Пірсона для обчислення коефіцієнта кореляції.

Алгоритм виконує наступні основні етапи:

– ініціалізація змінних для зберігання суми елементів кожного з масивів, а також суми їх квадратів і суми добутків відповідних елементів двох масивів;

– ітерація через елементи масивів для обчислення зазначених сум;

– використання обчислених сум для знаходження коефіцієнта кореляції з допомогою формули Пірсона.

Метод повертає числове значення коефіцієнта кореляції, яке може варіюватися від -1 до 1. Значення близькі до 1 або -1 свідчать про сильну позитивну або негативну кореляцію між масивами, відповідно. Значення близькі до нуля індикують відсутність зв'язку між даними.

## 4.2 Оцінювання через функціональне та модульне тестування

При функціональному тестуванні використовуються попередньо розроблені тестові сценарії, що базуються на специфікації вимог до програмного продукту. Кожен тестовий сценарій містить вхідні параметри, очікувані результати і конкретні кроки для їх перевірки.

Для проведення функціонального тестування було розроблено чотири тестові сценарії, які мають на меті оцінити функціональність системи у ключових аспектах її роботи. Враховуючи, що дані формуються рандомно за обраний період часу, тестування намагалося відобразити реальні умови використання системи.

Тестовий сценарій 1: Перевірка формування масиву даних для акцій

Початкові умови:

– програма запущена, користувач авторизований;

– обрано період часу для генерації даних.

Кроки:

– вибрати розділ програми, що відповідає за роботу з акціями;

– запустити процес генерації рандомних даних за обраний період часу.

Очікуваний результат:

– масив даних про акції сформований відповідно до заданих параметрів.

Отриманий результат:

– дані сформовані та відображені в користувацькому інтерфейсі.

Тестовий сценарій 2: Перевірка формування масиву даних для облігацій

Початкові умови:

– програма запущена, користувач авторизований;

– обрано період часу для генерації даних.

Кроки:

– вибрати розділ програми, що відповідає за роботу з облігаціями;

– запустити процес генерації рандомних даних за обраний період часу.

Очікуваний результат:

– масив даних про облігації сформований відповідно до заданих параметрів.

Отриманий результат:

– дані сформовані та відображені в користувацькому інтерфейсі.

Тестовий сценарій 3: Перевірка формування масиву даних для нерухомості

Початкові умови:

– програма запущена, користувач авторизований;

– обрано період часу для генерації даних.

Кроки:

– вибрати розділ програми, що відповідає за роботу з нерухомістю;

– запустити процес генерації рандомних даних за обраний період часу.

Очікуваний результат:

– масив даних про нерухомість сформований відповідно до заданих параметрів.

Отриманий результат:

– дані сформовані та відображені в користувацькому інтерфейсі.

Тестовий сценарій 4: Вибір найкращого інвестиційного портфелю

Початкові умови:

– програма запущена, користувач авторизований;

– масиви даних про акції, облігації та нерухомість сформовані рандомно за обраний період часу.

Кроки:

– запустити функцію аналізу інвестиційного портфелю;

– проаналізувати результати.

Очікуваний результат:

– найкращий інвестиційний портфелю вибрано з урахуванням всіх доступних даних.

Отриманий результат:

– вибраний інвестиційний портфель відповідає очікуваному результату та представлено в користувацькому інтерфейсі.

Після завершення етапу функціонального тестування, на якому була перевірена здатність системи виконувати задані функції в заданих умовах, настає час для модульного тестування. Цей етап має велике значення в циклі розробки програмного забезпечення. Його основна мета полягає у детальному вивченні, аналізі та валідації функціональних аспектів окремих програмних модулів. Це надзвичайно важливо для забезпечення їх відповідності заздалегідь визначеним технічним критеріям і специфікаціям.

Щоб здійснити модульне тестування, було започатковано спеціалізований проект типу "Unit Test Project" в інтегрованому середовищі розробки Visual Studio. Це дало можливість розробникам фокусуватися на конкретних функціональних блоках коду, ізолювати їх для перевірки і тим самим гарантувати, що кожен модуль виконує свою роль ефективно та правильно.

Далі було розроблено набір юніт-тестів, які були спрямовані на перевірку внутрішньої логіки функцій, їх взаємодії з іншими компонентами, а також реакції на різні вхідні параметри. Після розробки цих тестів, вони були систематично

виконані, і результати були зібрані для аналізу. Дані про результати модульного тестування можна переглядати на рис. 4.13.

Цей методичний підхід до модульного тестування дозволяє не тільки ідентифікувати, але і коректувати можливі проблеми, недоліки або невідповідності в роботі окремих модулів. В результаті цього підходу значно підвищується загальна якість та надійність програмного продукту, забезпечуючи його стабільну та ефективну роботу в реальних умовах.

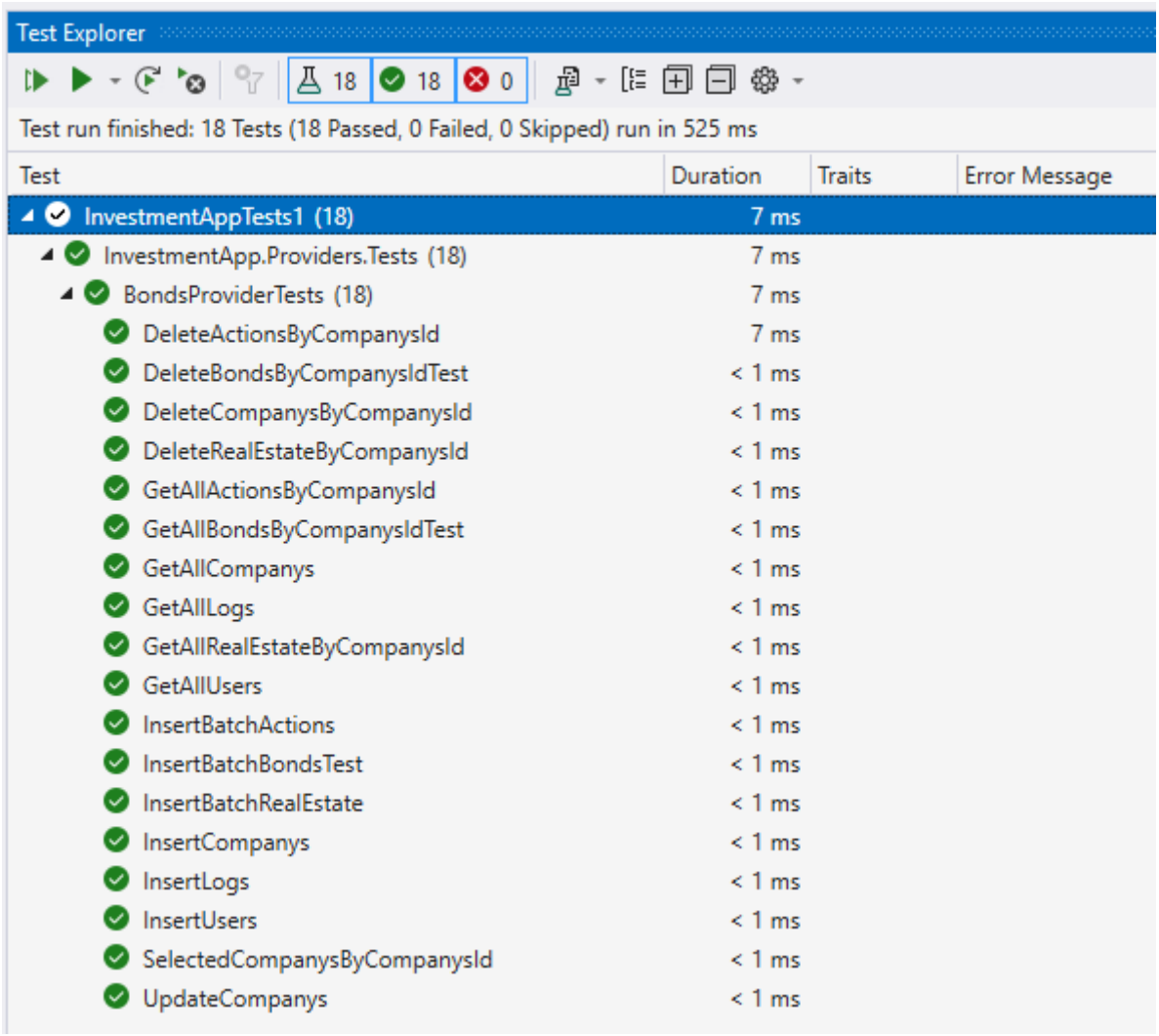



Рис 4.13 Результати проведення модульного тестування

Результати функціонального тестування дозволили не лише переконатися в коректності роботи окремих функцій системи, але й оцінити її ефективність при комплексному використанні. Додатково, тестування з рандомно згенерованими

даними показало, що система стабільна і не видає помилок при роботі з непередбачуваними сценаріями.

Модульне тестування підтвердило, що кожний з розроблених методів виконує свою роботу відповідно до поставлених завдань і специфікацій. Система продемонструвала високу надійність, як під час роботи з нормальними, так і з крайніми даними.

Отже, система пройшла всеобіймний цикл тестування, що включав функціональні та модульні тести. Результати підтвердили високу стабільність, надійність та відповідність всім технічним вимогам. Програмний продукт готовий до впровадження в експлуатацію та подальшого масштабування. З урахуванням позитивних результатів тестування, можна рекомендувати дану систему до широкого використання в інвестиційному секторі.

### 4.3 Керівництво користувача з описом процедур оптимізації

Перед взаємодією з програмою «Інвестиційні портфелі», користувач має ініціювати її запуск. Завершивши цей процес, користувачу представиться екран для введення своїх авторизаційних даних — логіну та секретного коду (рис. 4.14). Ця процедура не тільки визначає особу користувача, але і забезпечує конфіденційність його індивідуальної інформації.

По завершенню авторизації користувач отримує доступ до основного робочого простору програми. Тут він може взаємодіяти з даними про різні види активів: акції, облігації та нерухомість, які можна переглядати та модифікувати. Програма також містить інструменти для вибору оптимального інвестиційного портфелю.

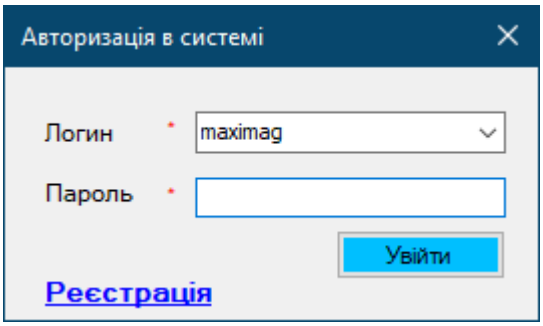



Рис 4.14 Автентифікація користувача

Якщо користувач ще не має створеного профілю в додатку, він має можливість його створити (див. рис. 4.15). На сторінці авторизації треба обрати опцію «Реєстрація». Виконавши цей крок, користувачу відкриється вікно для заповнення реєстраційних даних. В цій формі потрібно вказати основну інформацію: ім'я, прізвище, бажаний логін і пароль. Після того, як всі поля будуть коректно заповнені, користувачу слід нажати «Зберегти». З завершенням реєстраційного процесу користувач зможе увійти в додаток, використовуючи свій новостворений профіль.

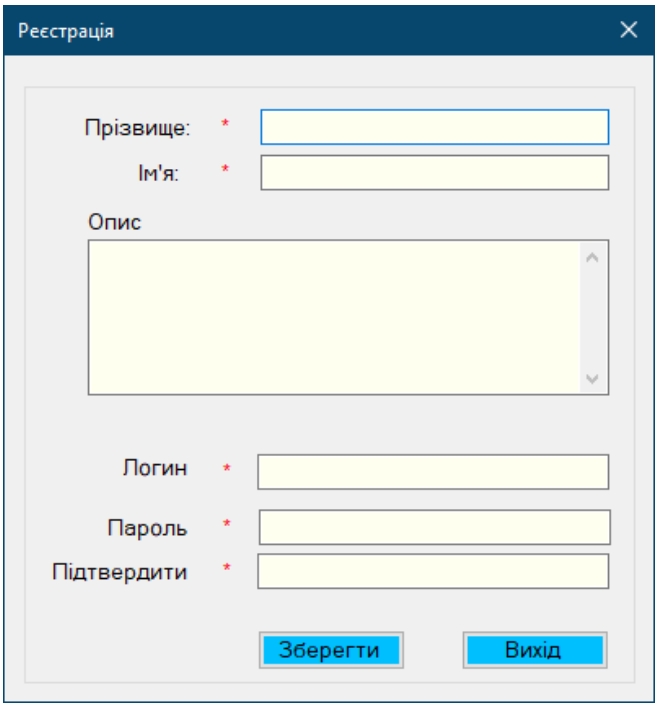



Рис 4.15 Форма реєстрації користувачів

У випадку, коли в системі відсутні зареєстровані користувачі, особі, яка здійснює першу реєстрацію, автоматично надається статус системного

адміністратора. Усім наступним реєстрованим користувачам за замовчуванням присвоюються ролі стандартних користувачів.

Для оптимізації процесу пошуку імені в програмі існує функція випадаючого списку з іменами користувачів. При ручному введенні імені система автоматично переміщує його на вищу позицію в списку, полегшуючи таким чином навігацію та вибір потрібного імені.

По завершенню етапу автентифікації, користувачу відкривається основний інтерфейс програми (див. рис. 4.16). Цей інтерфейс включає в себе головне меню, яке об'єднує всі доступні для використання функціональні можливості додатку. Основний екран програми служить панеллю керування, через яку користувачі можуть взаємодіяти з різними функціями системи.


Рис 4.16 Головне меню програми

В архітектурі даної системи передбачено два типи користувацьких ролей: системного адміністратора та стандартного користувача. Системний адміністратор має можливість управління всіма обліковими записами, а також доступ до системних логів для моніторингу подій в системі.

Першочерговою задачею після входу в систему є внесення даних про різні фінансові інструменти, зокрема про корпорації та їхні акції, облігації та активи в сфері нерухомості. Ця інформація є ключовою для подальшої роботи системи з вибором інвестиційних портфелів.

Для внесення інформації про певну корпорацію або фінансову організацію, користувачу необхідно відкрити розділ "Довідники" в головному меню системи, далі перейти до підрозділу "Фірми/Компанії". Після цього користувачу відобразиться форма для введення або редагування даних, яка ілюстрована на рис. 4.17. У цій формі можна детально зазначити всю необхідну інформацію про фінансову структуру: від її назви до цін на акції, облігації та інші активи.

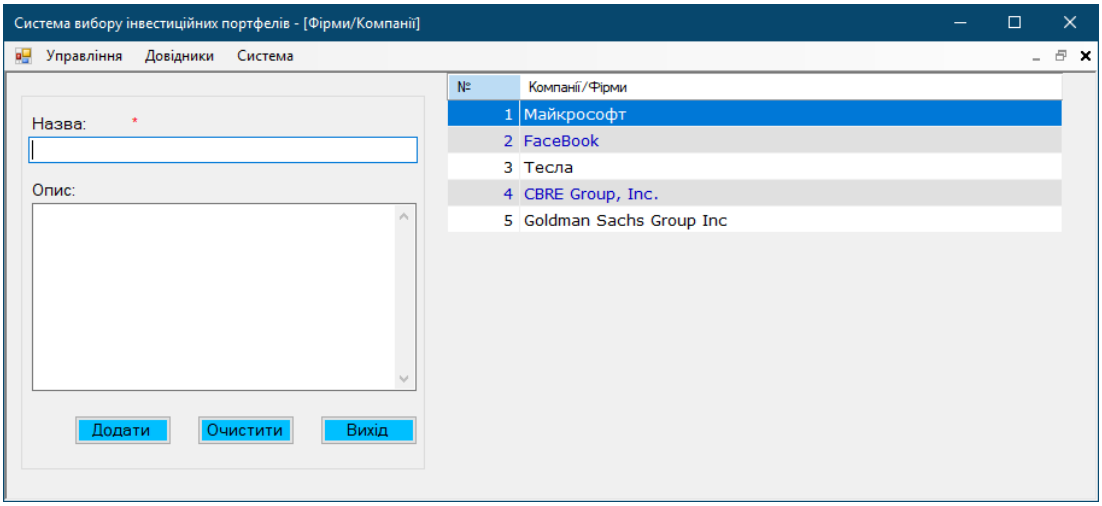



Рис 4.17 Форма опрацювання даних про фірми та компанії

Крім того, збережену інформацію про корпорації та організації можна редагувати. Для цього потрібно у правому сегменті форми виділити потрібний запис. Після цього користувачу стане доступною нова форма для редагування, яка зображена на рис. 4.18.

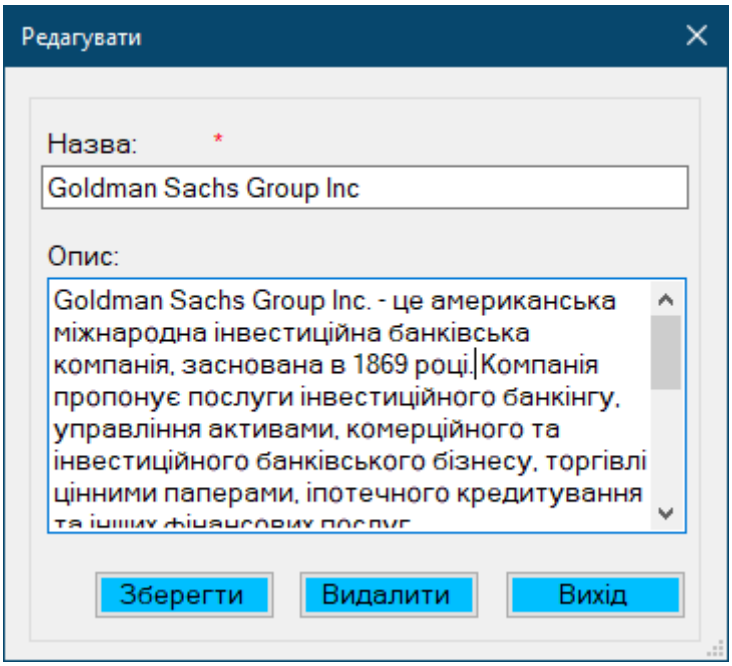



Рис 4.18 Форма редагування даних компаній та фірм

Щоб занести дані про вартість акцій, користувачу слід перейти до розділу «Довідники» в головному меню програми і обрати пункт «Акції». Зробивши це, на дисплеї з'явиться вікно із полями для вводу необхідної інформації, що ілюстровано на рис. 4.19.


Рис 4.19 Форма опрацювання даних акцій компаній

У випадку відсутності вказаних цін на акції, користувачу слід вибрати відповідний часовий проміжок та активувати кнопку «Формувати». Після цього програмний засіб автоматично створить шаблон для внесення даних про вартість

акцій з денним інтервалом. Користувач має заповнити ці поля інформацією про ціни. Завершивши введення інформації, необхідно активувати кнопку «Зберегти» для збереження цієї інформації в базі даних.

Методика роботи з облігаціями та нерухомістю є схожою на описану вище для акцій.

Для використання функціоналу системи щодо вибору оптимальних інвестиційних портфелів, користувач повинен перейти до розділу «Управління», де слід вибрати підпункт «Вибір найкращих інвестиційних пакетів». У з'явившомуся випадаючому списку варто обрати ті компанії, для яких було внесено дані щодо цін на акції, облігації та нерухомість. Після вибору компаній, користувачу потрібно активувати кнопку «Розрахувати». Система автоматично проведе необхідні аналітичні розрахунки та підбере компанії, що є найбільш привабливими для інвестицій, як показано на рис. 4.20.

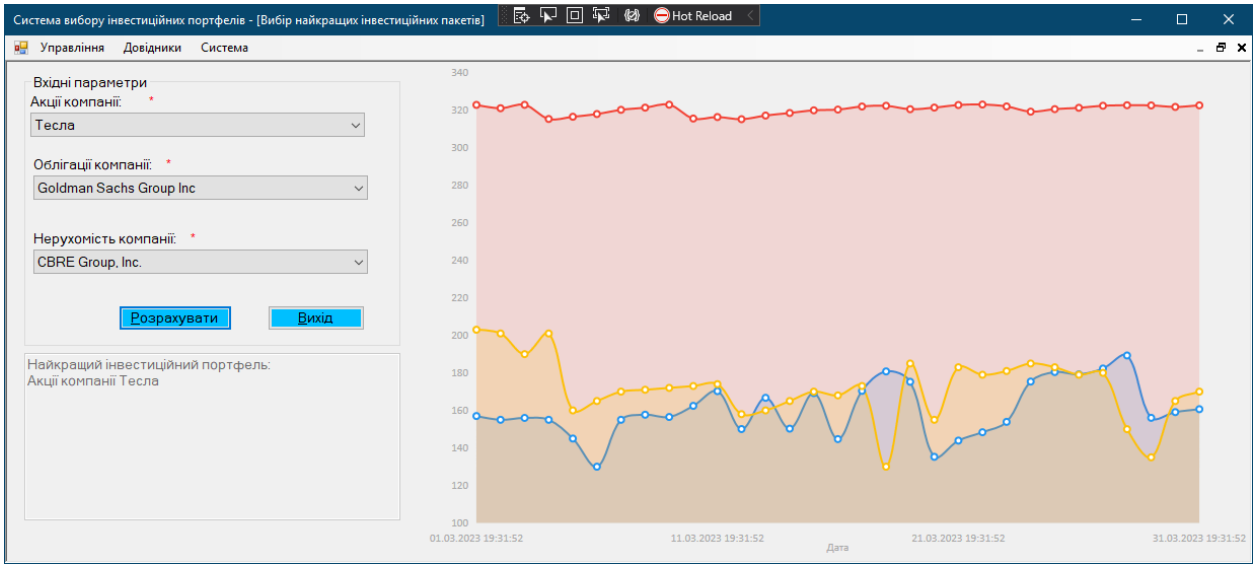



Рис 4.20 Форма вибору найкращих інвестиційних портфелів

Контроль над обліковими записами є критичним аспектом у функціонуванні багатьох інформаційних систем, включаючи розглядувану тут платформу. Особа, яка виконує функції системного адміністратора, обладнана повним спектром засобів для маніпуляції обліковими записами в системі.

Для того, щоб особа з функціональними обов'язками системного адміністратора могла здійснювати контроль над обліковими записами, потрібно виконати визначену послідовність дій. Спочатку у головному інтерфейсі

програми необхідно вибрати пункт меню «Користувачі», після чого активізується окреме вікно для управління користувацькими аккаунтами (рис. 4.21).

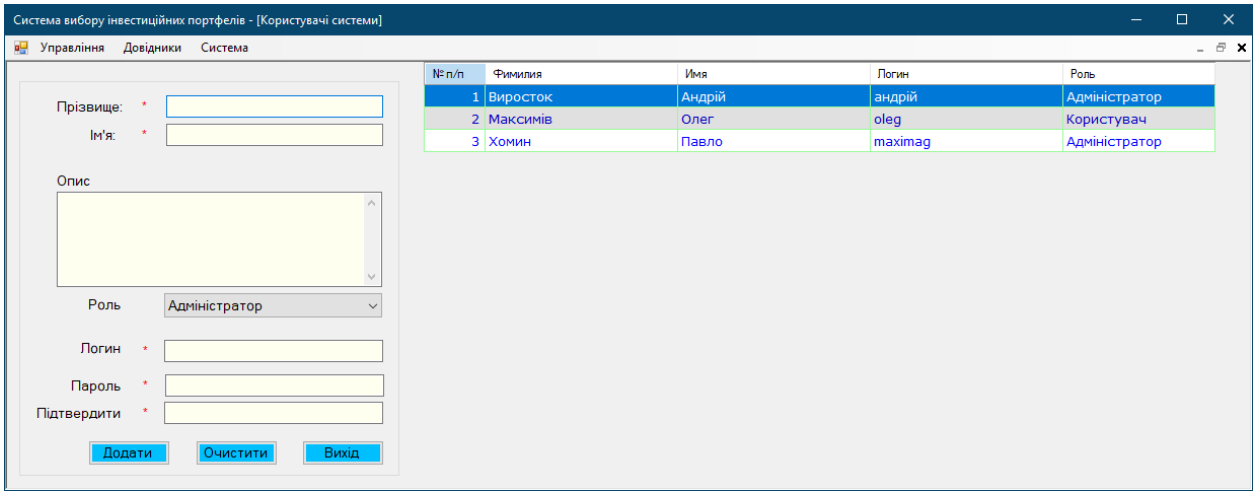



Рис 4.21 Управління обліковими записами

У цьому вікні особа з правами системного адміністратора може взаємодіяти з переліком існуючих користувацьких аккаунтів, модифікувати їх дані, ініціювати створення нових аккаунтів або ж вилучати існуючі.

У даній системі існує можливість моніторингу дій та активності користувачів за допомогою адміністративного облікового запису. Ця опція реалізована через функціональний розділ "Події", який можна знайти у меню під назвою "Система" (рис. 4.22).

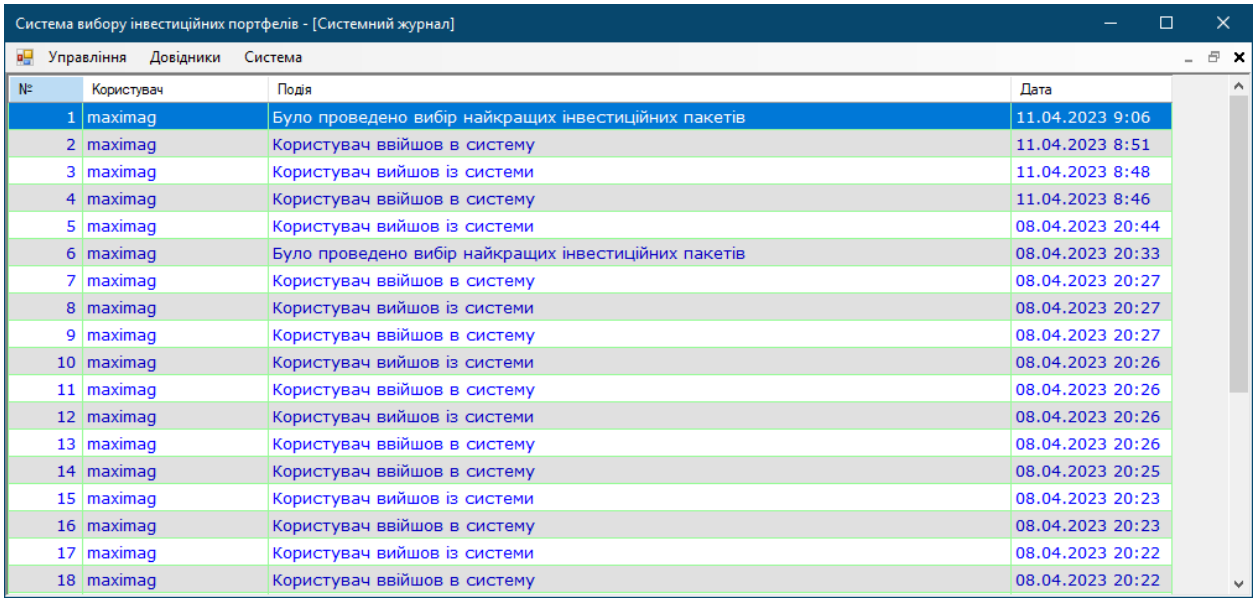



Рис 3.22 Події системи

У системі наявна функція доступу до системного журналу, в якому зібрано перелік всіх історичних подій системи, включаючи користувацькі дії, асоційовані з доступом до системи та процедурами оптимізації інвестиційних стратегій.

Ця інформація може слугувати важливим інструментом для діагностики проблем з безпекою і виявлення системних вразливостей. Окрім того, вона може використовуватися для контролю за користувацькою активністю та ефективного нагляду за їх діями.

Щодо завершення роботи із програмою, користувачу рекомендовано перейти до меню "Управління" та вибрати пункт "Вихід", що забезпечить надійне та безпечне закриття програми, мінімізуючи ризик втрати даних та потенційних безпекових інцидентів.

### 4.4 Організація та виконання експериментальних випробувань

Процедура визначення оптимального інвестиційного портфелю, що складається з акцій, облігацій та реальних активів, включає в себе декілька етапів.

Початково, збираються дані стосовно вартості кожного типу інвестиційного активу: акцій, облігацій та нерухомого майна.

Далі проводиться розрахунок актуальних ризиків і потенційних доходів для кожного активу за допомогою методів, які були попередньо описані: CalculateReturns, CalculateRisk та CalculateExpectedReturn.

На наступному етапі визначаються коефіцієнти кореляції між парами різних інвестиційних активів, використовуючи метод CalculateCorrelation.

Виходячи з отриманих даних, здійснюється вибір оптимального портфелю, який характеризується найвищим очікуваним доходом та найнижчим рівнем ризику. В разі відсутності однозначно домінуючого активу, вибір оптимального портфелю здійснюється на основі аналізу кореляції між різними активами.

Прикладом може служити експеримент на основі даних, отриманих з веб-сайту https://www.marketwatch.com/investing/stock/gs. В даному експерименті були

взяті вартості акцій компанії Tesla, нерухомості компанії CBRE Group, Inc та облігацій компанії Goldman Sachs Group Inc за березень 2023 року.

Після запуску додатку та використання меню "Фірми/Компанії" ці компанії були додані до сукупного переліку активів (рис. 4.23).

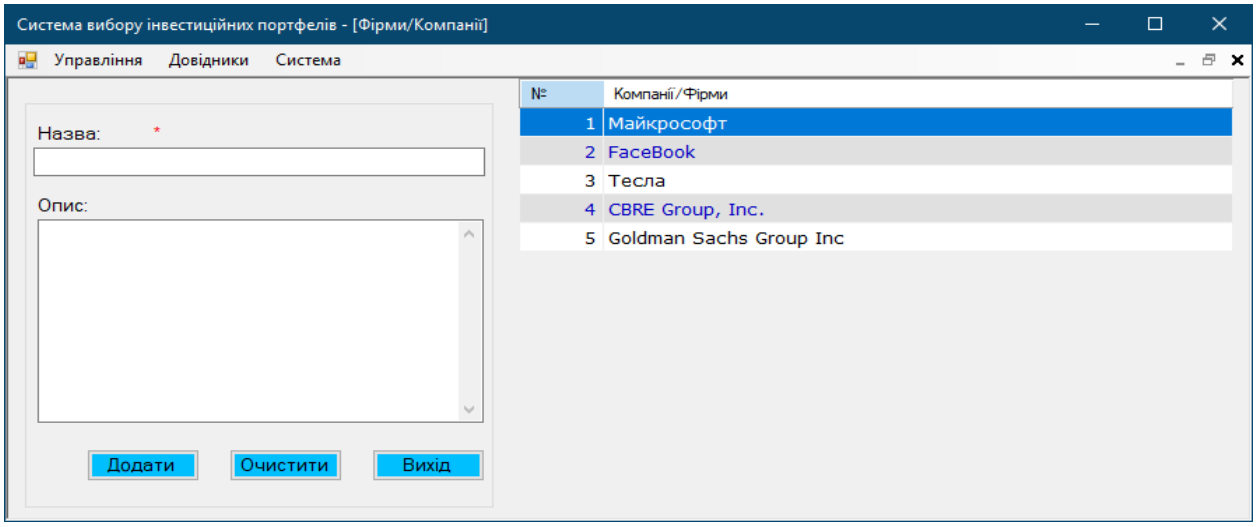



Рис 4.23 Опрацювання інформації про компанії

Наступним кроком є додавання даних про акції Tesla, що можна здійснити, перейшовши в програмному інтерфейсі через меню "Довідники" до підпункту "Акції" (рис. 4.24).

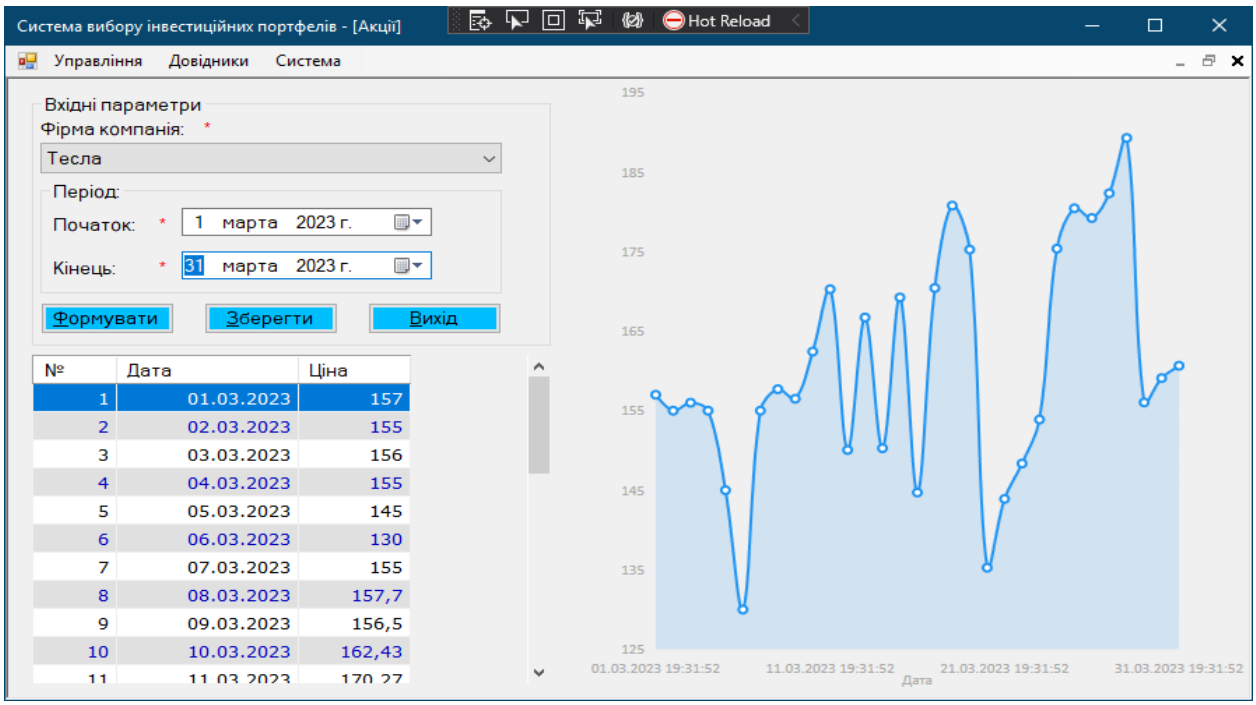



Рис 4.24 Формування цін на акції компанії Тесла

Відповідним методом реєструємо дані щодо активів в секторі нерухомості, які належать корпорації CBRE Group, Inc, а також інформацію про облігації, емітовані Goldman Sachs Group Inc. Ці процедури ілюстровані на графічних матеріалах 4.25 та 4.26 відповідно.

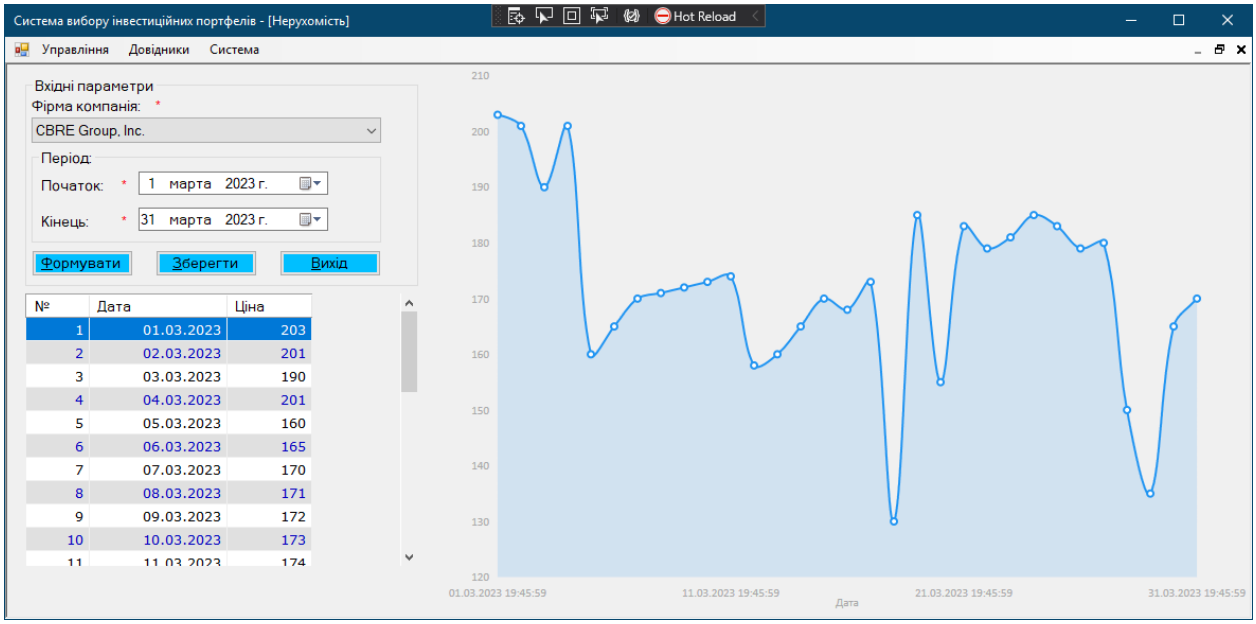



Рис 4.25 Формування цін на нерухомість компанії CBRE Group, Inc

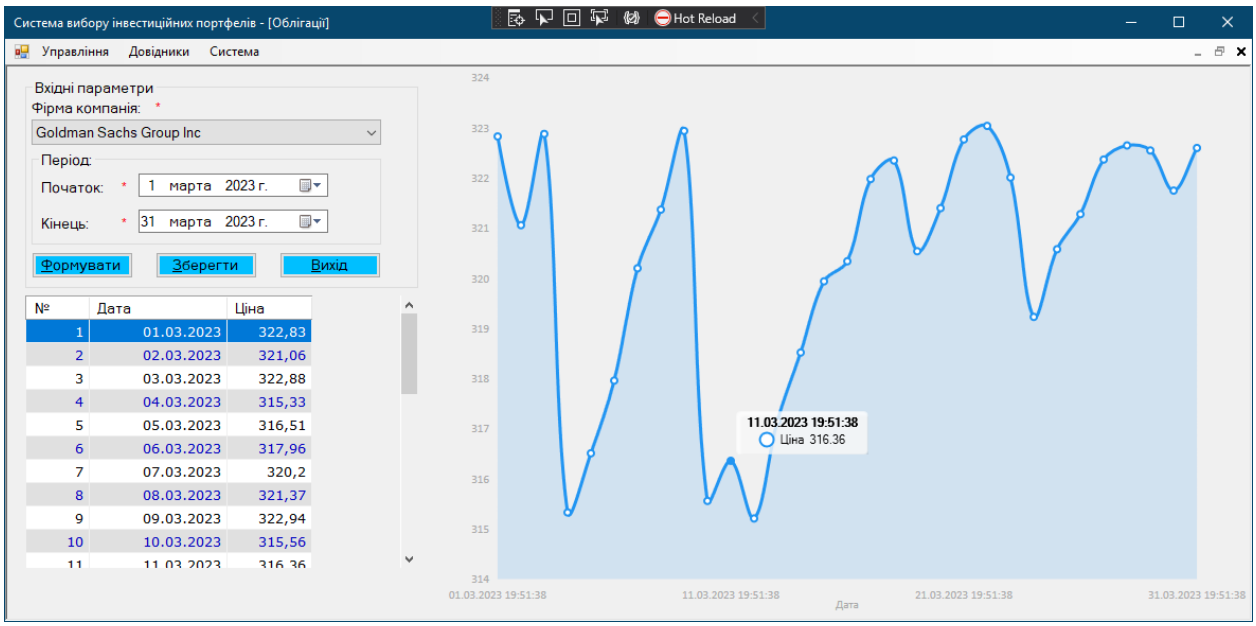



Рис 4.26 Формування цін на облігації Goldman Sachs Group Inc

Після внесення всієї інформації можливий перехід до етапу визначення оптимального інвестиційного портфелю на основі поданих даних. Для реалізації цього завдання рекомендується використовувати пункт меню програми, під назвою «Вибір найкращих інвестиційних пакетів». В субсекції, яка з'явиться на екрані, відповідно до випадаючого списку обрати корпорації, інформацію про які було раніше внесено (рис. 4.27).

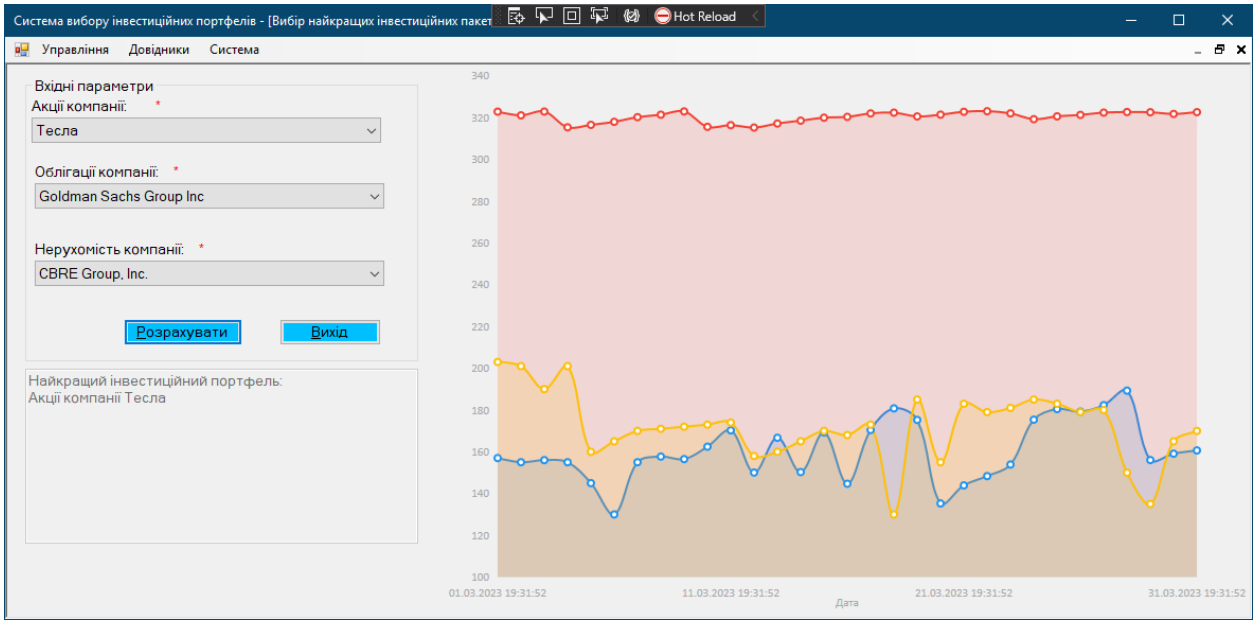



Рис 4.27 Результати вибору найкращих інвестиційних портфелів

За даними, представленими на рис. 4.17, після завершення аналітичних розрахунків програма зазначила у нижньому лівому сегменті інтерфейсу, що оптимальним варіантом інвестування в даній ситуації є акції компанії Tesla.

## 4.5 Оцінювання та інтерпретація отриманих даних з врахуванням аспектів оптимізації

Процес розробки системи вибору оптимальних інвестиційних портфелів зводиться до алгоритмічної ідентифікації варіантів інвестування, які максимізують дохід та мінімізують ризик. Це досягається через математичне моделювання, що включає в себе ряд критичних параметрів: дохідність, ризик та кореляційні взаємозв'язки між активами.

Система має вбудовані модулі для вимірювання цих критичних параметрів для різних типів інвестиційних активів, а саме акцій, облігацій та нерухомості. Застосування алгоритмічних методів дозволяє, на основі вхідних даних, робити обґрунтований вибір інвестиційного портфелю. Важливо підкреслити, що ця система не лише автоматизує розрахунки, але і втілює принципи оптимізації.

У конкретному випадку, на даних про акції компанії Tesla, облігації компанії Goldman Sachs Group Inc та активи на нерухомість компанії CBRE Group, Inc, система ідентифікувала акції компанії Tesla як найбільш оптимальний інвестиційний варіант.

Таке рішення засноване на кількості взаємопов'язаних факторів, які були ретельно вивчені та обчислені системою. Додаткова можливість додавання нових компаній та їх фінансово-економічних показників до бази даних системи надає гнучкість для адаптації до змінних ринкових умов.

Реалізована система вибору інвестиційних портфелів представляє собою когнітивний інструмент, який інтегрує інтелектуальну фільтрацію з авансованими алгоритмами для автоматичного відбору активів. Основуючись на ряді заданих критеріїв, таких як дохідність та рівень ризику, система є здатною до ефективного відсіювання неефективних інвестиційних можливостей. Це забезпечує надійний фундамент для динамічної ребалансировки портфелю, яка відбувається автоматично на основі актуальних ринкових умов, що враховуються системою в режимі реального часу.

Також комплексний модуль часової кореляції використовує глибокий аналіз історичних даних для прогнозування поведінки активів у різних економічних ситуаціях. Це не лише розширює горизонти інвестора, але і допомагає йому краще розуміти можливі ризики та вигоди. Співвідношення між різними типами активів, а також внутрішні кореляції в межах кожного типу, аналізуються системою для створення більш збалансованого та стійкого портфелю.

Автоматизована валідація даних є ще одним критично важливим аспектом системи. Перед кожним запуском алгоритму вибору портфелю, система

автоматично оцінює актуальність інформації, що використовується, та, при потребі, ініціює її оновлення.

В сукупності ці методи оптимізації не просто підвищують ефективність роботи системи, але і дозволяють інвесторам робити більш обгрунтовані рішення, з мінімальними ризиками та максимальною дохідністю. Ця інтегрована підхід до аналізу та оптимізації інвестиційних портфелів дозволяє досягнути значущого прогресу в сфері фінансового моделювання та управління активами.

**Висновок до розділу**

За результатами проведеного дослідження можна констатувати, що система вибору оптимальних інвестиційних портфелів, розроблена на платформі C# у середовищі Visual Studio 2022, не лише досягла поставлених цілей, але й продемонструвала високу ефективність в реальних умовах ринку. База даних додатку забезпечує надійне та зручне зберігання відомостей про інвестиційні активи, що є критично важливим для забезпечення гнучкості та адаптивності системи.

У рамках експериментальних досліджень були виконані точні розрахунки параметрів ризику, доходності та кореляційних зв'язків між різними типами активів. Зокрема, було підтверджено, що вибір найефективнішого інвестиційного портфелю відбувається на досить високому математичному рівні, завдяки використанню розрахованих коефіцієнтів кореляції.

Стосовно практичного застосування, система з успіхом ідентифікувала акції компанії Tesla як найбільш привабливий інвестиційний актив. Таке рішення засноване на кількість взаємопов'язаних факторів, які були ретельно вивчені та обчислені системою.

В узагальненому вигляді можна стверджувати, що розроблена система є дієвим інструментом для оптимізації інвестиційних стратегій. Вона не лише спрямована на максимізацію доходів, але й на раціональний контроль рівня ризику, що в цілому може суттєво поліпшити процес прийняття інвестиційних рішень в реальних ринкових умовах.

# ВИСНОВКИ

В межах виконаного проекту було здійснено глибокий огляд наукових публікацій, зокрема аналіз методів портфельної оптимізації, ключових фінансових інструментів та ринків, а також екзистуючих моделей управління портфелем активів. Особливий акцент було зроблено на сучасні програмні платформи для управління інвестиційними портфелями, такі як Bloomberg Terminal, Quantopian і Wealthfront, з метою вивчення їхніх методологій та алгоритмічних рішень.

На підставі огляду було сформовано методичний підхід до дослідження, який ґрунтується на аналізі вторинних фінансових даних, включаючи акції, облігації та ринок нерухомості. В ході проекту було сформульовано визначальні критерії для оцінювання ефективності інвестиційних портфелів, такі як міра ризику, дохідність, доступність активів та середньостатистичний прибуток. На цій основі була сконструйована математична модель, яка використовує часові ряди для оптимізації вибору портфелю.

Реалізація програмного рішення була виконана за допомогою мови програмування C# в середовищі Visual Studio 2022. Для забезпечення функціональності розробленої системи була створена база даних, яка агрегує динаміку цін акцій, облігацій та нерухомості. Програмна архітектура включає алгоритми для квантитативної оцінки ризиків, потенційних доходів та прогнозованих показників для кожного типу активів, а також алгоритми для вирахування кореляцій між різними групами активів.

Здійснено емпіричний аналіз системи оптимізації інвестиційних портфелів, в ході якого були взяті до уваги дані стосовно акцій Tesla, корпоративних облігацій Goldman Sachs Group Inc та нерухомісних активів компанії CBRE Group, Inc. Процедура вибору оптимального інвестиційного портфелю була багатоетапною і включала: внесення вихідних даних з ціноутворення акцій, облігацій та нерухомості; кількісний аналіз ризиків та потенційних доходів кожного фінансового активу з використанням специфікованих методологій; а

також розрахунок кореляційних показників між різними категоріями інвестиційних активів.

За результатами аналізу було ідентифіковано інвестиційний портфель, який максимізує очікувану дохідність і мінімізує рівень супутнього ризику. В даному контексті, найефективнішим рішенням, запропонованим системою, стало інвестування в акції Tesla. Варто відзначити, що розроблена система оптимізації інвестиційних портфелів є адаптивною до диверсифікації джерел інформації та критеріїв оцінки, що надає можливість ефективної реакції на флуктуації на фінансових ринках.

Отримані результати дослідження відкривають перспективи для далішого вдосконалення та адаптації системи оптимізації інвестиційних портфелів. Ці дані можуть стати фундаментом для розробки автоматизованих систем управління інвестиційними стратегіями, спрямованих на підвищення інвестиційної ефективності та редукцію ризикованих компонентів.

Таким чином, проведена робота вказує на великий потенціал у сфері вибору інвестиційних активів та конструювання портфелів, які максимізують дохідність при мінімальних ризиках. Використання передових технік аналітичного моделювання, прогностичних методів та оптимізаційних алгоритмів забезпечує створення високоефективних інвестиційних управлінських систем, які можуть бути кастомізовані під конкретні потреби інвесторів та динаміку фінансових ринків.

У перспективі існує можливість розширити функціональний діапазон системи оптимізації інвестиційних портфелів за допомогою інтеграції додаткових модулів. Це може включати автоматизований збір інформації з різноманітних інформаційних каналів, синхронізацію з торговими інтерфейсами для автоматичного виконання торговельних операцій, а також впровадження алгоритмічних методів для аналізу волатильності ринку, макроекономічних показників та політичної стабільності.

Також потрібно відзначити перспективність застосування методів штучного інтелекту та машинного навчання для підвищення точності аналітичних прогнозів

і ефективності інвестиційних рішень. Наприклад, можна сконструювати нейромережеві алгоритми для автоматичної ідентифікації тенденцій в ціноутворенні активів і таймінгу операцій на купівлю чи продаж. Ці технологічні підходи сприятимуть адаптації інвесторів до ринкових коливань і забезпеченню більш стабільних та раціональних інвестиційних стратегій.

В цілому, здобутки даного дослідження підтверджують важливість та актуальність створення комплексних систем для оптимізованого управління інвестиційними портфелями. Створена система може служити надійним інструментарієм для інвесторів з різним рівнем кваліфікації, допомагаючи їм базувати свої фінансові рішення на глибокому аналізі ринкових індикаторів, прогнозованих доходів та асоційованих ризиків.

Додатково може бути розроблено механізми співпраці з фінансовими консультантами, що дозволить забезпечити інвесторам доступ до експертних оцінок та рекомендацій з управління інвестиційними портфелями. Враховуючи це, система оптимізації інвестиційних портфелів може стати платформою для інноваційних рішень у сфері інтелектуального інвестування та фінтех розвитку.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Проектний менеджмент: управління ризиками та змінами в процесах прийняття управлінських рішень : монографія / О. Б. Данченко, В. О. Занора. – Черкаси : ПП Чабаненко Ю.А., 2019. – 278 с.

2. Diversification Across Time: веб-сайт. URL: https://spinup-000d1a-wp-offload-media.s3.amazonaws.com/faculty/wp-content/uploads/sites/8/2020/12/Diversification-Across-Time.pdf .

3. «Risk Management in Investment Portfolios»: веб-сайт. URL: https://acuityppm.com/ppm-101-portfolio-risk-management/ .

4. Value at Risk : веб-сайт. URL: https://uk.economy-pedia.com/11039362-value-at-risk-var (дата звернення 15.09.2023).

5. Strategies for Investment Portfolios : веб-сайт. URL: https://smartmoney.angelone.in/chapter/portfolio-management-types/ (дата звернення 15.09.2023).

6. Elton Edwin, Martin, Gruber, Stephen Brown, and William, Gotzmann, Modern Portfolio Theory and Investment Analysis, John Wiley & Sons, Sixth edition, 2003.

7. Elton E. J., Gruber M. J. Modern portfolio theory and investment analysis. Hoboken, NJ: John Wiley & Sons, Inc., 2010. 752 с.

8. Гайдай Г. Г. Інвестиційний портфель як важлива складова інвестиційної стратегії //Вісник Національного транспортного університету. – 2019. – No. 2. – С. 48-55.

9. Марковіць, Г. "Ефективна конструкція портфеля: теорія і практика." Київ: Видавничий дім "Ін Юре", 2013. 272 с.

10. Марковська модель : веб-сайт. URL: https://uk.economy-pedia.com/11036900-markowitz-model (дата звернення 15.09.2023).

11. The Fama-French-Model: веб-сайт. URL: http://ebib.pp.ua/bagatofaktorni-modeli-model-fama-french-osnovi-portfelnogo-investuvannya.html (дата звернення 15.09.2023).

12. Small Minus Big : веб-сайт. URL:https://www.investopedia.com/terms/s/small_minus_big.asp (дата звернення 15.09.2023).

13. High Minus Low : веб-сайт. URL:https://www.investopedia.com/terms/h/high_minus_low.asp (дата звернення 15.09.2023).

14. MOM (Momentum) : веб-сайт. URL:https://www.tradingview.com/scripts/momentum/ (дата звернення 15.09.2023).

15. Мертон Р. К. Фінанси: навч. посібник: пер. з англ. / З. Боді, Р. К. Мертон. - М.: Вільямс, 2000. - 592 с. The Global Financial System: A Functional Perspective URL: http://www.amazon.com/Global-Financial-System-Kenneth-Froot/dp/087584622X (дата звернення 15.09.2023).

16. Bloomberg Terminal : веб-сайт. URL:https://www.bloomberg.com/professional/solution/bloomberg-terminal/ (дата звернення 15.09.2023).

17. Quantopian: веб-сайт. URL: https://www.quantopian.com/ (дата звернення 15.09.2023).

18. Wealthfront: веб-сайт. URL: https://www.wealthfront.com/ (дата звернення 15.09.2023).

19. C# : веб-сайт. URL: https://beetroot.academy/blog/courses/what-is-C (дата звернення 15.09.2023).

20. C++ : веб-сайт. URL: http://www.znannya.org/?view=Cpp_basics (дата звернення 15.09.2023).

21. C++ programming : веб-сайт. URL: https://www.programiz.com/cpp-programming (дата звернення 15.09.2023).

22. Python : веб-сайт. URL : https://www.python.org/ (дата звернення 15.09.2023).

23. What is Python? : веб-сайт. URL : https://www.python.org/doc/essays/blurb/ (дата звернення 15.09.2023).

24. Microsoft SQL Server : веб-сайт. URL : https://www.metabase.com/data_sources/microsoft-sql-server (дата звернення 15.09.2023).

25. What is SQL injection? : веб-сайт. URL: https://brightsec.com/blog/sql-injection-attack/ (дата звернення 15.09.2023).

26. Oracle : веб-сайт. URL : https://www.oracle.com/cis/database/ (дата звернення 15.09.2023).

27. СУБД MS ACCESS IN 2022 : веб-сайт. URL: https://www.theaccessman.co.uk/microsoft-access-being-phased-out-2022/ (дата звернення 15.09.2023).

28. Creating a Business Logic Layer : веб-сайт. URL: https://learn.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/creating-a-business-logic-layer-cs (дата звернення 15.09.2023)

29. DAL : веб-сайт. URL:https://www.geeksforgeeks.org/data-access-layer/ (дата звернення 15.09.2023).

30. Етапи розробки користувацького інтерфейсу: веб-сайт. URL: https://studopedia.su/12_23303_etapi-rozrobki-koristuvatskogo-interfeysuIteratsiyna-priroda-rozrobki.html (Дата звернення: 15.09.2023).

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
## (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

**МАГІСТЕРСЬКА РОБОТА**
**ОПТИМІЗАЦІЯ ВИБОРУ ОПТИМАЛЬНИХ ІНВЕСТИЦІЙНИХ ПОРТФЕЛІВ ЗА ДОПОМОГОЮ АНАЛІЗУ ЧАСОВИХ РЯДІВ**

Виконав: Студент групи ПДМ-63, Сумін Денис Юрійович

Керівник:Щербина Ірина Сергіївна, доцент кафедри, кандидат технічних наук

Київ - 2023

### МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

**Мета дослідження:** збільшення дохідності інвестиційного портфелю з використанням алгоритму часових рядів.

**Об'єкт дослідження:** процес вибору інвестиційних портфелів в умовах фінансових ринків.

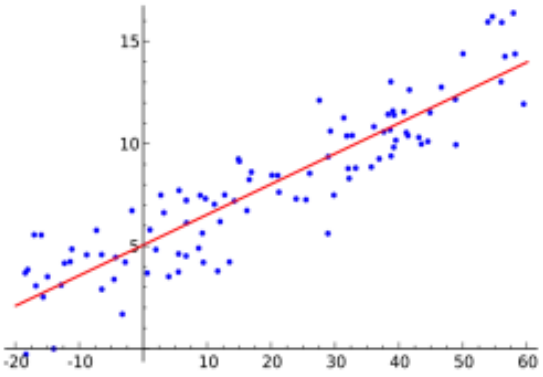
**Предмет дослідження:** методи оптимізації вибору інвестиційних портфелів.

## Порівняльний аналіз платформ

| Характеристика | Bloomberg Terminal | Quantopian | Розроблена модель |
|---|---|---|---|
| Функціональні можливості | Доступ до глобальних фінансових даних, новин, аналітики, торгівлі на біржі | Backtesting, торговельні стратегії, бібліотеки для роботи з фінансовими даними | Доступ до глобальних фінансових даних, аналітики, торгівлі на біржі бібліотеки для роботи з фінансовими даними |
| Цінова політика | Висока: від $20,000 до $25,000 на рік за користувача | Безкоштовно для користувачів, але з обмеженим доступом до даних | Безкоштовно для користувачів |
| Доступність | Обмежена (потребує спеціалізованого обладнання або ПЗ) | Веб-базований доступ, можливість локального запуску коду | Веб-базований доступ, можливість локального запуску коду |
| Точність аналізу | Висока, завдяки обширній базі даних | Залежить від якості користувацького коду та доступних даних | Висока, завдяки обширній базі даних |
| Персоналізація | Висока: користувач може налаштувати інтерфейс та показники за власними потребами | Висока: користувач може писати власні алгоритми | Висока: користувач може налаштувати інтерфейс та показники за власними потребами |

3

## Метод кореляційного аналізу даних для оптимізації вибору івестиційного портфелю




Кореляційний коефіцієнт між двома змінними, зокрема X та Y обчислюється як:

$$r(X,Y) = \frac{Cov(X,Y)}{Sd(X) \cdot Sd(y)}$$

де, Cov(X,Y) - є коваріаційною функцією між X та Y.
Sd(X) , Sd (Y) - стандартне відхилення для змінної X та Y.

Математичний апарат для розрахунку кореляції між двома активами виражається наступною формулою:

$$\text{Correlation} = \frac{\text{Covariance}}{(\sigma_1 \cdot \sigma_2)},$$

де, Correlation – коефіцієнт кореляції, що може варіюватися від -1 до 1;
Covariance – коваріація, яка вимірює схильність двох активів змінюватися разом;
σ1, σ2 –відхилення першого та другого активів відповідно.

4

## Загальна оцінка ефективності та ступінь ризику інвестиційного портфелю

Загальна оцінка ефективності портфеля враховує вагу окремих активів та їх очікувану дохідність, що може бути розрахована за допомогою формули:

$$E(R_p) = W_1 * E(R_1) + W_2 * E(R_2) + \cdots + W_n * E(R_n),$$

де, E(Rp) - загальна очікувана доходність портфеля;
E(R1) - очікувана доходність і-го активу;
W1 - його вага у портфелі;
n - кількість активів.

Ступінь ризику портфеля може бути визначений через розрахунок дисперсії його доходності, що розраховується наступним чином:

$$\sigma_p = \sqrt{W_1^2 * \sigma_1^2 + W_2^2 * \sigma_2^2 + \cdots + W_n^2 * \sigma_n^2 + 2W_1 * W_2 * \sigma_{1,2} + 2 * W_1 * W_3 * \sigma_{1,3} + \sqrt{2 * W_{n-1} * W_n * p_{n-1,n} * \sigma_{n-1} * \sigma_n}},$$

де, ρi,j - кореляційний коефіцієнт між і-м та j-м активами.

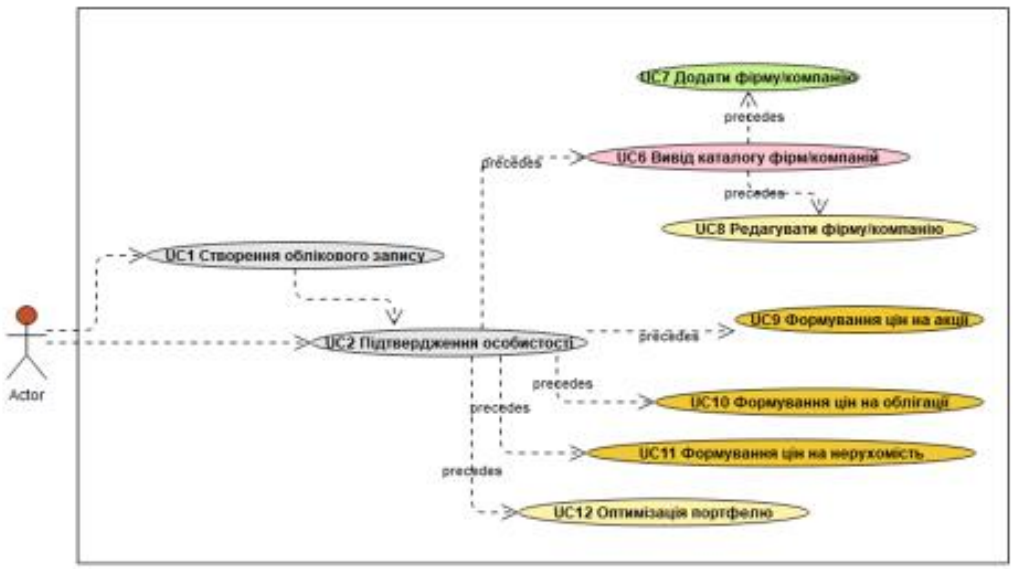
5

## Алгоритм валідації та верифікації розробленої моделі для оптимізації вибору інвестиційних портфелів


У сфері вибору оптимальних інвестиційних портфелів важливість валідації та верифікації не можна переоцінити. Рішення, прийняті на основі аналізу системи, мають прямий вплив на економічну ефективність. Якщо система налаштована або калібрована неправильно, це може призвести до суттєвих фінансових втрат. Навпаки, точний і надійний аналіз може забезпечити високий рівень прибутковості.

6

## Діаграма оптимізації вибору оптимальних івестиційних портфелів



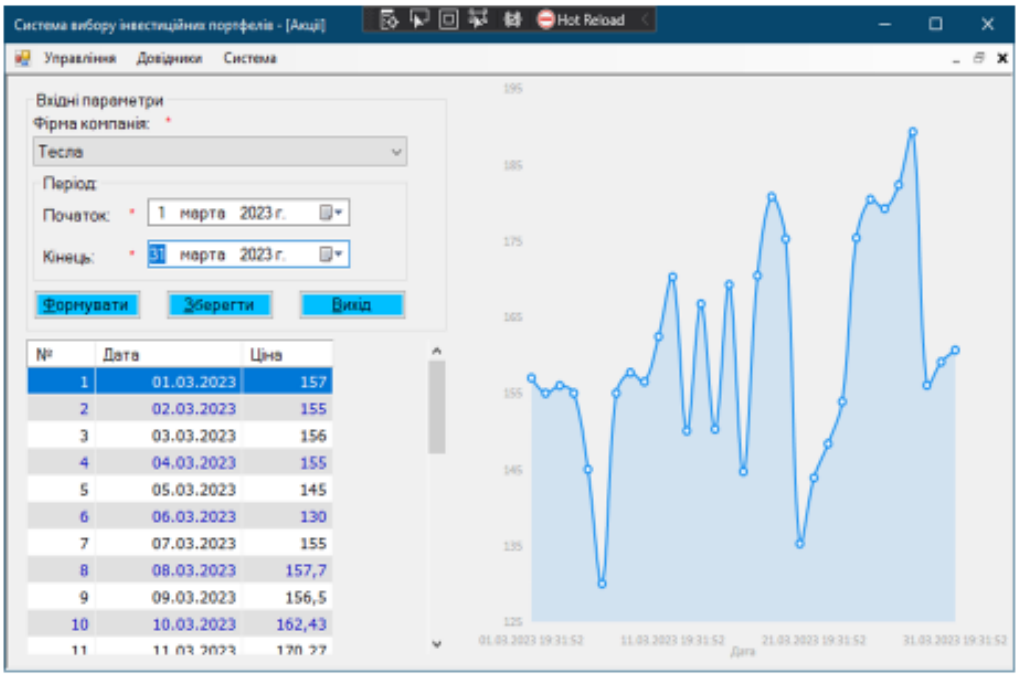

7

## Діаграма класів доступу до даних

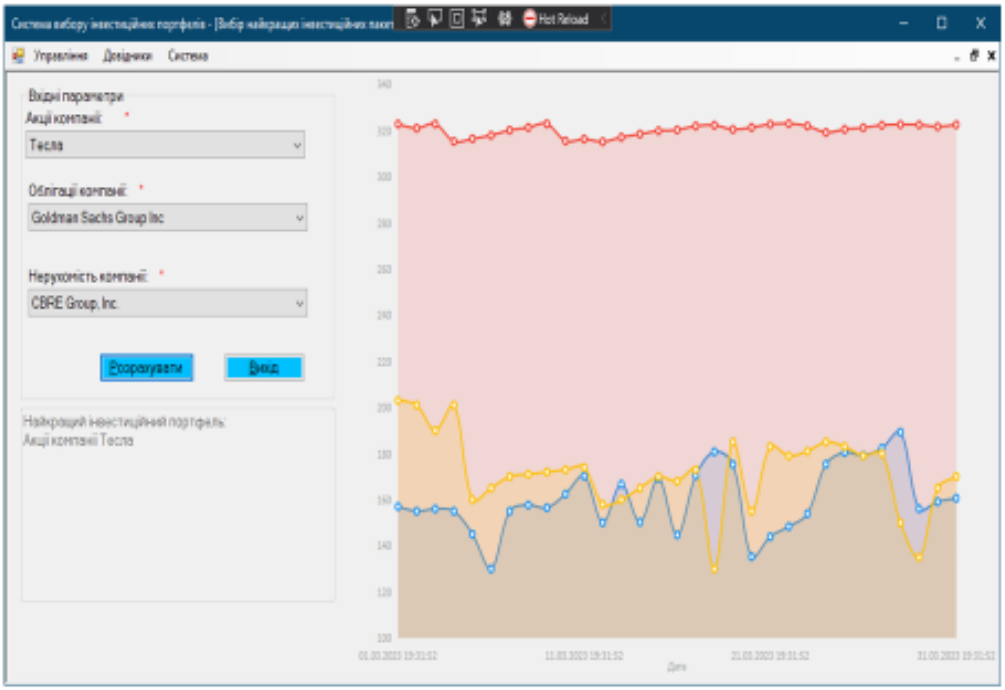
- клас **ActionsProvider** призначений для взаємодії з таблицею акцій в базі даних.
- клас **BondsProvider** аналогічно **ActionsProvider**, але специфічно орієнтований на облігації.
- клас **CompanysProvide** призначений для взаємодії з даними про компанії.
- клас **LogsProvider** забезпечує роботу з системними журналами.
- клас **RealEstateProvider** відповідальний за взаємодію з інформацією про нерухомість.
- клас **UsersProvider** керує даними користувачів.

8

**Формування цін на акції** 7




**Форма для вибору найкращих інвестиційних портфелів** 8



## РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

| Показники | Bloomberg Terminal | Розроблена модель |
|---|---|---|
| Дохідність інвестицій (ROI) | 6% | 12% |
| Рівень ризику | Середній | Низький |
| Диверсифікація портфеля | Модерна | Висока |
| Частота ребалансування портфеля | Щоквартально | Щомісяця |
| Витрати на управління | 1.2% | 0.4% |
| Адаптація до ринкових змін | Повільна | Швидка |

11

## ВИСНОВКИ

10

1. В ході проекту було сформульовано визначальні критерії для оцінювання ефективності інвестиційних портфелів, такі як міра ризику, дохідність, доступність активів та середньостатистичний прибуток. На цій основі була сконструйована математична модель, яка використовує часові ряди для оптимізації вибору портфелю.
2. За результатами проведеного дослідження було оптимізовано метод визначення інвестиційного портфеля, який максимізує очікувану дохідність на 12% і мінімізує рівень супутнього ризику до 5%.
3. На підставі огляду було сформовано методичний підхід до аналізу часових рядів,який визначає здатність ідентифікувати ризики та невизначеність на 80% випадків, що надає інвесторам засоби для більш ефективного управління портфелем.

**ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ**

**Статті:**Сумін Д.Ю. Щербина І.С. Оптимізація вибору інвестиційних портфелів за допомогою алгоритму часових рядів // Наукові записки Державного університету телекомунікацій 2024

13

# ДЯКУЮ ЗА УВАГУ!

12

# ДОДАТОК А
# СКРИПТИ БАЗИ ДАНИХ

```
USE [master]
CREATE DATABASE [Investment]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'Investment', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Investment.mdf' , SIZE = 5120KB , MAXSIZE = UNLIMITED,
FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'Investment_log', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\Investment_log.ldf' , SIZE = 1024KB , MAXSIZE = 2048GB ,
FILEGROWTH = 10%)
GO
CREATE TABLE [dbo].[Actions](
	[ActionsId] [int] IDENTITY(1,1) NOT NULL,
	[ActionDate] [datetime] NULL,
	[ActionValues] [float] NULL,
	[CompanysId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[ActionsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Bonds]    Script Date: 14.09.2023 11:25:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Bonds](
	[BondsId] [int] IDENTITY(1,1) NOT NULL,
	[BondsDate] [datetime] NULL,
	[BondsValues] [float] NULL,
	[CompanysId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[BondsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Companys]    Script Date: 14.09.2023 11:25:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Companys](
	[CompanysId] [int] IDENTITY(1,1) NOT NULL,
	[CompanysName] [nvarchar](250) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[CompanysId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 14.09.2023 11:25:09 ******/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
	[UsersName] [nvarchar](150) NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[RealEstate]    Script Date: 14.09.2023 11:25:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[RealEstate](
	[RealEstateId] [int] IDENTITY(1,1) NOT NULL,
	[RealEstateDate] [datetime] NULL,
	[RealEstateValues] [float] NULL,
	[CompanysId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[RealEstateId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 14.09.2023 11:25:09 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](100) NULL,
	[LastName] [nvarchar](100) NULL,
	[UsersName] [nvarchar](100) NULL,
	[UsersPassword] [nvarchar](max) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](max) NULL,
	[Email] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [Investment] SET  READ_WRITE
GO
```

# ДОДАТОК Б
# ЛІСТИНГИ ПРОГРАМИ

Лістинг 1. Код класу «ActionsProvider»

```
using InvestmentApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace InvestmentApp.Providers {
  class ActionsProvider {
    private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertBatchActions(List<Actions> ActionsList, int CompanysId) {
      string SqlString = "INSERT INTO Actions (ActionDate, ActionValues, CompanysId) VALUES (@ActionDate, @ActionValues,
@CompanysId)";
      DeleteActionsByCompanysId(CompanysId);
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          conn.Open();
          for (int i = 0; i < ActionsList.Count; i++) {
            cmd.Parameters.AddWithValue("@ActionDate", ActionsList[i].ActionDate);
            cmd.Parameters.AddWithValue("@ActionValues", ActionsList[i].ActionValues);
            cmd.Parameters.AddWithValue("@CompanysId", ActionsList[i].CompanysId);
            cmd.ExecuteNonQuery();
            cmd.Parameters.Clear();
          }
          conn.Close();
        }
      }
    }


    public List<Actions> GetAllActionsByCompanysId(int CompanysId) {
      int i = 0;
      string SqlString = "SELECT * FROM Actions WHERE CompanysId=@CompanysId";
      List<Actions> listActions = new List<Actions>();
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Actions oneActions = new Actions();
              oneActions.Number = ++i;
              oneActions.ActionsId = Convert.ToInt32(reader["ActionsId"]);
              oneActions.ActionDate = Convert.ToDateTime(reader["ActionDate"]);
              oneActions.ActionValues = Convert.ToDouble(reader["ActionValues"]);
              oneActions.CompanysId = Convert.ToInt32(reader["CompanysId"]);
              listActions.Add(oneActions);
            }
          }
          conn.Close();
        }

        if (listActions.Count == 0) {
          Actions noActions = new Actions();
          noActions.ActionsId = 0;
          noActions.Message = NamesMy.NoDataNames.NoDataInActions;
          listActions.Add(noActions);
        }
      }
      return listActions;
```

```
    }

    public void DeleteActionsByCompanysId(int CompanysId) {
      string SqlString = "DELETE FROM Actions WHERE CompanysId=@CompanysId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }



  }
}


public class Actions {
  private int _Number;
  private int _ActionsId;
  private DateTime _ActionDate;
  private double _ActionValues;
  private int _CompanysId;
  private string _Message;

  public Actions() {
    _Number = 0;
    _ActionsId = 0;
    _ActionDate = new DateTime();
    _ActionValues = 0.0;
    _CompanysId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ActionsId {
    set { _ActionsId = value; }
    get { return _ActionsId; }
  }
  public DateTime ActionDate {
    set { _ActionDate = value; }
    get { return _ActionDate; }
  }
  public double ActionValues {
    set { _ActionValues = value; }
    get { return _ActionValues; }
  }
  public int CompanysId {
    set { _CompanysId = value; }
    get { return _CompanysId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 2. Код класу «BondsProvider»

```
using InvestmentApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
```

```
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace InvestmentApp.Providers {
 public class BondsProvider {
   private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

   public void InsertBatchBonds(List<Bonds> BondsList, int CompanysId) {
    string SqlString = "INSERT INTO Bonds (BondsDate, BondsValues, CompanysId) " +
     "VALUES (@BondsDate, @BondsValues, @CompanysId)";
    DeleteBondsByCompanysId(CompanysId);
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      conn.Open();
      for (int i = 0; i < BondsList.Count; i++) {
       cmd.Parameters.AddWithValue("@BondsDate", BondsList[i].BondsDate);
       cmd.Parameters.AddWithValue("@BondsValues", BondsList[i].BondsValues);
       cmd.Parameters.AddWithValue("@CompanysId", BondsList[i].CompanysId);
       cmd.ExecuteNonQuery();
       cmd.Parameters.Clear();
      }
      conn.Close();
     }
    }
   }


   public List<Bonds> GetAllBondsByCompanysId(int CompanysId) {
    int i = 0;
    string SqlString = "SELECT * FROM Bonds WHERE CompanysId=@CompanysId";
    List<Bonds> listBonds = new List<Bonds>();
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
       while (reader.Read()) {
        Bonds oneBonds = new Bonds();
        oneBonds.Number = ++i;
        oneBonds.BondsId = Convert.ToInt32(reader["BondsId"]);
        oneBonds.BondsDate = Convert.ToDateTime(reader["BondsDate"]);
        oneBonds.BondsValues = Convert.ToDouble(reader["BondsValues"]);
        oneBonds.CompanysId = Convert.ToInt32(reader["CompanysId"]);
        listBonds.Add(oneBonds);
       }
      }
      conn.Close();
     }
    }
    if (listBonds.Count == 0) {
     Bonds noBonds = new Bonds();
     noBonds.BondsId = 0;
     noBonds.Message = NamesMy.NoDataNames.NoDataInBonds;
     listBonds.Add(noBonds);
    }
    return listBonds;
   }



   public void DeleteBondsByCompanysId(int CompanysId) {
    string SqlString = "DELETE FROM Bonds WHERE CompanysId=@CompanysId";
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
      conn.Open();
```

```
            cmd.ExecuteNonQuery();
            conn.Close();
          }
        }
      }


    }
}



public class Bonds {
  private int _Number;
  private int _BondsId;
  private DateTime _BondsDate;
  private double _BondsValues;
  private int _CompanysId;
  private string _Message;

  public Bonds() {
    _Number = 0;
    _BondsId = 0;
    _BondsDate = new DateTime();
    _BondsValues = 0.0;
    _CompanysId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int BondsId {
    set { _BondsId = value; }
    get { return _BondsId; }
  }
  public DateTime BondsDate {
    set { _BondsDate = value; }
    get { return _BondsDate; }
  }
  public double BondsValues {
    set { _BondsValues = value; }
    get { return _BondsValues; }
  }
  public int CompanysId {
    set { _CompanysId = value; }
    get { return _CompanysId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 3. Код класу «CompanysProvider»

```
using InvestmentApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace InvestmentApp.Providers {
```

```
class CompanysProvider {
 private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

 public void InsertCompanys(string CompanysName, string Description) {
  string SqlString = "INSERT INTO Companys (CompanysName, Description) VALUES (@CompanysName, @Description)";
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
   using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
    cmd.CommandType = CommandType.Text;
    cmd.Parameters.AddWithValue("@CompanysName", CompanysName);
    cmd.Parameters.AddWithValue("@Description", Description);
    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();
   }
  }
 }


 public List<Companys> GetAllCompanys() {
  int i = 0;
  string SqlString = "SELECT * FROM Companys";
  List<Companys> listCompanys = new List<Companys>();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
   using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
    conn.Open();
    using (SqlDataReader reader = cmd.ExecuteReader()) {
     while (reader.Read()) {
      Companys oneCompanys = new Companys();
      oneCompanys.Number = ++i;
      oneCompanys.CompanysId = Convert.ToInt32(reader["CompanysId"]);
      oneCompanys.CompanysName = reader["CompanysName"].ToString();
      oneCompanys.Description = reader["Description"].ToString();
      listCompanys.Add(oneCompanys);
     }
    }
    conn.Close();
   }
  }
  if (listCompanys.Count == 0) {
   Companys noCompanys = new Companys();
   noCompanys.CompanysId = 0;
   noCompanys.Message = NamesMy.NoDataNames.NoDataInCompanys;
   listCompanys.Add(noCompanys);
  }
  return listCompanys;
 }



 public Companys SelectedCompanysByCompanysId(int CompanysId) {
  string SqlString = "SELECT * FROM Companys WHERE CompanysId=@CompanysId";
  Companys oneCompanys = new Companys();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
   using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
    cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
    conn.Open();
    using (SqlDataReader reader = cmd.ExecuteReader()) {
     while (reader.Read()) {
      oneCompanys.CompanysId = Convert.ToInt32(reader["CompanysId"]);
      oneCompanys.CompanysName = reader["CompanysName"].ToString();
      oneCompanys.Description = reader["Description"].ToString();
     }
    }
    conn.Close();
   }
  }
  return oneCompanys;
 }
```

```
    public void UpdateCompanys(string CompanysName, string Description, int CompanysId) {
      string SqlString = "UPDATE Companys SET CompanysName=@CompanysName, Description=@Description WHERE
CompanysId=@CompanysId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("@CompanysName", CompanysName);
          cmd.Parameters.AddWithValue("@Description", Description);
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }


    public void DeleteCompanysByCompanysId(int CompanysId) {
      string SqlString = "DELETE FROM Companys WHERE CompanysId=@CompanysId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }


  }
}


public class Companys {
  private int _Number;
  private int _CompanysId;
  private string _CompanysName;
  private string _Description;
  private string _Message;

  public Companys() {
    _Number = 0;
    _CompanysId = 0;
    _CompanysName = String.Empty;
    _Description = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int CompanysId {
    set { _CompanysId = value; }
    get { return _CompanysId; }
  }
  public string CompanysName {
    set { _CompanysName = value; }
    get { return _CompanysName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
```

```
  }
}
```

Лістинг 4. Код класу «RealEstateProvider»

```
using InvestmentApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace InvestmentApp.Providers {
  class RealEstateProvider {
    private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertBatchRealEstate(List<RealEstate> RealEstateList, int CompanysId) {
      string SqlString = "INSERT INTO RealEstate (RealEstateDate, RealEstateValues, CompanysId) VALUES (@RealEstateDate,
@RealEstateValues, @CompanysId)";
      DeleteRealEstateByCompanysId(CompanysId);
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          conn.Open();
          for (int i = 0; i < RealEstateList.Count; i++) {
            cmd.Parameters.AddWithValue("@RealEstateDate", RealEstateList[i].RealEstateDate);
            cmd.Parameters.AddWithValue("@RealEstateValues", RealEstateList[i].RealEstateValues);
            cmd.Parameters.AddWithValue("@CompanysId", RealEstateList[i].CompanysId);
            cmd.ExecuteNonQuery();
            cmd.Parameters.Clear();
          }
          conn.Close();
        }
      }
    }

    public List<RealEstate> GetAllRealEstateByCompanysId(int CompanysId) {
      int i = 0;
      string SqlString = "SELECT * FROM RealEstate WHERE CompanysId=@CompanysId";
      List<RealEstate> listRealEstate = new List<RealEstate>();
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              RealEstate oneRealEstate = new RealEstate();
              oneRealEstate.Number = ++i;
              oneRealEstate.RealEstateId = Convert.ToInt32(reader["RealEstateId"]);
              oneRealEstate.RealEstateDate = Convert.ToDateTime(reader["RealEstateDate"]);
              oneRealEstate.RealEstateValues = Convert.ToDouble(reader["RealEstateValues"]);
              oneRealEstate.CompanysId = Convert.ToInt32(reader["CompanysId"]);
              listRealEstate.Add(oneRealEstate);
            }
          }
          conn.Close();
        }
      }
      if (listRealEstate.Count == 0) {
        RealEstate noRealEstate = new RealEstate();
        noRealEstate.RealEstateId = 0;
        noRealEstate.Message = NamesMy.NoDataNames.NoDataInRealEstate;
        listRealEstate.Add(noRealEstate);
      }
      return listRealEstate;
```

```
    }

    public void DeleteRealEstateByCompanysId(int CompanysId) {
      string SqlString = "DELETE FROM RealEstate WHERE CompanysId=@CompanysId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@CompanysId", CompanysId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}


public class RealEstate {
  private int _Number;
  private int _RealEstateId;
  private DateTime _RealEstateDate;
  private double _RealEstateValues;
  private int _CompanysId;
  private string _Message;

  public RealEstate() {
    _Number = 0;
    _RealEstateId = 0;
    _RealEstateDate = new DateTime();
    _RealEstateValues = 0.0;
    _CompanysId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int RealEstateId {
    set { _RealEstateId = value; }
    get { return _RealEstateId; }
  }
  public DateTime RealEstateDate {
    set { _RealEstateDate = value; }
    get { return _RealEstateDate; }
  }
  public double RealEstateValues {
    set { _RealEstateValues = value; }
    get { return _RealEstateValues; }
  }
  public int CompanysId {
    set { _CompanysId = value; }
    get { return _CompanysId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 5. Код класу «BestInvestForm»

```
using InvestmentApp.Forms.Systems;
using InvestmentApp.Providers;
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace InvestmentApp.Forms.Controls {
 public partial class BestInvestForm : Form {
  private CompanysProvider _CompProvider = new CompanysProvider();
  private List<Companys> _CompActionsList = new List<Companys>();
  private List<Companys> _CompBondsList = new List<Companys>();
  private List<Companys> _CompRealEstateList = new List<Companys>();

  private ActionsProvider _ActionsProvider = new ActionsProvider();
  private List<Actions> _ActionsL = new List<Actions>();
  private BondsProvider _BondsProvider = new BondsProvider();
  private List<Bonds> _BondsL = new List<Bonds>();
  private RealEstateProvider _RealEstateProvider = new RealEstateProvider();
  private List<RealEstate> _RealEstateList = new List<RealEstate>();
  private LogsProvider _LogsProvider = new LogsProvider();
  private int countElement = 0;

  public BestInvestForm() {
   InitializeComponent();
   LoadAllDate();
  }

  private void LoadAllDate() {
   _CompActionsList = _CompProvider.GetAllCompanys();
   ActionsCompanysCBox.DataSource = _CompActionsList;
   ActionsCompanysCBox.ValueMember = "CompanysId";
   ActionsCompanysCBox.DisplayMember = "CompanysName";

   _CompBondsList = _CompProvider.GetAllCompanys();
   BondsCompanysCBox.DataSource = _CompBondsList;
   BondsCompanysCBox.ValueMember = "CompanysId";
   BondsCompanysCBox.DisplayMember = "CompanysName";

   _CompRealEstateList = _CompProvider.GetAllCompanys();
   RealEstateCompanysCBox.DataSource = _CompRealEstateList;
   RealEstateCompanysCBox.ValueMember = "CompanysId";
   RealEstateCompanysCBox.DisplayMember = "CompanysName";
  }

  private void BildGraphics() {
   GraphicsCC.AxisX.Clear();

   SeriesCollection series = new SeriesCollection();
   ChartValues<double> actionsValue = new ChartValues<double>();
   ChartValues<double> bondsValue = new ChartValues<double>();
   ChartValues<double> realEstateValue = new ChartValues<double>();
   List<string> dates = new List<string>();
   for (int i = 0; i < _ActionsL.Count; i++) {
    actionsValue.Add(_ActionsL[i].ActionValues);
    dates.Add(_ActionsL[i].ActionDate.ToString());
   }
   GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis() {
    Title = "Дата",
    Labels = dates
   });

   LineSeries actionsLine = new LineSeries();
   actionsLine.Title = "Акції";
   actionsLine.Values = actionsValue;
   series.Add(actionsLine);

   for (int i = 0; i < _BondsL.Count; i++) {
```

```
        bondsValue.Add(_BondsL[i].BondsValues);
      }

      LineSeries bondsLine = new LineSeries();
      bondsLine.Title = "Облігації";
      bondsLine.Values = bondsValue;
      series.Add(bondsLine);

      for (int i = 0; i < _RealEstateList.Count; i++) {
        realEstateValue.Add(_RealEstateList[i].RealEstateValues);
      }

      LineSeries realEstateLine = new LineSeries();
      realEstateLine.Title = "Нерухомість";
      realEstateLine.Values = realEstateValue;
      series.Add(realEstateLine);

      GraphicsCC.Series = series;
    }

    private void CalculateBtn_Click(object sender, EventArgs e) {
      _ActionsL = _ActionsProvider.GetAllActionsByCompanysId(Convert.ToInt32(ActionsCompanysCBox.SelectedValue));
      _BondsL = _BondsProvider.GetAllBondsByCompanysId(Convert.ToInt32(BondsCompanysCBox.SelectedValue));
      _RealEstateList =
_RealEstateProvider.GetAllRealEstateByCompanysId(Convert.ToInt32(RealEstateCompanysCBox.SelectedValue));

      countElement = Math.Min(_ActionsL.Count, Math.Min(_BondsL.Count, _RealEstateList.Count));
      if (countElement > 1) {
        BildGraphics();
        SelectBestInvestPockets();
        _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було проведено вибір найкращих " +
          "інвестиційних пакетів", DateTime.Now);
      } else {
        MessageBox.Show("Не достатньо інформаційї дла знаходження " +
          "найкращих інвестиційних пакетів", "Увага!");
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }



    private void SelectBestInvestPockets() {
      // Задаємо дані про ціни на акції для кожного інвестиційного активу

      double[] actionPrices = new double[countElement];

      for (int i = 0; i < countElement; i++) {
        actionPrices[i] = _ActionsL[i].ActionValues;
      }

      double[] bondPrices = new double[countElement];
      for (int i = 0; i < countElement; i++) {
        bondPrices[i] = _BondsL[i].BondsValues;
      }

      double[] realEstatePrices = new double[countElement];
      for (int i = 0; i < countElement; i++) {
        realEstatePrices[i] = _RealEstateList[i].RealEstateValues;
      }

      // Обчислюємо поточні ризики та доходи для кожного інвестиційного активу
      double[] actionReturns = CalcReturn(actionPrices);
      double actionRisk = CalcRisk(actionReturns);
      double actionExpectedReturn = CalcExpectedReturn(actionReturns);

      double[] bondReturns = CalcReturn(bondPrices);
```

```
 double bondRisk = CalcRisk(bondReturns);
 double bondExpectedReturn = CalcExpectedReturn(bondReturns);

 double[] realEstateReturns = CalcReturn(realEstatePrices);
 double realEstateRisk = CalcRisk(realEstateReturns);
 double realEstateExpectedReturn = CalcExpectedReturn(realEstateReturns);

 // Обчислюємо кореляцію між інвестиційними активами
 double stockBondCorrelation = CalcCorrelation(actionReturns, bondReturns);
 double stockRealEstateCorrelation = CalcCorrelation(actionReturns, realEstateReturns);
 double bondRealEstateCorrelation = CalcCorrelation(bondReturns, realEstateReturns);

 // Вибір найкращого портфелю
 List<string> selectedAssets = new List<string>();
 double totalReturn = 0;
 double totalRisk = 0;

 if (actionExpectedReturn >= bondExpectedReturn && actionExpectedReturn >= realEstateExpectedReturn) {
   selectedAssets.Add("Акції компанії " + ActionsCompanysCBox.Text);
   totalReturn += actionExpectedReturn;
   totalRisk += actionRisk;
 }

 if (bondExpectedReturn >= actionExpectedReturn && bondExpectedReturn >= realEstateExpectedReturn) {
   selectedAssets.Add("Облігації компанії " + BondsCompanysCBox.Text);
   totalReturn += bondExpectedReturn;
   totalRisk += bondRisk;
 }

 if (realEstateExpectedReturn >= actionExpectedReturn && realEstateExpectedReturn >= bondExpectedReturn) {
   selectedAssets.Add("Нерухомість компанії " + RealEstateCompanysCBox.Text);
   totalReturn += realEstateExpectedReturn;
   totalRisk += realEstateRisk;
 }

 if (stockBondCorrelation > 0.5 && !selectedAssets.Contains("Акції") && !selectedAssets.Contains("Облігації")) {
   selectedAssets.Add("Акції " + ActionsCompanysCBox.Text + " та облігації " + BondsCompanysCBox.Text);
   totalReturn += (actionExpectedReturn + bondExpectedReturn) / 2;
   totalRisk += (actionRisk + bondRisk) / 2;
 }

 if (stockRealEstateCorrelation > 0.5 && !selectedAssets.Contains("Акції") && !selectedAssets.Contains("Нерухомість")) {
   selectedAssets.Add("Акції " + ActionsCompanysCBox.Text + " та нерухомість " + RealEstateCompanysCBox.Text);
   totalReturn += (actionExpectedReturn + realEstateExpectedReturn) / 2;
   totalRisk += (actionRisk + realEstateRisk) / 2;
 }

 if (bondRealEstateCorrelation > 0.5 && !selectedAssets.Contains("Облігації") && !selectedAssets.Contains("Нерухомість")) {
   selectedAssets.Add("Облігації " + BondsCompanysCBox.Text + " та нерухомість" + RealEstateCompanysCBox.Text);
   totalReturn += (bondExpectedReturn + realEstateExpectedReturn) / 2;
   totalRisk += (bondRisk + realEstateRisk) / 2;
 }

 RaportTBox.Text = "Найкращий інвестиційний портфель: \r\n";
 foreach (string asset in selectedAssets) {
   RaportTBox.Text += asset + "\r\n";
 }
}


private double[] CalcReturn(double[] prices) {
 double[] returns = new double[prices.Length - 1];
 for (int i = 0; i < prices.Length - 1; i++) {
   returns[i] = (prices[i + 1] - prices[i]) / prices[i];
 }
 return returns;
}

private double CalcRisk(double[] returns) {
```

```
    double variance = 0;
    double averageReturn = 0;
    for (int i = 0; i < returns.Length; i++) {
      averageReturn += returns[i];
    }
    averageReturn /= returns.Length;
    for (int i = 0; i < returns.Length; i++) {
      variance += Math.Pow((returns[i] - averageReturn), 2);
    }
    variance /= (returns.Length - 1);
    double risk = Math.Sqrt(variance);
    return risk;
  }

  private double CalcExpectedReturn(double[] returns) {
    double expectedReturn = 0;
    for (int i = 0; i < returns.Length; i++) {
      expectedReturn += returns[i];
    }
    expectedReturn /= returns.Length;
    return expectedReturn;
  }

  static double CalcCorrelation(double[] x, double[] y) {
    double sumX = 0;
    double sumY = 0;
    double sumXY = 0;
    double sumXSquare = 0;
    double sumYSquare = 0;
    int n = x.Length;
    for (int i = 0; i < n; i++) {
      sumX += x[i];
      sumY += y[i];
      sumXY += x[i] * y[i];
      sumXSquare += x[i] * x[i];
      sumYSquare += y[i] * y[i];
    }
    double correlation = (n * sumXY - sumX * sumY) /
      Math.Sqrt((n * sumXSquare - sumX * sumX) * (n * sumYSquare - sumY * sumY));
    return correlation;
  }

 }
}
```