МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко
підпис
«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Програмне забезпечення прогнозування цін на основі інструментарію штучного інтелекту

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 408.2011002

Здобувач _____Я. С. Антонов
підпис
«__» _____ 2024 р.

Керівник PhD, ст. викладач _____ І. О. Кандиба
підпис
«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
підпис
«__» _____ 2024 р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри Інженерії Програмного

Забезпечення _____Є. О. Давиденко

«_22__» ___грудня____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 408 факультету комп'ютерних наук

_____Антонов Ярослав Сергійович_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Програмне забезпечення прогнозування цін на основі інструментарію штучного інтелекту_____

Затверджена наказом по ЧНУ від «22» ___грудня___ 2023 р. № ___269_____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є програмне забезпечення для прогнозування цін

_____

_____

4. Перелік питань, що підлягають розробці:

- аналіз існуючих аналогів для прогнозування цін на основі ШІ;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проєктування ПЗ;
- проєктування та моделювання ПЗ;
- розробка ПЗ, що реалізує обрані методи ШІ;
- проведення тестування та виправлення помилок.

5. Перелік графічних матеріалів

__Презентація_____

6. Завдання до спеціальної частини

__Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи ____PhD, ст викладач__Кандиба Ігор Олександрович _____

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____ Антонов Ярослав Сергійович_____

(прізвище, ім’я, по батькові)

_____

(підпис)

Дата видачі завдання «_22_» ____грудня_____ 2023р.

КАЛЕНДАРНИЙ ПЛАН

виконання бакалаврської кваліфікаційної роботи

Тема: «Програмне забезпечення прогнозування цін на основі інструментарію штучного інтелекту»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	22.12.23	22.12.23	Виконано
2	Огляд літератури за темою роботи	04.01.24	04.01.24	Виконано
3	Складання календарного плану КРБ	15.01.24	15.01.24	Виконано
4	Аналіз предметної області	18.01.24	18.01.24	Виконано
5	Проєктування ПЗ	22.01.24	30.01.24	Виконано
6	Моделювання та конструювання	22.01.24	30.01.24	Виконано
7	Кодування ПЗ	22.01.24	30.01.24	Виконано
8	Розробка керівництва користувача	20.03.24	20.03.24	Виконано
9	Розробка частини з охорони праці	20.03.24	20.03.24	Виконано
10	Оформлення КРБ та презентації	20.03.24	20.03.24	Виконано
11	Відгук керівника КРБ	29.03.24	29.03.24	Виконано
12	Попередній захист	5.06.24	5.06.24	Виконано
13	Рецензування	15.06.24	18.06.24	Виконано
14	Захист кваліфікаційної роботи			

Розробив студент _____Антонов Ярослав Сергійович_____ _____

(прізвище, ім’я, по батькові) (підпис)

«15» __січня__ 2024 р.

Керівник роботи _PhD, ст викладач___Кандиба Ігор Олександрович _____

(посада, прізвище, ім’я, по батькові) (підпис)

«15» _січня__ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Програмне забезпечення прогнозування цін на основі інструментарію штучного інтелекту»

Студент 408 гр,: Антонов Ярослав Сергійович

Керівник: PhD, ст. викладач Кандиба І. О.

Кваліфікаційна робота присвячена розробці та впровадженню програмного забезпечення, спрямованого на надання функціоналу прогнозування цін на різноманітні товари.

Об'єктом кваліфікаційної роботи є процес створення програмного забезпечення для прогнозування цін на основі ШІ.

Предметом кваліфікаційної роботи є інструментарій та методи ШІ, що використовують для прогнозування цін.

Метою кваліфікаційної роботи є розробка ПЗ для прогнозування цін на основі ШІ, що спростить процес керування портфелем акцій.

В першому розділі проаналізовано предметну область та проведено порівняння аналогічних рішень з виділенням переваг та недоліків. Сформовано основні функції та виявлено конкретні специфікації вимог.

В другому розділі вирішено задачі пов’язані з моделюванням та проєктуванням, розроблено низку проєктних рішень, а саме сценарії використання, діаграми послідовності та діаграми діяльності

В третьому розділі обрано архітектуру та сформовано моделі основних класів системи. Досліджено технічні рішення, обрано технологічний стек. Також побудовано макети користувацького інтерфейсу.

В четвертому розділі реалізовано усі необхідні функціональні модулі та представлення застосунку.

КРБ викладена на 59 сторінок, вона містить 4 розділи, 33 ілюстрацій, 7 таблиць, 21 джерел в переліку посилань

Ключові слова: програмне забезпечення, штучний інтелект, прогнозування цін, акції, модель навчання.

ABSTRACT

for the bachelor's qualification work

"Price forecast software based on artificial intelligence tools"

Student of group 408: Antonov Yaroslav Serhiyovich

Supervisor: PhD, senior lecturer Kandyba I. O.

This work is dedicated to the development and implementation of software aimed at providing functionality for forecasting the prices of various goods.

The object of the qualification work is the process of creating software for predicting prices based on AI.

The subject of the qualification work is the tools and methods of AI used for price prediction.

The aim of the qualification work is to develop AI-based software for price prediction that will be accurate, reliable, and easy to use.

In the first section, the subject area was analyzed, and a comparison of similar solutions was carried out based on the observed advantages and shortcomings. The main functions have been formulated and requirement specifications have been formulated.

In second section, tasks related to modeling and design have been solved, developed use-case scenarios, sequence diagrams and activity diagrams.

In the third section, the architecture is selected and a model of the main classes of the system is formed. Technical solutions have been researched and a technology stack has been developed. The mockup of the core interface was also created.

In the fourth section, all the necessary functional modules and views are implemented.

The qualification work consists of 59 pages, it contains 4 chapters, 33 illustrations, 7 tables, and 21 sources in the reference list.

Keywords: software, artificial intelligence, price forecast, stocks, learning model.

ЗМІСТ

ВСТУП 3
1 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН 5
1.1 Визначення цільової платформи 5
1.2 Аналіз реалізацій аналогічних систем 6
1.3 Специфікація вимог до програмного забезпечення 11
Висновки до розділу 1 16
2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН 17
2.1 Моделювання сценаріїв використання системи 17
2.2 Моделювання поведінки системи 22
2.3 Діаграма діяльності 25
Висновки до розділу 2 28
3.1 Вибір архітектури 29
3.2 Діаграма класів та компонентів 32
3.3 Розробка макетів інтерфейсу 34
3.4 Вибір технологій та рішень для реалізації ПЗ 39
Висновки до розділу 3 43
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ПРОГНОЗУВАННЯ ЦІН 44
4.1 Налаштування базового оточення розробки 44
4.2 Реалізація основних класів застосунку 46
4.3 Реалізація представлення застосунку 50
Висновки до розділу 4 57
ВИСНОВКИ 58
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 59

ВСТУП

Прогнозування цін – важлива задача в багатьох галузях, таких як торгівля, фінанси, виробництво та логістика. Навіть звичайні інвестори та аналітики використовують інструменти прогнозування. У сучасному світі обсяги інформації та доступність потужних обчислювальних ресурсів роблять штучний інтелект досить перспективним та актуальним інструментом для вирішення цієї задачі.

Практичне значення полягає в тому, що розробка ПЗ для прогнозування цін на основі ШІ мати значний вплив на різні сфери. Це може допомогти:

– зменшити ризики інвестицій та торгівлі: прогнозування цін може допомогти інвесторам та трейдерам приймати більш обґрунтовані рішення, тим самим зменшуючи ризики втрат;

– оптимізувати портфелі, що дозволить інвесторам максимізувати прибуток і мінімізувати ризики свого інвестиційного портфелю;

– розробити нові фінансові продукти та послуги: за допомогою такого застосунку фінансові установи зможуть створювати різноманітні фінансові продукти наприклад деривативи та опціони простіше;

– оптимізувати ланцюги постачання та виробничі процеси, що дозволить компаніям зменшити витрати та покращити ефективність;

– прогнозувати попит на свою продукцію компаніям, що допоможе оптимізувати виробництво;

– регулювати ринок уряду, тим самим захищаючи споживачів та забезпечуючи справедливу конкуренцію.

Об'єктом кваліфікаційної роботи є процес створення програмного забезпечення для прогнозування цін на основі ШІ.

Предметом кваліфікаційної роботи є інструментарій та методи ШІ, що використовують для прогнозування цін.

Метою кваліфікаційної роботи є розробка ПЗ для прогнозування цін на основі ШІ, що спростить процес керування портфелем акцій.

Для досягнення цієї мети визначено наступні завдання:

1. Аналіз існуючих аналогів для прогнозування цін на основі ШІ.
2. Формування специфікації вимог до програмного забезпечення.
3. Визначення архітектури для проєктування ПЗ.
4. Проєктування та моделювання ПЗ.
5. Розробка ПЗ, що реалізує обрані методи ШІ.
6. Проведення тестування та виправлення помилок.

Для реалізації поставлених задач використано мову програмування Python та різноманітні бібліотеки для роботи з даними та статистикою(pandas, matplotlib) а також фреймворк представлення streamlit.

1 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН

1.1 Визначення цільової платформи

Вибір цільової платформи для ПЗ прогнозування цін залежить від багатьох факторів. Спершу було виявлено цільову аудиторію та проведено аналіз їх потреб.

– інвестори та трейдери: Ця група користувачів потребує зручного інтерфейсу, який дозволяє їм швидко та легко отримувати доступ до прогнозів цін, досліджень та інших даних. Мобільний застосунок або вебсайт можуть бути найкращими варіантами для цієї групи;

– фінансові установи: Цим користувачам може знадобитися більш просунута функціональність, така як можливість інтегрувати ПЗ з їхніми власними системами. Програма для персонального комп`ютера або онлайн платформа можуть бути кращим вибором для цієї групи;

– прості користувачі: Ця група може просто хотіти отримувати базові прогнози цін або інформацію про те, як інвестувати. Вебсайт або мобільний застосунок з простим інтерфейсом може бути найкращим вибором для цієї групи.

Враховуючи аналіз цільової аудиторії вебсайт може бути чудовою цільовою платформою для ПЗ прогнозування цін, завдяки своїм численним перевагам [3]:

– доступність: Вебсайти доступні через будь-який браузер на будь-якому пристрої, що робить їх доступними для широкої аудиторії;

– простота використання: Вебсайти можуть бути розроблені з простим та інтуїтивно зрозумілим інтерфейсом, що робить їх зручними для користувачів;

– масштабованість: Вебсайти можуть легко масштабуватися, щоб обслуговувати велику кількість користувачів;

– низькі витрати: Розробка та обслуговування вебсайту, як правило, дешевша, ніж розробка та обслуговування мобільних додатків або програм для комп`ютерів;

– гнучкість: Вебсайти можна легко оновлювати та змінювати, що робить їх ідеальними для динамічних продуктів, таких як ПЗ прогнозування цін.

Отож використання вебсайту як основної цільової платформи є доцільним рішенням.

1.2 Аналіз реалізацій аналогічних систем

Аналіз аналогів є важливою частиною процесу розробки програмного забезпечення. Це допоможе нам:

– зрозуміти ринок: проаналізувавши існуючі рішення, можна краще зрозуміти потреби користувачів та те, як інші компанії вирішують ці потреби;

– визначити можливості: Можливо знайти нові ідеї та можливості для свого продукту, проаналізувавши те, що вже роблять інші;

– уникнути помилок: Навчання на помилках інших компаній та уникати їх повторення у своєму власному продукті;

– створити кращий продукт: Завдяки аналізу аналогів можна створити продукт, який кращий за те, що вже є на ринку.

TradingView

TradingView - це онлайн-платформа для аналізу ринків та соціальної торгівлі, яка використовується трейдерами та інвесторами для дослідження, відстеження та торгівлі фінансовими інструментами. Вона пропонує широкий спектр функцій, включаючи:

– інтерактивні графіки: TradingView пропонує широкий спектр інтерактивних графіків, які можна використовувати для аналізу ринкових

даних. Користувачі можуть налаштувати графіки, додаючи індикатори, лінії трендів, фігури та інші технічні інструменти;

– технічні індикатори: TradingView пропонує понад 100 вбудованих технічних індикаторів, які можна використовувати для виявлення трендів, моделей та можливостей торгівлі;

– сценарії: Користувачі можуть створювати та ділитися власними сценаріями, які автоматично застосовують технічні індикатори та інші правила до графіків;

– соціальні функції: TradingView має активну спільноту трейдерів, які можуть ділитися своїми ідеями, аналізом та прогнозами;

– торгівля: TradingView пропонує інтеграцію з деякими брокерами, що дозволяє користувачам торгувати безпосередньо з платформи;

– прогнозування: TradingView пропонує кілька функцій прогнозування, які допомагають трейдерам та інвесторам виявляти потенційні можливості на ринку. Ці функції ґрунтуються на технічному аналізі, машинному навчанні та інших методах.

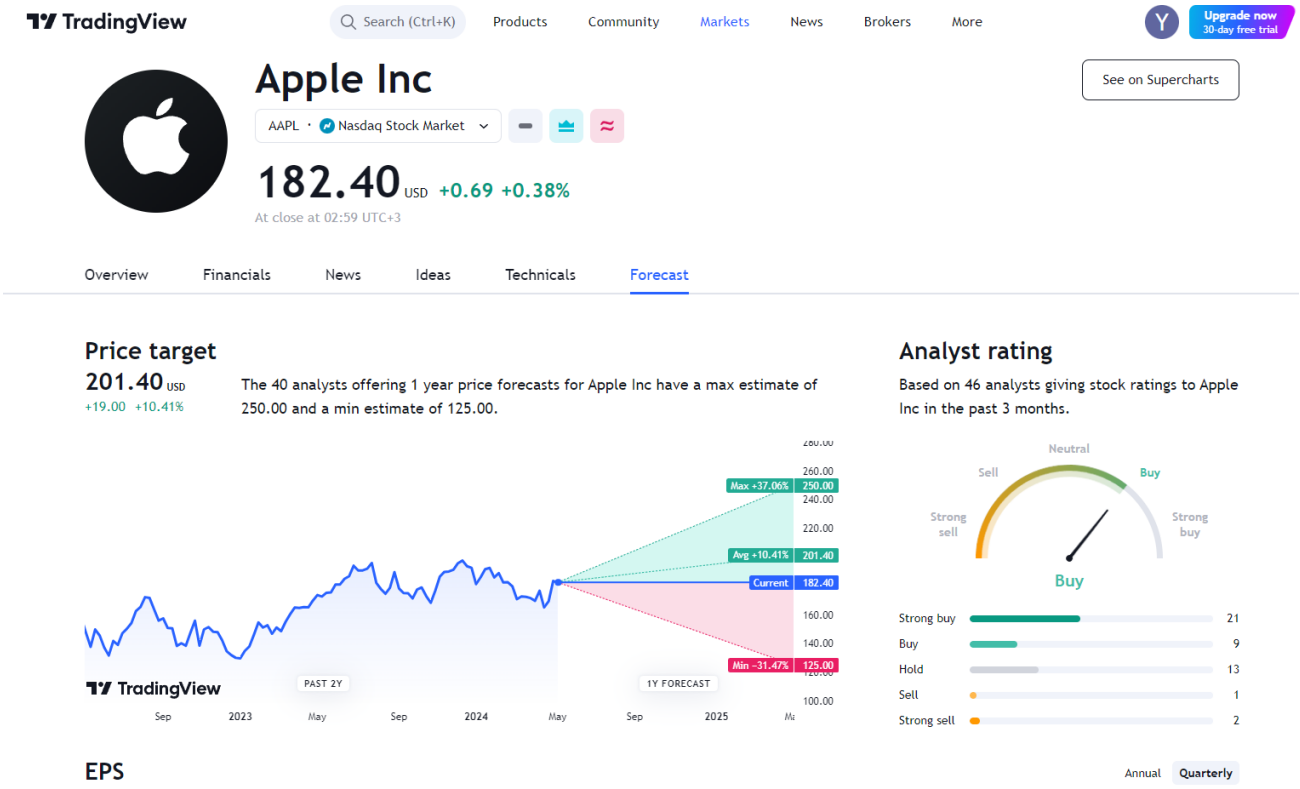

Рисунок 1.1 – Сторінка з прогнозами на ціну акції в Trading View

StockRover

Stock Rover - це платформа для дослідження та аналізу інвестицій, яка використовується інвесторами для виявлення потенційно вигідних акцій. Вона пропонує широкий спектр функцій, наприклад:

– скринінг акцій: Stock Rover дозволяє користувачам фільтрувати тисячі акцій за різними критеріями, такими як фінансові показники, оцінка та динаміка цін;

– аналіз портфеля: Stock Rover дозволяє користувачам відстежувати та аналізувати свої інвестиційні портфелі.

– порівняння акцій: Stock Rover дозволяє користувачам порівнювати різні акції одна з одною;

– дослідження: Stock Rover пропонує доступ до фінансових звітів, новин та інших дослідницьких матеріалів;

– візуалізація: Stock Rover дозволяє користувачам візуалізувати дані про акції за допомогою графіків та діаграм.

Рисунок 1.2 – Приклад дослідження ціни акції за допомогою StockRover

Intrinio

Intrinio відрізняється від TradingView та Stock Rover тим, що воно не є платформою для безпосереднього аналізу ринку або прогнозування цін. Натомість, Intrinio фокусується на тому, щоб бути постачальником фінансових даних для розробників та фінансових установ. Вони пропонують широкий спектр високоякісних фінансових даних через API та набори даних, які можна інтегрувати у власні програми або використовувати для розробки інструментів прогнозування цін.

Ось що пропонує Intrinio:

– фінансові дані США: Intrinio пропонує широкий спектр фінансових даних США, включаючи ціни на акції, фінансові показники, новини, та дані альтернативних джерел.

– глобальні фінансові дані: Intrinio також пропонує обмежений набір глобальних фінансових даних, що постійно розширюється;

– історичні та потокові дані: Intrinio пропонує як історичні, так і потокові дані, що дозволяє розробникам створювати програми, які реагують на ринкові зміни в режимі реального часу;

– API та набори даних: Intrinio надає свої дані через зручні API та набори даних, які можна легко інтегрувати у власні програми.

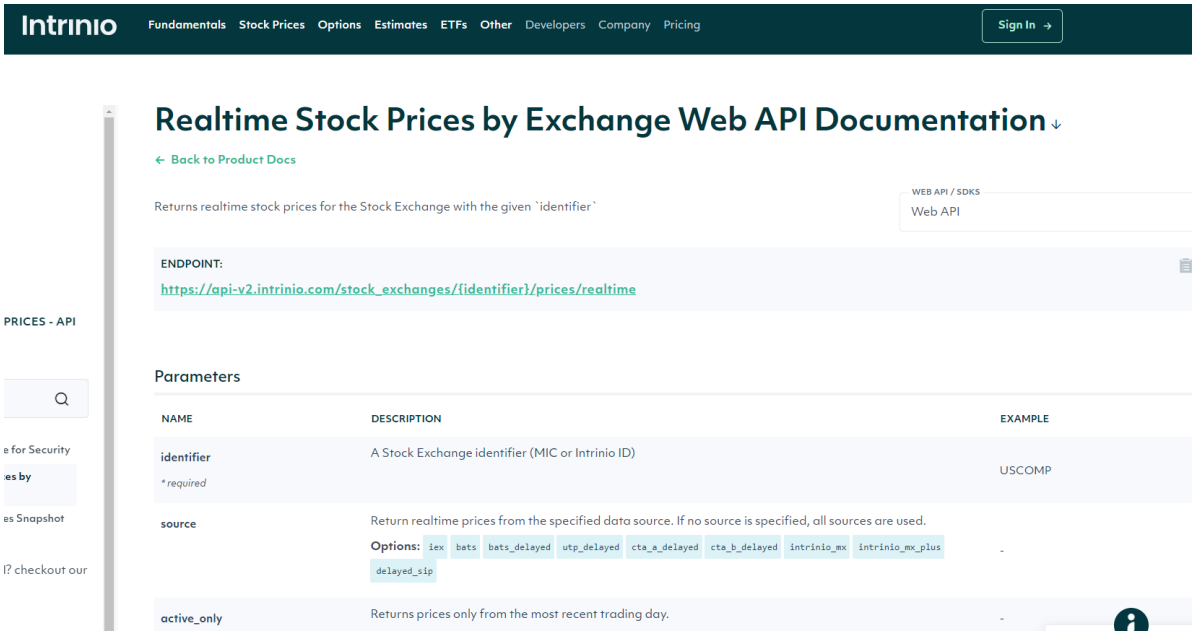

Рисунок 1.3 – Приклад документації Intrinio

Загалом, Intrinio є цінним ресурсом для розробників, які хочуть створити власні інструменти прогнозування цін. Вони пропонують високоякісні фінансові дані та гнучкість, необхідну для розробки складних моделей. Однак Intrinio не призначений для індивідуальних інвесторів, які шукають готового рішення для прогнозування цін.

Отож далі було виділено основні переваги та недоліки наведених аналогічних систем.

Таблиця 1.1 – Переваги та недоліки аналогічних систем

Назва системи	**Переваги**	**Недоліки**
TradingView	1) Широкий спектр інструментів для аналізу та прогнозування. 2) Простий та зручний інтерфейс. 3) Велика спільнота трейдерів. 4) Доступність безкоштовної версії	1) Платна підписка для доступу до деяких функцій. 2) Відсутність деяких інструментів для фундаментального аналізу.
Stock Rover	1) Потужні інструменти для фундаментального аналізу. 2) Простий та зручний інтерфейс. 3) Доступні ціни на підписку. 4) Дослідницькі матеріали	1) Відсутність деяких інструментів для технічного аналізу. 2) Немає можливості прогнозувати ціни. 3) Вартість
Intrinio	1) Широкий спектр даних про активи. 2) Потужні інструменти для алгоритмічної торгівлі. 3) Доступ до API. 4) Гнучкість	1) Складний інтерфейс. 2) Висока ціна підписки. 3) Не призначений для індивідуальних інвесторів 4) Немає можливості прогнозувати ціни

1.3 Специфікація вимог до програмного забезпечення

Після проведення аналізу аналогів та виділення їх основного функціоналу було вирішено розробити специфікацію вимог до ПЗ. Специфікація вимог до ПЗ - це документ, який описує функціональні та нефункціональні вимоги до програмного забезпечення[4]. Вона

використовується для чіткого визначення того, що має робити ПЗ, як воно має працювати та які його характеристики.

Розробка вимог є важливою частиною процесу розробки ПЗ, тому що вона[6]:

– забезпечує чітке розуміння того, що має робити ПЗ;

– допомагає уникнути непорозумінь між розробниками, користувачами та іншими зацікавленими сторонами;

– слугує основою для тестування та прийняття ПЗ;

– допомагає керувати процесом розробки ПЗ.

Призначення системи (застосунку), для якої розробляється програмне забезпечення:

– інвестиції: допомагати індивідуальним інвесторам приймати обґрунтовані рішення про купівлю та продаж акцій, валют, товарів та інших активів;

– торгівля: допомагати трейдерам визначити потенційно вигідні торгові можливості;

– управління ризиками: допомагати компаніям та інвесторам управляти ризиками, пов'язаними з коливаннями цін;

– дослідження: допомагати дослідникам вивчати поведінку ринків та розробляти нові теорії прогнозування цін;

– фінансове планування: допомагати людям планувати своє фінансове майбутнє, наприклад, при плануванні пенсії або купівлі будинку чи будь яких інших активів.

Характеристики користувачів

Користувач (User) – роль з правами доступу до основних функцій системи. Також функціонал користувача розширюється при автентифікації до системи.

Загальна структура і склад системи

Система повинна представляти з себе веб-застосунок, що складається з основного багатосторінкового сайту до якого отримують доступ усі користувачі.

Загальні обмеження

– непередбачувані події - на ринках акцій можуть траплятися непередбачувані події, такі як стихійні лиха, терористичні атаки або політичні потрясіння. Ці події можуть суттєво вплинути на ціни акцій, і їх неможливо передбачити за допомогою жодної моделі прогнозування;

– маніпуляції - ринок акцій може піддаватися маніпуляціям з боку великих інвесторів або інсайдерів, що може призвести до короткострокових коливань цін, які не відображають реальну вартість компанії;

– емоційний фактор - на поведінку інвесторів часто впливають емоції, такі як страх та жадібність. Ці емоції можуть призвести до ірраціональної поведінки, яка може призвести до короткострокових коливань цін, які не відображають реальну вартість компанії.

Функції системи:

– система повинна збирати дані з перевірених джерел (біржі, фінансові вебсайти, API даних, новинні статті);

– система повинна обробляти дані за допомогою методів машинного навчання та ШІ;

– система повинна генерувати прогнози цін на акції для різних часових горизонтів (короткострокові, середньострокові, довгострокові);

– система повинна надавати користувачу графіки, діаграми, таблиці та інші візуальні елементи;

– система повинна дозволяти користувачам налаштовувати елементи візуалізації;

– система повинна надавати зареєстрованому користувачу можливість додавати до свого портфелю акції та оцінювати їх прибутковість використовуючи інструменти прогнозування цін;

– система повинна надавати користувачу вибірку з найпопулярніших акцій для прогнозування цін.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Система розробляється для функціонування на пристроях з можливістю отримати доступ до інтернету за допомогою браузеру та з графічним дисплеєм.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Клієнт-серверна архітектура з трьома основними компонентами[5]:

– сервер - надає прогнози цін, дані та інші послуги клієнтським програмам. Використовує ШІ та машинне навчання для обробки даних та генерування прогнозів. Повинен бути масштабованим та надійним, щоб обробляти велику кількість запитів;

– клієнти - веб-інтерфейси, що відображають прогнози цін, дані та інші послуги сервера. Надають користувачу інтерфейс для взаємодії з системою;

– мережа - забезпечує зв'язок між сервером та клієнтами. Повинна мати достатню пропускну здатність для обробки трафіку.

Системне програмне забезпечення

Система має бути розгорнута на обладнанні із встановленою операційною системою сімейства GNU/Linux. Також має бути використаний засіб контейнеризації. (Docker, Kubernetes)

Мережеве програмне забезпечення

Веб-сервер (Apache, Nginx) для обслуговування HTTP-запитів від клієнтів.

Програмне забезпечення ведення інформаційної бази

У якості системи управління базами даних повинна виступати SQLServer версії 2019 і вище.

Мова і технологія розробки ПЗ

Серверна частина застосунку повинна бути реалізована на мові програмування Python, з використанням бібліотек pandas, matplotlib та sklearn.

Клієнтська частина застосунку повинна бути реалізована за допомогою Python бібліотеки streamlit. Та взаємодіяти з серверною частиною за допомогою інтерфейсу Rest API.

Комунікаційний протокол

Система повинна використовувати протокол HTTPS для передачі даних між клієнтом та сервером.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Безпека

Система повинна бути безпечною та захищати дані користувачів від несанкціонованого доступу

Масштабування

Система повинна бути масштабованою та здатною обробляти зростаючий обсяг даних та трафіку.

Продуктивність

Час обробки запиту серверною частиною застосунку має становити не більше 10 секунд при нормальному навантаженні обладнання.

Переносимість

Застосунок повинен бути кроссбраузерним та відображатися однаково у всіх браузерах. Також інтерфейс має бути адаптивним незалежно від пристрою користувача.

Висновки до розділу 1

У першому розділі проаналізовано предметну область та визначено цільову платформу для системи що розробляється. Було прийнято рішення про використання веб-застосунку як основної цільової платформи

Враховуючи проведений аналіз аналогічних систем на цільовій платформі зроблено висновок про наявність недоліків у проаналізованих системах – відсутність функцій прогнозування та платна підписка для доступу для багатьох функцій.

Переглянуті системи в більшості мають досить великий функціонал та вимагають купувати підписку для доступу до функцій більшість з яких може бути не потрібна звичайному індивідуальному інвестору. Цей факт робить доцільним розробку простого у використанні інструменту для прогнозування цін, адже значно збільшує кількість потенційних користувачів, долучаючи навіть аудиторію не знайомою з технічним аналізом та взагалі теорією фінансових ринків.

Із результатів аналізу програмних рішень для прогнозування цін можна сформулювати завдання, що стоїть при розробці цієї системи: потрібно розробити систему прогнозування цін на основі штучного інтелекту що забезпечить реалізацію усіх основних стадій прогнозування цін, від пошуку певної акції до представлення прогнозу на різних часових проміжках у різних варіантах представлення а також надасть користувачу можливість слідкувати за певними акціями.

Результатом проведеної у розділі роботи стало специфікація вимог до розроблюваного програмного забезпечення.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН

2.1 Моделювання сценаріїв використання системи

Наступним після аналізу аналогів та визначення усіх необхідних вимог і специфікацій кроком було визначено моделювання теоретичної основи системи, а саме системне моделювання. Моделювання системи може мати багато переваг для розробки ПЗ, наприклад:

Покращене розуміння - моделювання допомагає візуалізувати та зрозуміти складні системи. Це може допомогти краще зрозуміти, як працює система, як різні компоненти взаємодіють один з одним та як зміни в одному компоненті вплинуть на інші.

Раннє виявлення проблем - моделювання може допомогти виявити потенційні проблеми в системі на ранніх етапах розробки. Це може допомогти уникнути дорогих помилок і переробок пізніше.

Покращена комунікація - моделі можуть бути потужним інструментом спілкування. Можливо використовувати їх, щоб поділитися своїми ідеями з іншими зацікавленими сторонами, такими як клієнти, розробники та керівники.

Збільшення продуктивності - моделювання може допомогти розробити більш ефективні системи. Наприклад використовувати моделювання, щоб оптимізувати продуктивність системи та ідентифікувати вузькі місця.

Наступним кроком було обрано методи системного моделювання. Вирішено що для моделювання функціональної складової системи прогнозування цін на основі ШІ найкраще підійдуть UML діаграми – діаграми створені за принципами мови Unified Modeling Language, яка використовується для опису структури та поведінки програмних систем. Також було вирішено використовувати сценарії використання що допоможуть

описати процес взаємодії користувача із системою та способи якими користувачі можуть використовувати систему для прогнозування цін.

Сценарій використання – це опис того, як користувач, тобто дійова особа, взаємодіє з системою для досягнення певної мети [1]. Іншими словами, він описує послідовність кроків, які користувач виконує, щоб використовувати певну функціональність системи та описує поведінку цієї системи при її відповідях. Також нерідко до сценарію додають альтернативні шляхи описуючи варіації поведінки.

Не існує чіткого стандарту написання сценаріїв, що зазвичай призводить до створення кожною організацією певних власних стандартів та шаблонів документів сценаріїв. Проте зазвичай задля створення сценарію виділяють правило в невеликий обсяг кроків (не більше 10) та зрозумілий формат самого сценарію що включає:

– опис функції: Що користувач намагається зробити? Лаконічне описання сценарію;

– діючі особи: актори, або ж учасники взаємодії сценарію;

– передумови: Які умови повинні бути виконані, перш ніж користувач зможе розпочати сценарій;

– кроки: Які кроки виконує користувач для досягнення мети? Зазвичай не перебільшує одне речення;

– очікуваний результат: Що очікує користувач побачити або отримати після виконання кроків;

– розширення: за наявності альтернативних кроків чи результатів у сценарії їх вказують в окремій секції.

Отож наступним кроком було сформовано сценарії використання системи для прогнозування цін базуючись на специфікаціях вимог сформованих в минулому розділі. Важливо зазначити що найбільш розповсюджені сценарії, такі як реєстрація чи авторизація, не розглядаються.

Таблиця 2.1 – Сценарій перегляду прогнозу на найпопулярніші акції

Діючі особи	Користувач, Система
Мета	Отримати прогноз цін на найпопулярніші акції
Передумова	Користувач знаходиться на головній сторінці застосунку
Успішний сценарій: 1) Користувач натискає кнопку «Отримати популярні прогнози»; 2) Система обробляє запит за допомогою алгоритмів машинного навчання; 3) Система оновлює сторінку з відповідними результатами обробки;	
Результат	Користувач отримує зведений звіт по найпопулярніших прогнозах
Розширення:	
2а	Система видає кешовані данні якщо такий запит нещодавно оброблювався

Таблиця 2.2 – Сценарій прогнозування ціни на певну акцію

Діючі особи	Користувач, Система
Мета	Отримати прогноз ціни на певну акцію
Передумова	Користувач знаходиться на головній сторінці застосунку
Успішний сценарій: 1) Користувач вводить у пошуковий рядок назву акції; 2) Система надає список знайдених за назвою акцій; 3) Користувач обирає потрібну акцію та обирає часовий проміжок прогнозування; 4) Система генерує прогноз ціни на акцію на основі алгоритмів штучного інтелекту; 5) Користувач може переглянути прогноз у вигляді таблиці;	
Результат	Користувач отримує прогноз ціни на певну акцію

Кінець таблиці 2.2

5а	Користувач може переглянути прогноз у вигляді графіку

Таблиця 2.3 – Сценарій прогнозування ціни на певну акцію

Діючі особи	Користувач, Система
Мета	Додати акцію до свого портфелю
Передумова	Користувач автентифікований у системі та знаходиться у меню особистого портфелю
Успішний сценарій: 1) Користувач вводить у пошуковий рядок назву акції; 2) Система надає список знайдених за назвою акцій; 3) Користувач обирає потрібну акцію та натискає кнопку додавання до портфелю; 4) Система додає до БД інформацію про зміни у портфелі користувача; 5) Користувач може побачити акцію в свому портфелі;	
Результат	Користувач додав акцію до свого порфтелю

Таблиця 2.4 – Сценарій прогнозування ціни на певну акцію

Діючі особи	Користувач, Система
Мета	Отримати прогноз ціни на додану до портфелю акцію
Передумова	Користувач автентифікований у системі та знаходиться у меню особистого портфелю
Успішний сценарій: 1) Користувач обирає з свого портфелю акцію та натискає кнопку «отримати прогноз ціни»; 2) Система генерує прогноз ціни на акцію на основі алгоритмів штучного інтелекту; 3) Користувач може переглянути прогноз у вигляді таблиці;	
Результат	Користувач отримує прогноз ціни на певну акцію зі свого портфелю

Кінець таблиці 2.4

Розширення	
За	Користувач може переглянути прогноз у вигляді графіку

Таблиця 2.5 – Сценарій прогнозування ціни на певну акцію

Діючі особи	Користувач, Система
Мета	Отримати зведену статистику по портфелю користувача
Передумова	Користувач автентифікований у системі та знаходиться у меню особистого портфелю
Успішний сценарій:	1) Користувач обирає опцію «отримати звіт по портфелю»; 2) Система аналізує портфель користувача та формує звіт; 3) Система надає користувачу інформацію про прибуткові та витратні акції його портфелю, їх відношення та кількість;
Результат	Користувач отримав зведену статистику по своєму інвестиційному портфелю

Також гарним доповненням сценаріїв використання можна вважати діаграму прецедентів - інструмент моделювання мовою UML, який використовується для опису взаємодій між користувачами (акторами) та системою. Фактично така діаграма складається з двох частин. Актор – тобто логічно пов`язана множина ролей, що виконується під час взаємодії з прецедентами або сутностями. Та сам прецедент – випадок використання, а точніше опис певної поведінки системи з точки зору користувача. Такий випадок показує саме кінцевий результат якщо порівнювати метод з сценаріями використання.

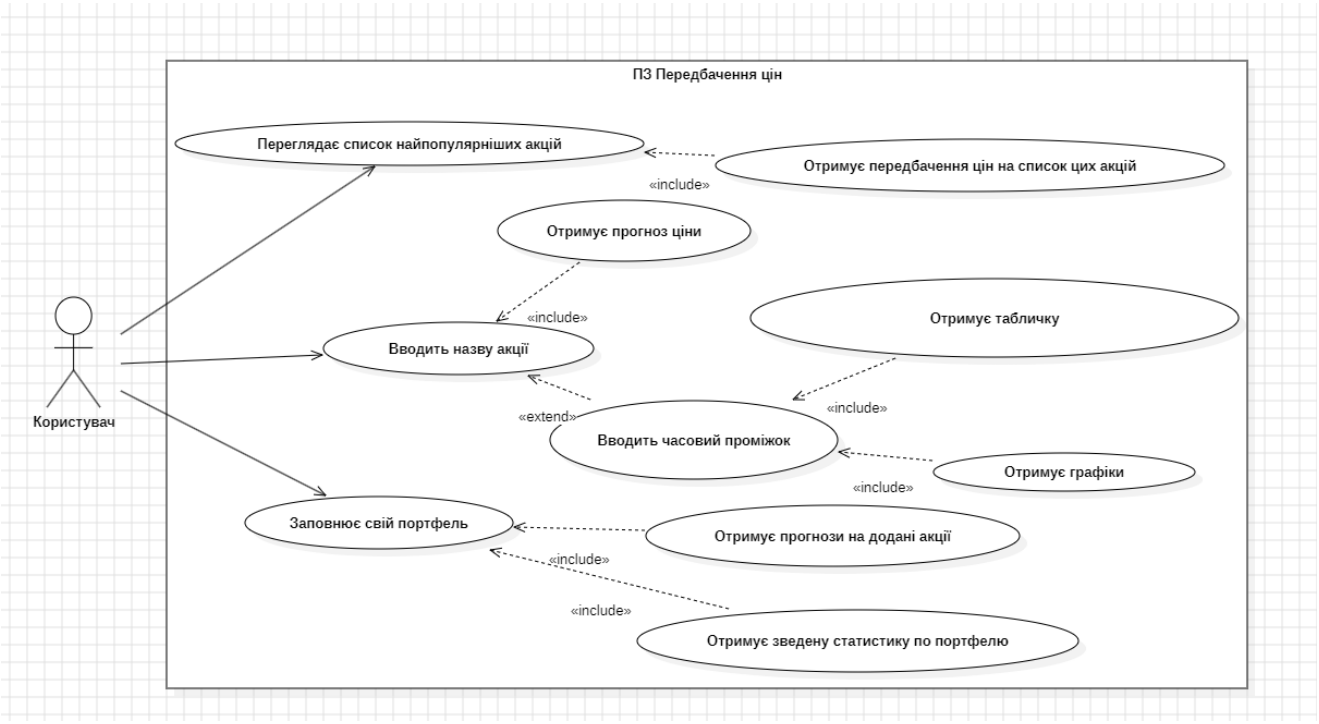

Рисунок 2.1 – Діаграма прецедентів системи

На рис. 2.1 згідно специфікацій вимог до системи представлений основний актор системи а саме користувач та наведені його основні можливості, що збігаються з кінцевими результатами сценаріїв використання системи:

– отримання прогнозу цін на найпопулярніші акції;

– отримання прогнозу цін на певну акцію;

– заповнення інвестиційного портфелю та перегляд прогнозів по ньому;

– отримання звіту по прибутковості чи витратності портфелю на основі передбачень цін.

2.2 Моделювання поведінки системи

Діаграма послідовностей – це тип UML-діаграми, яка використовується для опису взаємодії між об'єктами в системі [2]. Вона візуально показує, як об'єкти надсилають один одному повідомлення протягом певного періоду часу. У контексті розглянутих методів моделювання можна сказати що

діаграма послідовностей більш технічний та детальний сценарій використання що показує послідовність відносин між користувачем а також різними об`єктами системи (зазвичай класами)

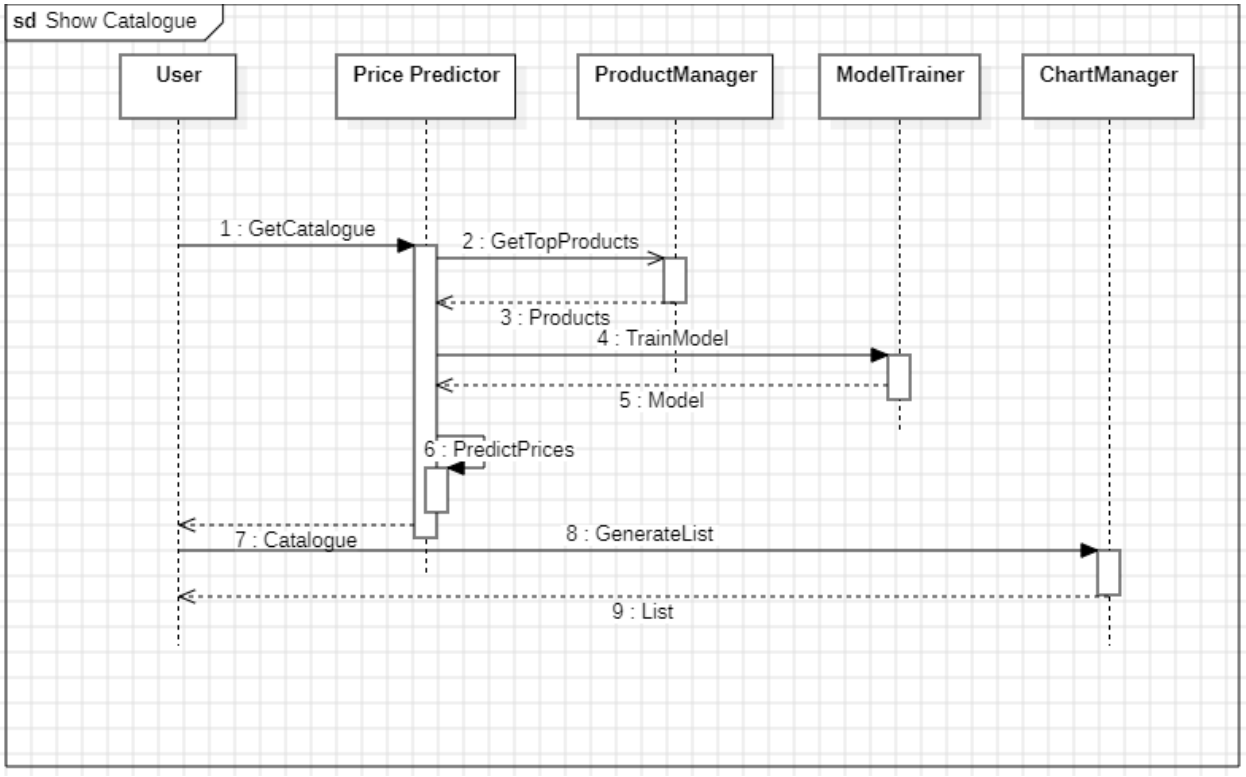

Рисунок 2.2 – Діаграма послідовності «Каталог популярних прогнозів»

Діаграма послідовності «Каталог популярних прогнозів» (рис. 2.2) відображає вже розглянутий в минулих підрозділах сценарій отримання прогнозів на найпопулярніші акції. При виконанні цього варіанта використання користувач надсилає запит на каталог товарів (1) з головної сторінки, після чого головний контролер прогнозування цін надсилає запит (2) контролеру продуктів (акцій в нашому випадку) та після отримання популярних акцій (3) та їх історичних даних надсилає запит контролеру машинного навчання (4). Після успішного тренування моделей машинного навчання вони повертаються до контролеру прогнозування (5) який в свою чергу робить прогнозування цін (6). Врешті користувач отримує каталог

популярних прогнозів, графічні моделі для них створюються (7,8) менеджером графіки.

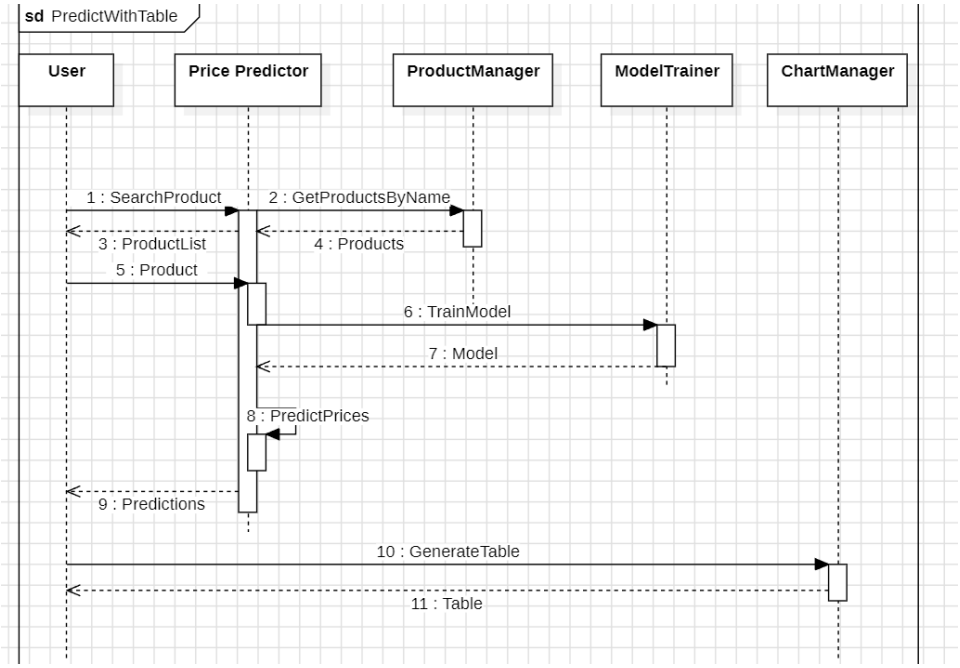

Рисунок 2.3 – Діаграма послідовності «Прогноз на певну акцію у вигляді таблиці»

Отож було розроблено декілька діаграм послідовності для головних сценаріїв використання системи прогнозування цін (рис. 2.3, рис. 2.4)

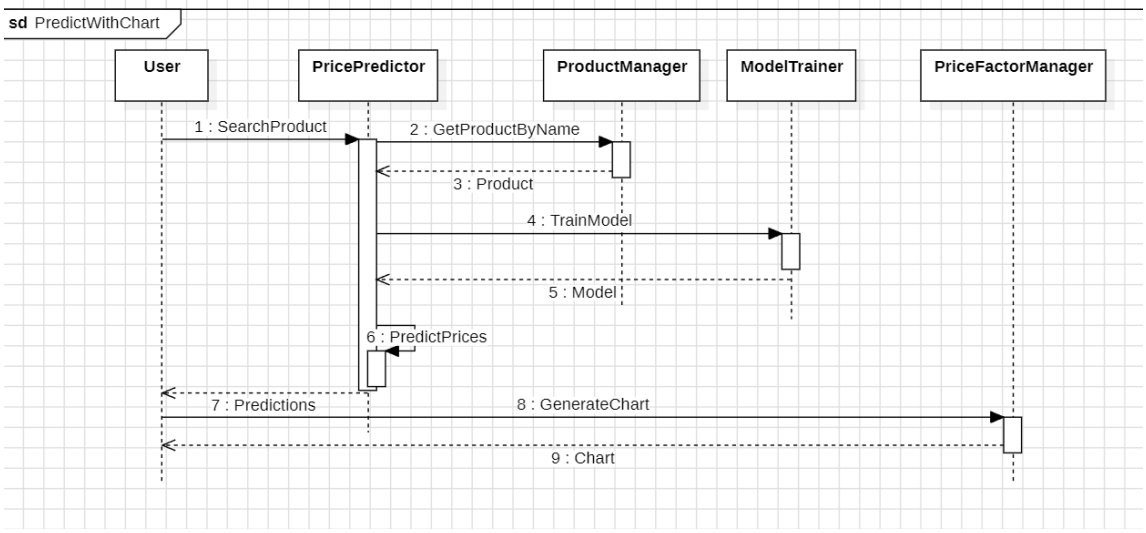

Рисунок 2.4 – Діаграма послідовності «Прогноз на певну акцію у вигляді графіку»

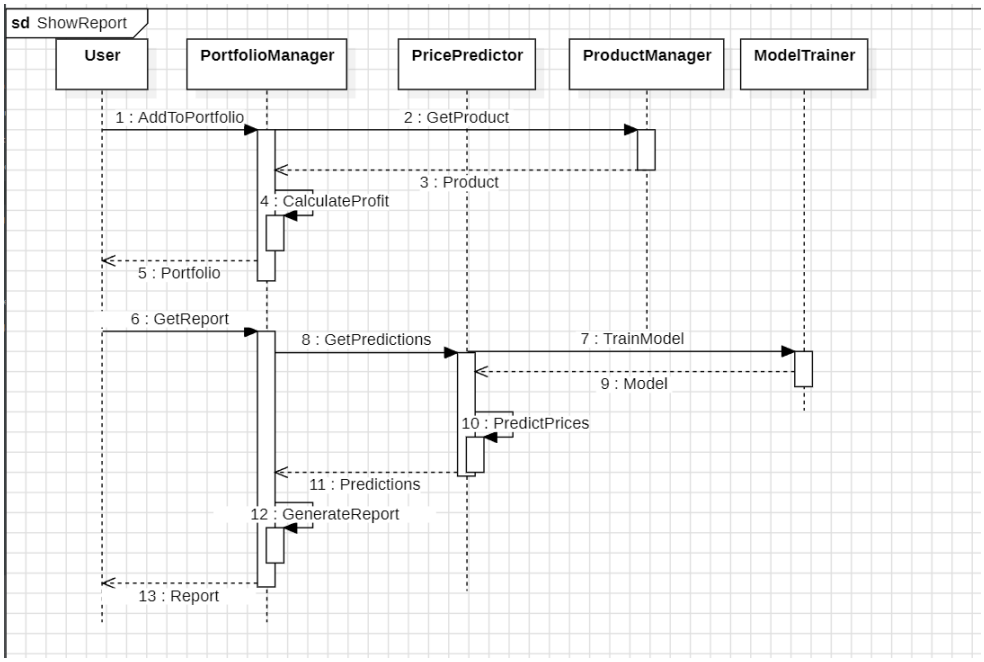

Рисунок 2.5 – Діаграма послідовності «Перегляд звіту по портфелю користувача»

2.3 Діаграма діяльності

Діаграма діяльності – це тип UML-діаграми, яка використовується для опису послідовності дій, необхідних для виконання певного завдання або бізнес-процесу. Вона візуально показує кроки, які потрібно виконати, прийняті рішення, а також задіяні об'єкти та потоки даних.

Діаграми діяльності більше фокусуються на діях та рішеннях, які приймаються користувачем, вони дуже схожі на блок схеми адже показують шлях від початкової до кінцевої точки, враховуючи різні шляхи прийняття рішень що існують під час процесу виконання дії. Отож вони придатні для відображення як послідовних так і одночасних процесів обробки діяльності.

Ось декілька відмінностей з іншими методами моделювання програмного забезпечення:

– сценарії використання зазвичай зосереджуються на тому, як користувач взаємодіє з системою, тоді як діаграми діяльності можуть

описувати будь-який процес, незалежно від того, чи включає він взаємодію з користувачем;

– діаграми послідовностей більше фокусуються на хронологічному порядку повідомлень, що надсилаються та отримуються об'єктами, тоді як діаграми діяльності можуть описувати більш загальні потоки даних та прийняття рішень;

– діаграми потоків даних більше фокусуються на потоках даних, що проходять через систему, тоді як діаграми діяльності більше зосереджуються на діях та рішеннях, які приймаються.

Отож враховуючи універсальність методу для моделювання досить широкого кола процесів та візуальна привабливість, та простота використання, діаграми діяльності можуть бути найкращим вибором для моделювання складного процесу системи з точки зору користувача.

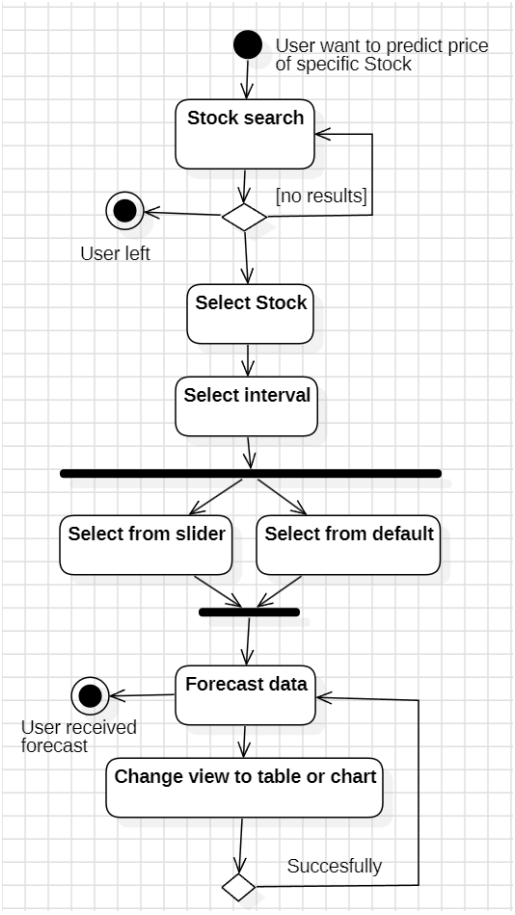

Рисунок 2.6 – Діаграма діяльності «Прогноз на певну акцію»

На рис. 2.6 зображена діаграма діяльності «Прогноз на певну акцію» що є одним з основних сценаріїв використання системи звичайним користувачем. Після того як користувач вводить назву акції що шукає, система приймає рішення чи було знайдено певну акцію за назвою. Цей процес відображається на діаграмі ромбом та повертає користувача до введення назви акції у разі невдачі. Також можна побачити що після визначення акції користувач стикається з розгалуженням а саме вибором інтервалу прогнозування. Після вибору інтервалу одним із способів процес поєднується в один потік. Також в кінці діаграми діяльності після отримання користувачем прогнозу він має можливість обрати інший вигляд цього прогнозу (табличка або графік) що створює певний рекурсивний потік діяльності.

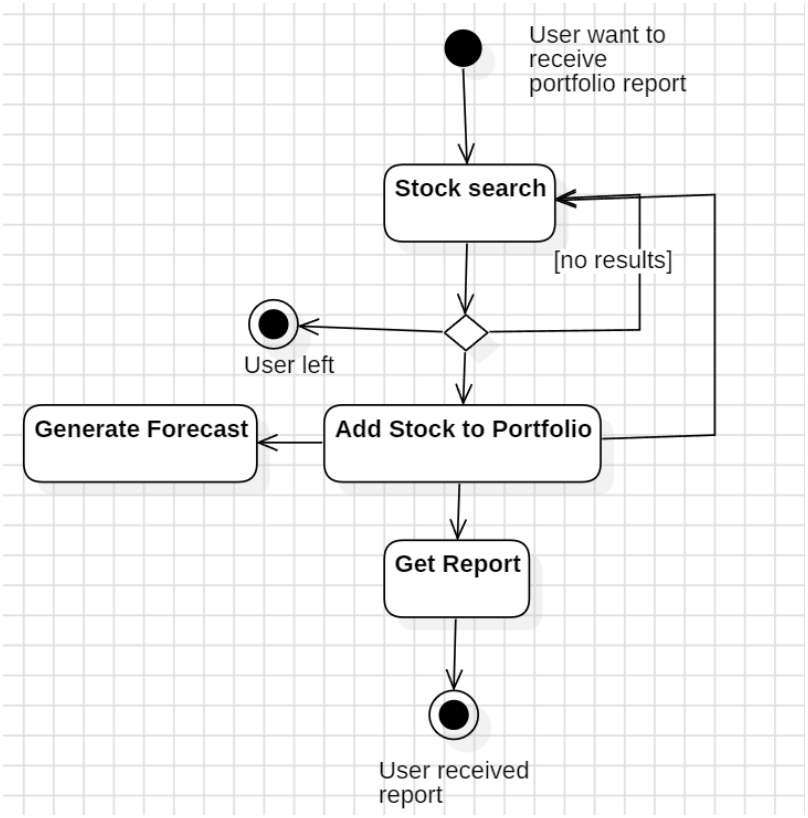

Рисунок 2.7 – Діаграма діяльності «Перегляд звіту по портфелю користувача»

Також було розроблено діаграму діяльності «Перегляд звіту по портфелю користувача» (рис. 2.7). На діаграмі також показується рекурсивна

можливість додавати до портфелю акції перед тим як робити запит на зведений звіт.

Висновки до розділу 2

У другому розділі розглянуто створення різноманітних проєктних рішень для системи що розробляється. Усі рішення були розроблені відповідно до специфікацій вимог до ПЗ що були виявлені у першому розділі. Представлені такі рішення у вигляді сценаріїв використання, тобто вербального опису взаємодії користувача з системою в контексті конкретних її функцій. Також у другій частині розділу були розроблені рішення у вигляді графічних моделей, а саме UML-діаграм – послідовності та діяльності.

Розглянуто основні переваги та мету створення наведелних моделей, та покроково описано їх складові. Насамперед сформовані сценарії використання допомагають сформулювати події та певний алгоритм дій користувача під час використання функій ПЗ, а також надають альтернативні кроки цих дій. Розроблена діаграма прецедентів узагальнює створені сценарії в одну графічну модель що добре репрезентує функціонал системи загалом. Діаграма послідовності в свою чергу моделює той же функціонал з точки зору програми як сукупності класів що послідновно взаємодіють між собою, а діаграма діяльності визначає потік самого алгоритму основним об`єктом моделювання.

Отож створені моделі допоможуть при програмній реалізації застосунку прогнозування цін.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН

3.1 Вибір архітектури

Враховуючи прийняте в першому розділі рішення про вибір вебсайту як основної цільової платформи було розглянуто основні типи відповідних архітектурних рішень.

Монолітна архітектура – це традиційна модель розробки ПЗ, що використовує одну базу коду для виповнення усіх бізнес-завдань. Всі компоненти веб-застосунку зазвичай поєднані в єдиний модуль [7].

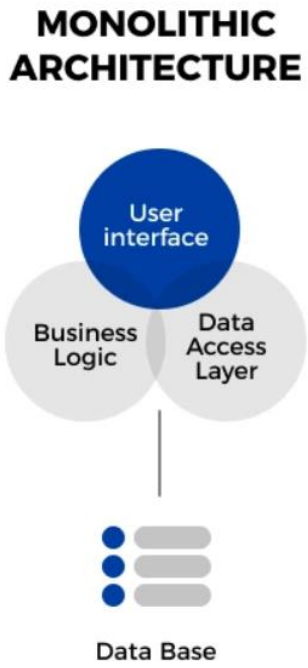

Рисунок 3.1 – Монолітна архітектура

Трирівнева архітектура – архітектурна модель що розділяє веб-застосунок на три рівні [7]:

– рівень представлення: фронтальний інтерфейс, що відображається у браузері користувача;

– рівень логіки: бізнес-логіка програми, обробка та взаємодія з БД;

– рівень даних: БД, де зберігаються дані веб-застосунку.

Рисунок 3.2 – Трирівнева архітектура

Мікросервісна архітектура – розбивка веб-застосунку на маленькі, незалежні та самодостатні сервіси що зазвичай взаємодіють між собою за допомогою API [7].

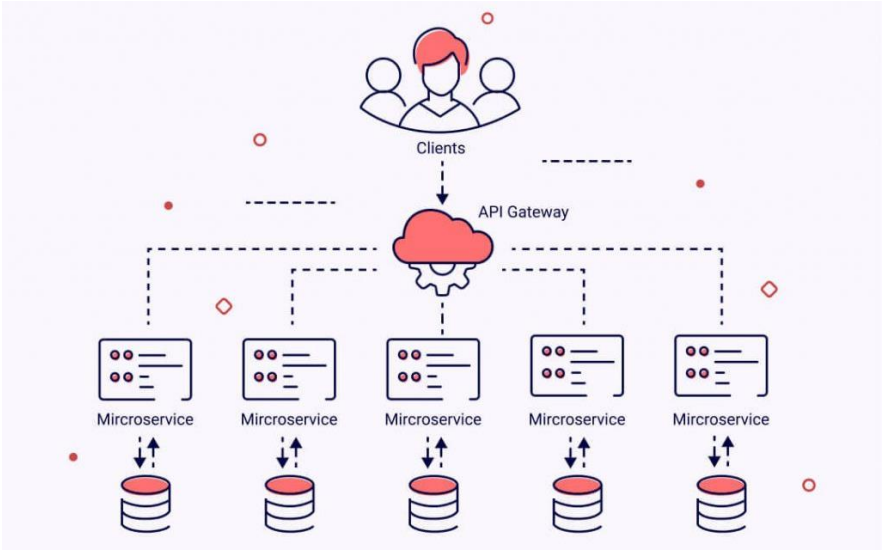

Рисунок 3.3 – Мікросервісна архітектура

Безсерверна архітектура – полягає у використанні хмарних сервісів для роботи веб-застосунку без необхідності керувати інфраструктурою. Застосунок так само буде працювати на сервері але керування цими серверами хмарний провайдер бере повністю на себе. Все більше великих компаній створюють свої хмарні сервіси для безсерверних архітектур але головними гравцями залишаються Amazon Web Services, Microsoft Azure та Google Cloud Platform.

Отож було сформовано порівняльну таблицю переваг та недоліків наведених архітектурних рішень (таб. 3.1).

Таблиця 3.1 – Переваги та недоліки архітектур веб-застосунків

Архітектура	**Переваги**	**Недоліки**
Моноліт	– простота розробки; – легкість розгортання; – чітке розуміння взаємодії компонентів.	– масштабованість; – складність обслуговування та внесення змін; – жорстка залежність між компонентами.
Трирівнева	– модульність; – масштабованість; – гнучкість та простота обслуговування.	– складність розробки; – додаткові витрати на розгортання та підтримку.
Мікросервісна	– висока масштабованість; – гнучкість; – можливість розробки та розгортання незалежно.	– складність розробки, координації та моніторингу; – необхідність ретельного проєктування API.
Безсерверна	– економічність; – висока масштабованість; – автоматичне масштабування; – простота розгортання.	– відсутність контролю над інфраструктурою; – залежність від хмарного провайдера; – складність проєктування для динамічного масштабування.

Враховуючи наведені переваги та недоліки та головну мету роботи, було прийнято рішення про використання монолітної архітектури з деякими елементами трьохрівневої архітектури. Це дозволить створити застосунок що буде відповідати розробленим специфікаціям ПЗ, при цьому не буде перенавантажений складними та непотрібними архітектурними рішеннями. Монолітна архітектура добре підходить для сервісів що надають користувачу рішення для конкретної невеликої задачі, що потребує швидкості і точності, та надає повний контроль над компонентами.

3.2 Діаграма класів та компонентів

Діаграма класів – це візуальний інструмент, що використовують для опису структури об’єктно орієнтованого ПЗ. Така модель чітко ілюструє класи, їхні атрибути, методи та залежності між ними [8]. Головними перевагами використання діаграми класів є:

– ясність та розуміння: Діаграма класів допомагає чітко уявити структуру ПЗ, що робить код більш зрозумілим для розробників та тестувальників;

– раннє виявлення проблем: Діаграма класів допомагає виявити потенційні проблеми дизайну ПЗ на ранніх стадіях розробки, що багатократно економить час та ресурси;

– покращення комунікації: Діаграма класів ефективно описує модель системи що пришвидшує спілкування між розробниками, полегшуючи розуміння та обговорення дизайну ПЗ;

– підтримка та модифікація: Діаграма класів допомагає в підтримці та модифікації ПЗ, роблячи код більш доступним для розуміння та оновлення.

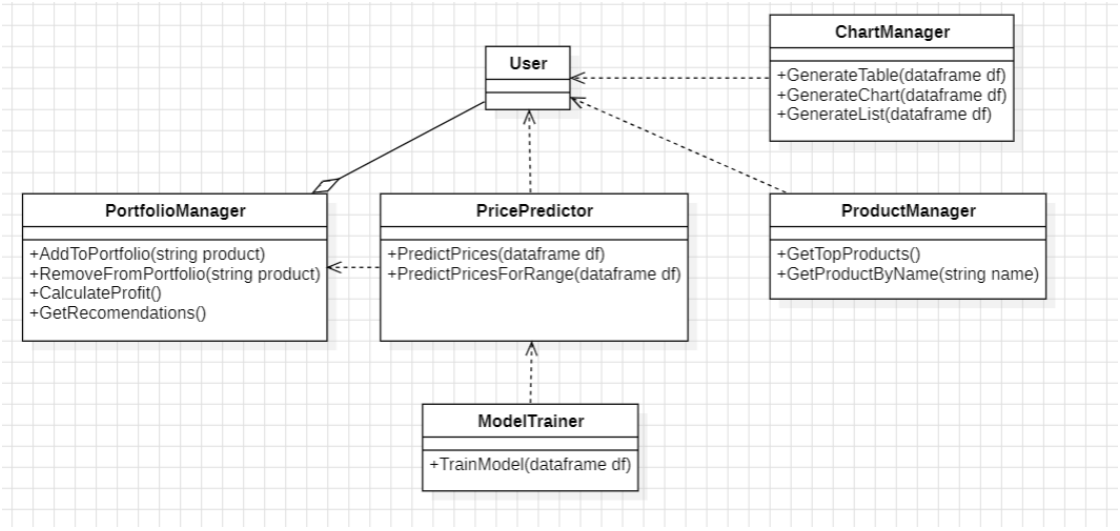

Рисунок 3.4 – Діаграма класів

Отож було розроблено діаграму класів (рис. 3.4), що відповідатиме специфікаціям та усьому необхідному функціоналу системи що

розробляється. Сутність User фактично представляє собою користувацький інтерфейс через який користувач взаємодіє з усіма іншими класами бізнес логіки. Клас ProductManager відповідальний за завантаження історичних даних про акції. Центральний з точки зору основного функціоналу клас PricePredictor надає логіку прогнозування цін на акції використовуючи історичні дані як вхідні параметри та інкапсулює ще один клас необхідний для прогнозування цін – ModelTrainer. Цей клас відповідальний за тренування конкретної моделі машинного навчання, що підходить для прогнозування часових рядів. Також частка бізнес логіки зосереджена в класі PortfolioManager що надає користувачу можливість створити власний портфель акцій, додавати чи видаляти певні тікери зі свого портфелю а також генерувати звіти та одиничні прогнози. Останнім необхідним класом було визначено ChartManager, що інкапсулює логіку генерації представлення даних – створення списків, таблиць та навіть графіків.

Також було вирішено розробити діаграму компонентів (рис. 3.5) – схожий на діаграму класів інструмент що моделює більш абстрактний рівень системи, а саме компоненти [9]. Діаграма чітко показує що класи PricePredictor, ModelTrainer та PortfolioManager інкапсулюють бізнес логіку, та згруповані за цією ознакою.

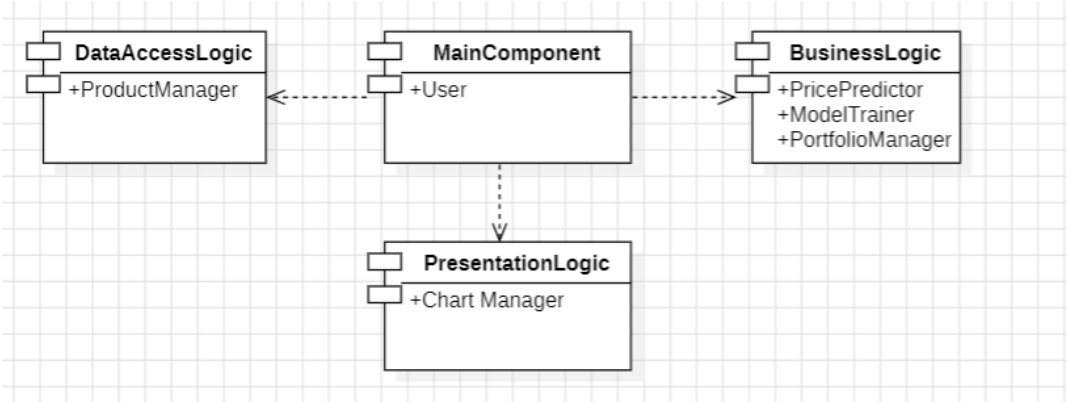

Рисунок 3.5 – Діаграма компонентів

3.3 Розробка макетів інтерфейсу

Наступним кроком було вирішено розробити макети інтерфейсу ПЗ для прогнозування цін. Макети інтерфейсу - це візуальні прототипи інтерфейсу користувача вебсайтів, мобільних застосунків або інших будь-яких цифрових продуктів [3.10]. Розробка таких макетів надає багато переваг, серед яких:

– візуалізація дизайну: Макет дає чітке уявлення про те, як буде виглядати інтерфейс, допомагаючи виявити та виправити проблеми дизайну ПЗ на ранніх стадіях;

– покращення комунікації: Макет полегшує обговорення та узгодження дизайну;

– зменшення витрат: Виявлення та виправлення проблем на етапі макетування економить час і ресурси на етапі розробки;

– підвищення якості продукту: Макетування допомагає створити більш зручний та інтуїтивно зрозумілий інтерфейс для користувачів [3.11].

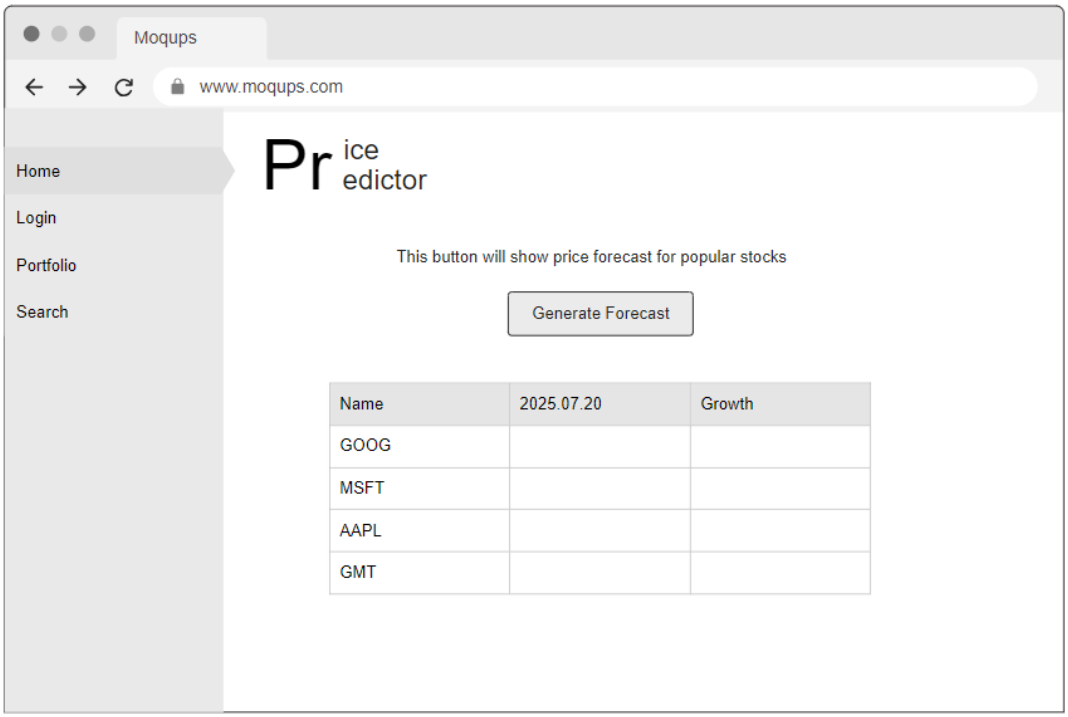

Рисунок 3.6 – Головна сторінка застосунку

Головна сторінка застосунку повинна знайомити користувача з вебсайтом та надавати можливість одразу спробувати основний функціонал ПЗ, а саме отримати прогноз на популярні акції. Отож на макеті (рис. 3.6) представлена головна сторінка, короткий опис функціоналу та кнопка при натисканні на яку користувач отримує зазначений каталог прогнозів. Також на макеті присутнє бокове меню навігації.

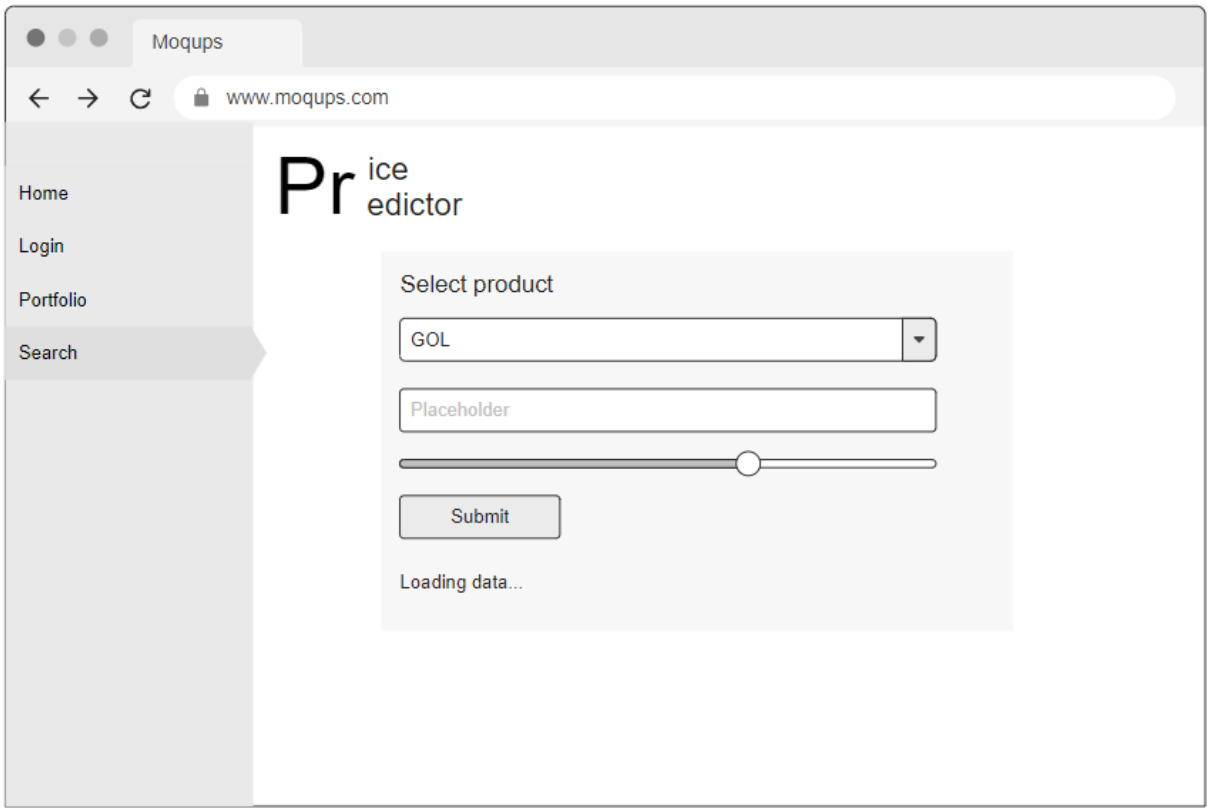

Рисунок 3.7 – Форма на сторінці пошуку акцій для прогнозування

Наступною не менш важливою функцією застосунку є більш розгорнуті прогнози на конкретні акції, цей функціонал було вирішено вивести на окрему сторінку «Search» макет якої представлено на рисунку 3.7. Сторінка містить форму що дозволяє обрати акції зі списку або ж ввести в поле так званий тікер – тобто ідентифікатор акції на біржі. Також на формі присутній слайдер що дозволяє обрати часовий проміжок майбутнього, від 1 до 5 років для прогнозування. В кінці форми розташована кнопка для підтвердження

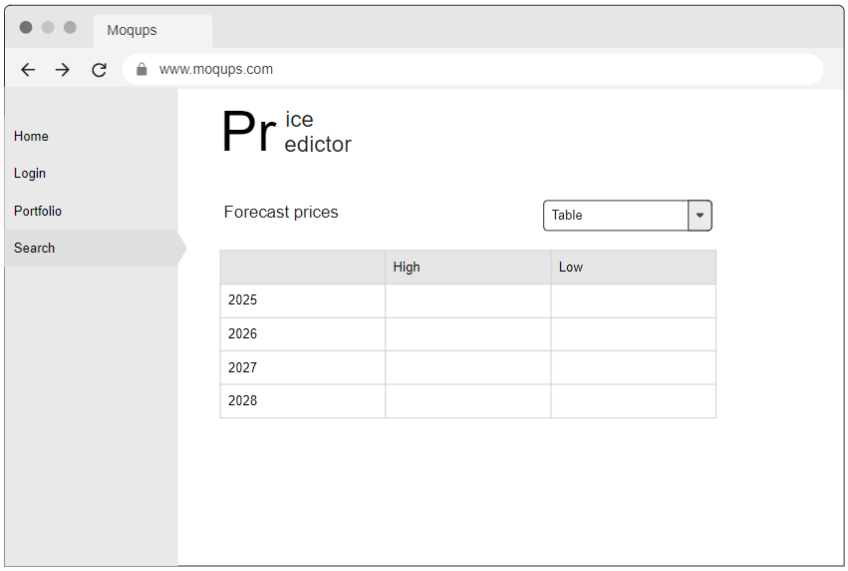

Рисунок 3.8 – Прогноз в представленні таблиці

Після підтвердження зазначеної форми користувач отримує історичні дані про акції та після короткого очікування також отримує саме прогнозування цін у вигляді таблиці або ж графіку. На макетах (рис. 3.8 та рис. 3.9) сконструйовані саме ці представлення.

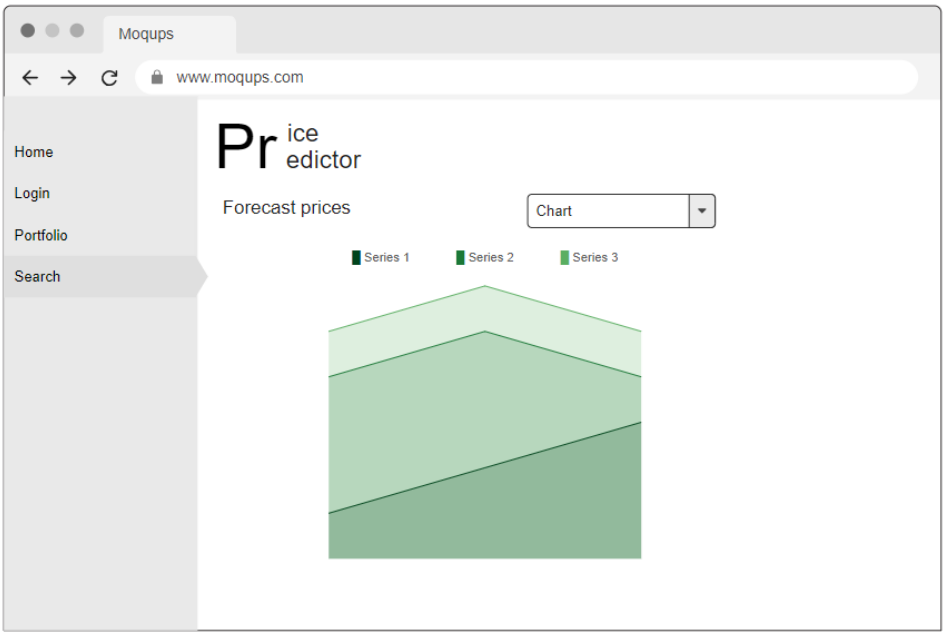

Рисунок 3.9 – Прогноз в представленні графіку

Також слід зазначити що користувач може перемикатися між представленнями за допомогою компоненту «select».

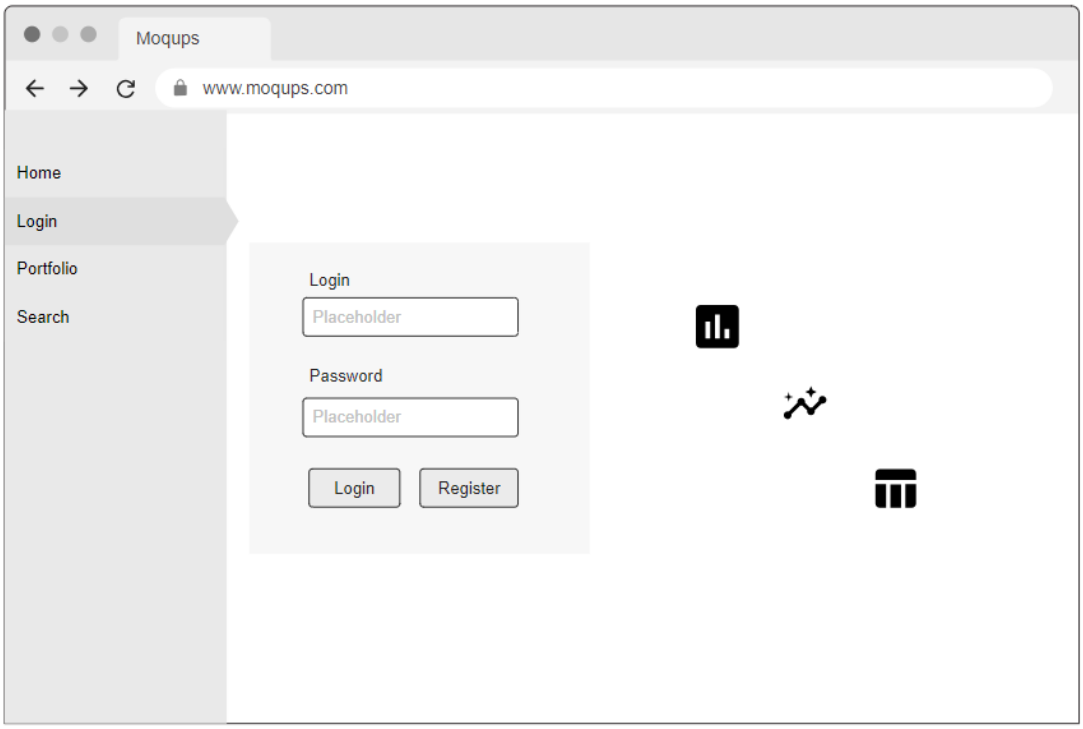

Рисунок 3.10 – Сторінка логіну та регістрації

Наступним кроком було розроблено макет для сторінки логіну або регістрації (рис. 3.10). Сторінка містить лише просту форму з двома полями та кнопками та після проходження автентифікації повинна відображати користувачу його поточні данні та надавати можливість зробити скидання паролю або ж зміну інших користувацьких даних.

Для останньої не менш важливої сторінки – а саме сторінки портфелю користувача також було розроблено макет інтерфейсу (рис. 3.11). На сторінці представлений список акцій що вже додані до портфелю. Кожний елемент списку окрім імені та можливості зробити прогнозування також надає користувачу можливість видалити продукт з портфелю.

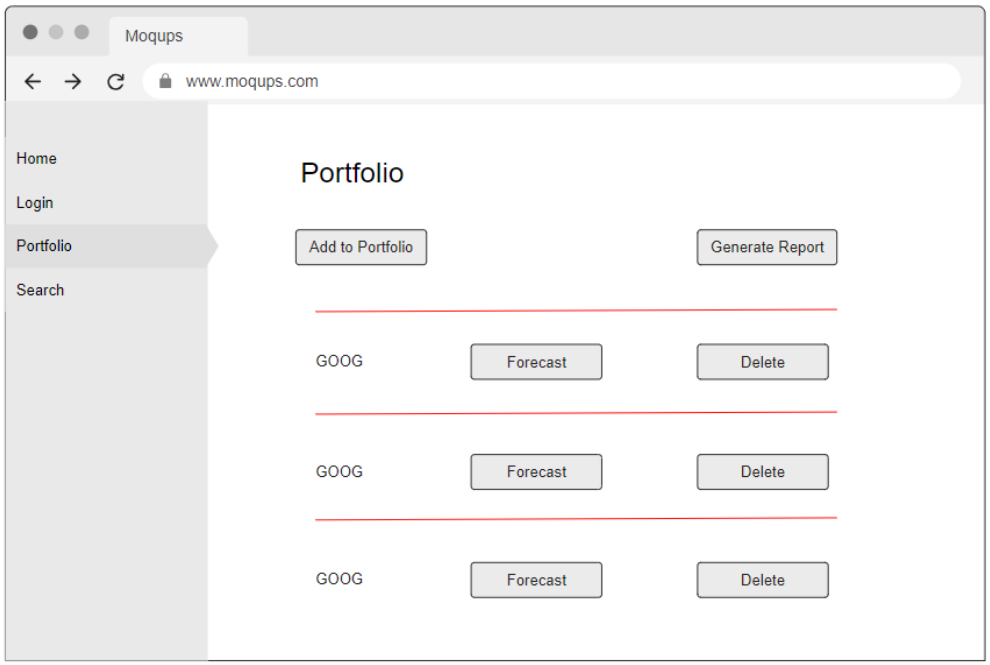

Рисунок 3.11 – Сторінка портфелю користувача

Над списком також присутні кнопки додавання до портфелю нової акції що викликає форму ідентичну до наведеної на сторінці «Search» та кнопку що генерує звіт по портфелю, макет до якого представлений на рисунку 3.12.

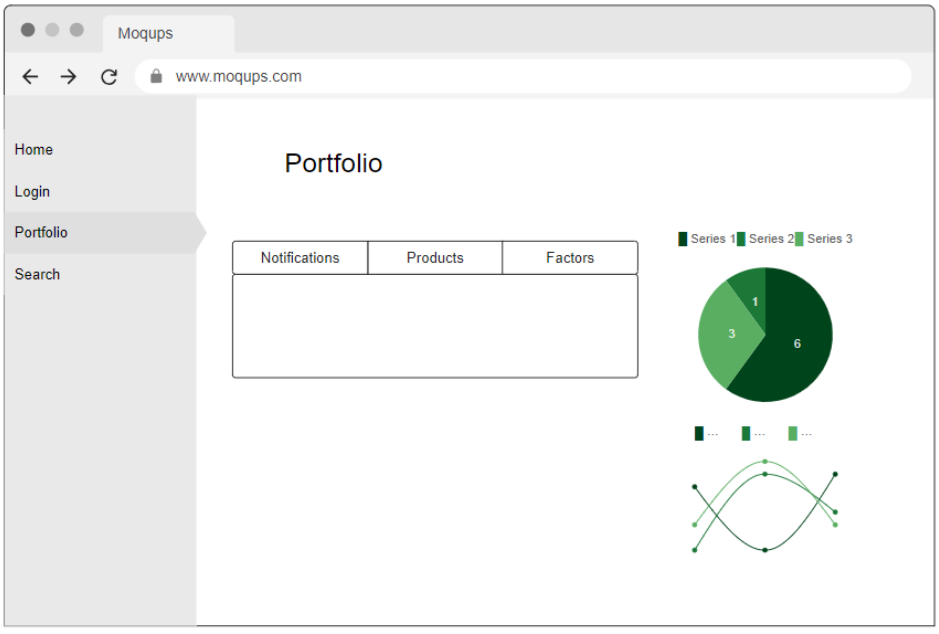

Рисунок 3.12 – Сторінка зі звітом по портфелю користувача

3.4 Вибір технологій та рішень для реалізації ПЗ

Вибір технологій та рішень для реалізації ПЗ прогнозування цін є досить важливим завданням що впливає на весь життєвий цикл проєкту . До основних факторів вибору належать [12]:

– складність та масштаб проекту: для простої системи може бути достатньо використовувати мову програмування Python та бібліотеки машинного навчання. Для більш складних систем з великими наборами даних та високими вимогами до обчислювальних ресурсів може знадобитися розглянути хмарні платформи та відповідні інструменти;

– вимоги до продуктивності: Якщо система повинна генерувати складні прогнози в режимі реального часу, може знадобитися використовувати спеціалізовані технології, наприклад потокові платформи (Apache Kafka, Amazon Kinesis) або ж інструменти для машинного навчання з низькою затримкою.

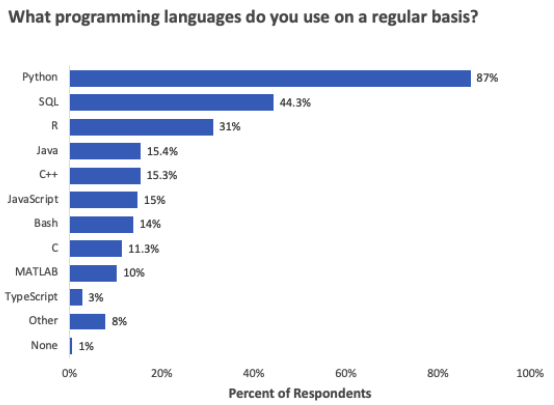

Рисунок 3.12 – Опитування про основну мову програмування серед Data Scientists/ ML Engineers

Першим було прийнято рішення про використання мови програмування Python для написання усієї бази коду застосунку. Враховуючи безліч переваг цієї мови програмування та значну популярність у сфері Data Science та

Machine Learning що підтверджується великою кількістю статистичних даних (рис. 3.12) можна вважати цей вибір повністю виправданим [13].

Далі було розглянуто основні фреймворки для Python, адже у сучасному світі фреймворк – це центральний елемент кожного веб-застосунку що визначає обмеження та переваги системи що розробляється.

Django – це повноцінний фреймворк для розробки веб-додатків на Python, який пропонує широкий спектр функцій, включаючи систему ORM (Object-Relational Mapping), систему шаблонів, адміністративну панель та багато іншого.

Переваги Django - фреймворк відомий своєю простотою, надійністю та масштабованістю. Він добре підходить для розробки складних вебзастосунків, таких як сайти онлайн магазинів, системи управління контентом (CMS) та соціальні мережі.

Flask – це легкий і гнучкий фреймворк для веб-розробки на Python. Він пропонує базові компоненти для створення веб-додатків, залишаючи розробникам більше контролю над архітектурою.

Переваги Flask: ідеально підходить для простих веб-додатків, API та прототипування. Його гнучкість дозволяє розробникам використовувати різні інструменти та бібліотеки.

Рисунок 3.13 – Популярні фреймворки Python

FastAPI – це високопродуктивний фреймворк для створення API на Python. Він поєднує в собі функції веб-фреймворку з можливостями валідації даних, документування та генерації Swagger.

Переваги FastAPI: ідеально підходить для розробки RESTful API, які потребують високої продуктивності та чіткої документації.

Проте враховуючи специфіку проєкту а саме аналіз даних та прогнозування часових рядів та прийняте рішення про використання монолітної архітектури через масштаб системи було прийнято рішення про використання бібліотеки-фреймворку Streamlit.

Streamlit – це бібліотека Python, спеціально розроблена для швидкого створення веб-застосунків, орієнтованих на завдання з аналізу даних [14]. До основних особливостей бібліотеки відноситься:

– streamlit дозволяє перетворювати Python-скрипти на інтерактивні веб-застосунки за лічені хвилини;

– розробнику не потрібно володіти HTML, CSS або Javascript для розробки фронтенду;

– streamlit зосереджується на робочих процесах аналізу даних, що робить його ідеальним для створення інструментів дослідження даних, панелей керування, інтерфейсів для моделей машинного навчання (ML) та інших програм, орієнтованих на дані.

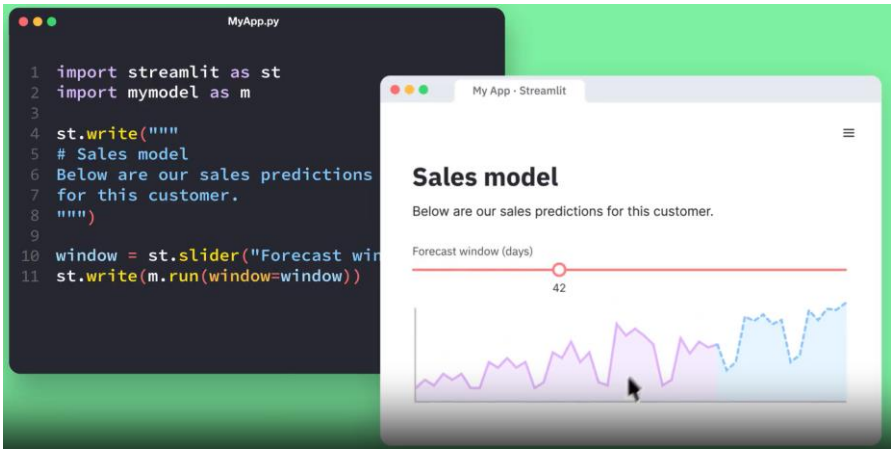

Рисунок 3.14 – Приклад роботи бібліотеки Streamlit

Також слід зазначити що ця бібліотека імплементує клієнт-серверну архітектуру просто інкапсулюючи усю логіку представлення.

Також було обрано декілька основних бібліотек що будуть використовуватися для розробки основного функціоналу застосунку.

Fbprophet – це бібліотека з відкритим кодом для прогнозування часових рядів, розроблена Facebook. Вона використовує статистичні методи та машинне навчання для створення прогнозів майбутніх значень часових рядів. fbprophet добре підходить для прогнозування різноманітних показників, таких як продажі, трафік вебсайту та ціни [15]. Методи цієї бібліотеки дозволяють робити досить швидкі прогнозування що вигідно виділяє її на фоні інших бібліотек.

Pandas – це потужна бібліотека Python для роботи з даними. Вона пропонує структури даних, подібні до таблиць БД, такі як DataFrame, для зберігання та маніпулювання даними. Pandas також надає різноманітні функції для очищення, перетворення та аналізу даних. Та загалом є потужним та дуже популярним інструментом для будь яких задач пов’язаних з великими об’ємами даних (Data Science, Data Engineering, Machine Learning) та надає можливість використовувати DataFrame як абстракцію над безліччю табличних джерел даних [16].

Plotly – це бібліотека для візуалізації даних. Вона дозволяє створювати різноманітні типи діаграм: лінійні, гістограми та точкові діаграми. Особливо корисними є інтерактивні функції plotly, які дозволяють користувачам досліджувати дані більш детально [17].

Yfinance – це популярна бібліотека Python з відкритим кодом, яка дозволяє легко завантажувати історичні та динамічні дані про фінансові ринки з вебсайту Yahoo Finance. Враховуючи відсутність безкоштовних планів більшості фінансових API, використання такої бібліотеки-скраперу було визначено виправданим. Також слід зазначити що бібліотека імплементує паттерн кешування тож більшість запитів на історичні данні про ціни акцій

будуть виповнені лише один раз на день, після чого закешовані та використані повторно [18].

Висновки до розділу 3

У третьому розділі було розглянуто та обрано з різноманітних архітектурних рішень для системи що розробляється. Усі рішення були розроблені відповідно до специфікацій вимог до ПЗ що були виявлені у першому розділі. Насамперед було визначено основну архітектуру застосунку, розроблено UML - моделі системи, а саме діаграму класів та діаграму компонентів.

У третій частині розділу було розглянуто розробку макетів інтерфейсу – візуальних прототипів користувацького інтерфейсу. Саме розробка таких макетів дозволить просто розробити представлення застосунку. Також ця модель визначає велику кількість рішень дизайну коду та позбавляє багатьох помилок що можуть виникнути при взаємодії користувача з вебсайтом. Отож було розроблено макети інтерфейсу усіх основних сторінок застосунку.

В останньому підрозділі було досліджено та обрано технічні рішення для розробки ПЗ прогнозування цін. Було обрано мову програмування python та фреймворк Streamlit що ідеально підходить для швидкого створення застосунків для аналізу даних. Також було обрано yfinance як бібліотеку для доступу до історичних даних акцій, pandas для маніпулювання наборами даних та plotly для візуалізації даних.

Отож результатом проведеної у третьому розділі роботи стали конкретні архітектурні та проєктні рішення що дозволить значно спростити розробку, мати чіткі моделі як системи так і користувацького інтерфейсу та розуміти призначення усіх обраних інструментів та технологій.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ПРОГНОЗУВАННЯ ЦІН

Фінальним кроком розробки ПЗ прогнозування цін стала програмна реалізація, що включає у себе такі кроки як налаштування середовища, реалізація бізнес логіки та представлення, отож в цьому розділі розглянуто конкретні технологічні рішення та готові лістинги коду, описано базові класи та продемонстровано результати програмної реалізації.

4.1 Налаштування базового оточення розробки

Першим кроком було налаштовано базове оточення розробки. Уся розробка здійснювалася в середовищі VS Code. Також було вирішено використовувати інструмент для створення віртуального середовища Python.

Venv – це віртуальне середовище Python, яке використовується для ізоляції проектів Python та їхніх залежностей. Це дозволяє мати різні версії Python та бібліотек для різних проектів, не впливаючи на інші проекти або глобально встановлений Python [19]. Деякі переваги цього інструменту:

– ізоляція проектів. Кожен проект має власне середовище з власними бібліотеками. Це запобігає конфліктам між проектами та дозволяє використовувати різні версії бібліотек в одному проекті;

– портативність. Інструмент надає можливість легко переміщувати venv-проект на інший комп'ютер, не турбуючись про те, що він вплине на глобальне встановлення Python на локальній машині;

– управління залежностями. Venv робить зручним керування залежностями, адже надає можливість легко встановити та видалити бібліотеки, необхідні для конкретного проекту, не впливаючи на інші проекти.

Після встановлення на віртуальне середовище усіх необхідних бібліотек було також протестовано бібліотеку streamlit. Для цього в корінному каталозі проєкту було створено простий Python файл та викликано необхідну команду

(рис. 4.1). Streamlit бере на себе всю роботу по створенню представлення з чистого коду Python, компілює його та навіть запускає простий сервер.

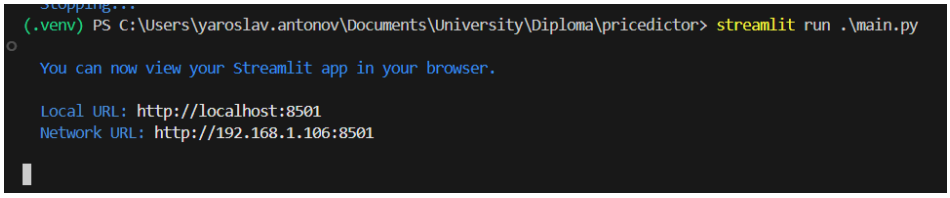

Рисунок 4.1 – Команда запуску застосунку Streamlit

Після вдалої спроби запуску серверу було вирішено розбити директорію проєкту на окремі папки відповідно до діаграми компонентів розробленої в попередньому розділі (рис. 4.2). Директорія .venv автоматично сгенерована та містить віртуальне середовище Python. Бізнес логіку застосунку було винесено в папку core. Data та Presentation відповідно містять класи доступу до даних та логіки презентації. Файл main.py та директорія pages містить самі сторінки що компілюються за допомогою бібліотеки streamlit. Також корінний каталог містить файл auth.yaml для конфігурації автентифікації та yfinance.cache що містить кеш зазначеної бібліотеки фінансового API.

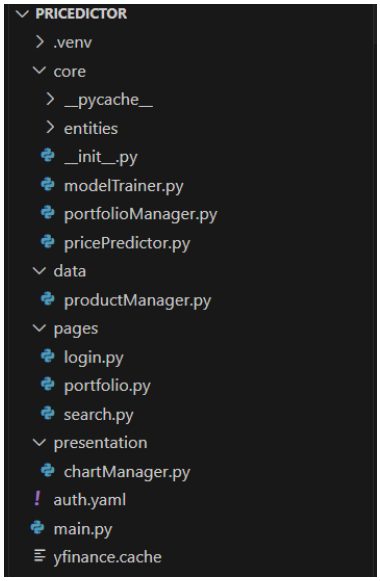

Рисунок 4.2 – Директорія проєкту

4.2 Реалізація основних класів застосунку

Враховуючи основний потік виконання застосунку прогнозування цін першим кроком в розробці мав стати клас що інкапсулює логіку отримання історичних даних про ціни на акції – **ProductManager.**

```
@st.cache_data
  def load_data(_self, ticker: str, period = "10y") -> pd.DataFrame:

    data = yf.download(ticker, period = period)
    data.reset_index(inplace=True)
    return data

  @st.cache_data
  def load_top_products(_self, period = "10y") -> list[pd.DataFrame]:

    products_data = {}
    for stock in _self.stocks[:5]:
      product_data = yf.download(stock, period = period)
      product_data.reset_index(inplace=True)

      products_data[stock] = product_data

    return products_data
```

Наведений лістинг коду демонструє два основних методи цього класу що відповідають за завантаження історичних даних за допомогою бібліотеки yfinance. Методи визначені з декоратором st.cache_data що відповідає за кешування, отож при виклику методу з вже відомими параметрами він не буде відпрацьовувати знову. Окрім параметра за замовченням period що визначає межі історичних даних перший метод також має параметр ticker – тобто ідентифікатор акції. Методи повертають об’єкт типу DataFrame та список DataFrame відповідно.

Наступним логічним кроком алгоритму прогнозування цін на акцію є тренування моделі на зібраних історичних даних, отож було розроблено відповідний клас – **ModelTrainer.**

```
@st.cache_resource
  def train_model(_model, product_data: pd.DataFrame) -> Prophet:

    train_data = product_data[['Date','Adj Close']]
```

```
    train_data = train_data.rename(columns={"Date": "ds", "Adj Close": "y"})

    _model.fit(train_data)

    return _model
```

Цей клас використовує функціонал бібліотеки prophet для тренування моделі. Він отримує історичні дані про ціни та замінює назву необхідних колонок, а саме дату та середню ціну акції за цю дату на «ds» та «y» відповідно. Після цього він викликає метод fit моделі та повертає вже готову для прогнозування модель.

Слід зазначити що за методом fit скривається досить цікава комбінація простих та в той же час потужних статистичних методів машинного навчання [15]. Prophet розбиває часовий ряд на такі складові як:

– тренд: загальний напрямок зміни значень часових рядів у часі;

– сезонність: повторювані коливання значень часових рядів протягом певних періодів (місяців, кварталів тощо);

– шум: випадкові коливання значень часових рядів.

Далі об’єднуючи методи в ансамбль prophet використовує модель авторегресії для тренду, додає до нього сезонні компоненти та вирівнює данні враховуючи шуми.

Отож далі було розроблено сам клас **PricePredictor**, що інкапсулює взаємодію з класом тренування моделі, створює необхідний часовий ряд майбутнього та робить саме прогнозування.

```
def predict_price(self, product_data, period: str) -> pd.DataFrame:

    self.model = ModelTrainer.train_model(self.issue_new_model(), product_data)

    future = self.model.make_future_dataframe(periods=period)

    return self.model.predict(future)

  def predict_popular_prices(self, products_data: list[pd.DataFrame]) ->
list[pd.DataFrame]:

    in_year_date = date.today() + relativedelta(years = 1)
```

```
    forecasts = [['Ticker', in_year_date, 'Growth']]

    for product in products_data:

      model = ModelTrainer.train_model(self.issue_new_model(),
products_data[product])

      forecast = model.predict(pd.DataFrame({'ds': [in_year_date]}))

      growth = (forecast.at[0, 'yhat'] / products_data[product]["Adj
Close"].iloc[-1] - 1) * 100

      forecasts.append([product, forecast.at[0, 'yhat'], growth]  )

    return forecasts
```

Наведені методи з класу PricePredictor відповідають саме за функціонал прогнозування. Важливо зазначити що метод issue_new_model() лише повертає новий об’єкт класу Prophet але необхідний адже кожна окрема модель може бути натренована лише один раз. Головною відмінністю методів окрім логічного прогнозування для одного та для декількох продуктів є також період прогнозування. В той час як перший метод визначає прогноз ціни на кожний день протягом бажаного періоду (параметр period), другий метод розраховує прогноз лише на поточну дату плюс один рік. Також в методі predict_popular_prices присутня логіка розрахунку зміни ціни акції за цей рік (колонка growth).

Наступним кроком в алгоритмі прогнозування цін є саме представлення даних у необхідному вигляді. Саме цю логіку було реалізовано в класі **ChartManager.**

```
class ChartManager:

  def generateChart(model, forecast) -> plt.Figure:

    return plot_plotly(model, forecast)

  def generateList(forecast) -> pd.DataFrame:

    df = pd.DataFrame(forecast)
```

```
    new_header = df.iloc[0]
    df = df[1:]
    df.columns = new_header

    return df
```

GenerateChart – метод що використовує функціонал бібліотеки prophet для створення графіку що поєднує історичні данні та нові, спрогнозовані ціни. Для цього він приймає модель та сам датафрейм прогнозу в якості параметрів та повертає модель класу Figure з бібліотеки plotly.

GenerateList – приймає список прогнозування та робить з нього датафрейм в якому після цього встановлюються назви колонок що відповідають першому рядку вкладеного списку. Таким чином дані прогнозування каталогу популярних товарів стають більш зрозумілішими та легко відображаються в представленні адже мають csv подібну структуру.

Останній клас, що було розроблено – **PortfolioManager**. Цей клас відповідальний за базові операції з інвестиційним портфелем акцій користувача.

```
  def delete_from_portfolio(self, product) -> None:

    self.products.remove(product)

    self.config['credentials']['usernames'][st.session_state['username']]['produc
ts'] = self.products

    with open('auth.yaml', 'w') as file:
        file.write(yaml.dump(self.config, default_flow_style=False))

    st.rerun()

  def add_to_portfolio(self, product) -> None:
    self.products.append(product)

    self.config['credentials']['usernames'][st.session_state['username']]['produc
ts'] = self.products

    with open('auth.yaml', 'w') as file:
        file.write(yaml.dump(self.config, default_flow_style=False))

    st.rerun()
```

В наведеному лістингу представлено два методи цього класу – add_to_portfolio та delete_from_portfolio. Відповідно до назви перший метод додає зазначений у параметрах продукт (акцію) до портфелю а другий метод видаляє. Слід зазначити що методи використовують файл з даними користувачів, та фактично модифікують змінну products поточного користувача, що визначається за допомогою змінної st.session_state. Ця змінна зберігає деякі данні сесії в тому числі і логін автентифікованого користувача.

4.3 Реалізація представлення застосунку

Наступним та фінальним кроком було реалізовано представлення застосунку. Так як для цієї задачі було використано зазначену бібліотеку Streamlit, кожна сторінка фактично компілюється з чистого файлу Python в якому використовуються необхідні компоненти бібліотеки. На цьому етапі важливо також розуміти потік виконання будь якого застосунку Streamlit [20]:

– запуск програми: Сервер запускає Python-скрипт, який містить код Streamlit;

– обробка даних: Бекенд виконує код скрипта, обробляє дані та оновлює стан програми;

– генерація HTML: Бекенд генерує HTML-код для фронтенду на основі стану програми та компонентів Streamlit;

– оновлення фронтенду: HTML-код надсилається до браузера користувача, де оновлюється веб-інтерфейс Streamlit.

Слід зазначити що 2,3 та 4 пункти послідовно змінюють один одного створюючи основний цикл потоку виконання застосунку. Кожний раз коли користувач змінює глобальний стан застосунку за допомогою взаємодії з віджетом (компонентом) наприклад натискаючи на кнопку підтвердження відправки форми – Streamlit повторно компілює весь код зверху вниз. Таким чином основним інструментом керування потоком виконання програми та відображенням необхідного представлення і є самі віджети. Майже кожен

компонент з яким можна взаємодіяти приймає функцію зворотнього виклику – callback в якості параметру on_change (при зміні стану) або ж on_click (при натисканні) або ж взагалі повертає логічне значення. Таким чином розробник може використовувати як і функції зворотнього виклику так і умовні вирази, або ж взагалі комбінувати ці два підходи.

Отож спочатку було розроблено сторінку search що містить головний функціонал прогнозування цін на конкретні акції.

```
with st.form("ProductForm"):

    selected_stock = st.selectbox('Select stock for prediction',
productManager.stocks, None)
    stock_to_search = st.text_input('Or try input ticker by yourself')
    period = st.slider('Years of prediction:', 1, 4) * 365

    submit_button = st.form_submit_button()

    if submit_button:
        if len(stock_to_search) == 0 and selected_stock is None:
            st.error("Please fill in all required fields.")
        else:
            submitted = True
            ticker = stock_to_search or selected_stock

if submitted:
    data_load_state = st.text('Loading data...')
    data = productManager.load_data(ticker)
    data_load_state.text('Loading data... done!')

    st.subheader('Raw data')
    st.write(data.tail())

    plot_raw_data()

    forecast = pricePredictor.predict_price(data, period)

    st.subheader(f'Forecast data for {period // 365} years')

    fig1 = ChartManager.generateChart(pricePredictor.model, forecast)

    st.write(forecast)
```

Наведений лістинг містить основний функціональний фрагмент коду з файлу представлення сторінки Search. Спочатку фрагменту ініціалізовано

форму вибору акції за допомогою компоненту st.form та менеджеру контексту шо визначає межі цієї форми. Форма містить одні з основних компонентів що гарно ілюструють можливості бібліотеки:

– st. selectbox що приймає список об’єктів з яких користувач зможе обрати один;

– st.text_input для вводу текстових даних;

– st.slider для простого вибору числового значення користувачем;

– st.form_submit_button для підтвердження відправки форми.

Також слід зазначити що майже кожен компонент має параметр Label що дозволяє просто проінформувати користувача про головну ціль віджета та досить багато інших параметрів що дозволяють його гнучко налаштувати [20].

Далі код містить просту валідацію що перевіряє введені дані та у разі успіху оновлює стан файлу, що розпочинає процес завантаження даних, їх виведення на представлення, процесс прогнозування та вивід самого прогнозу.

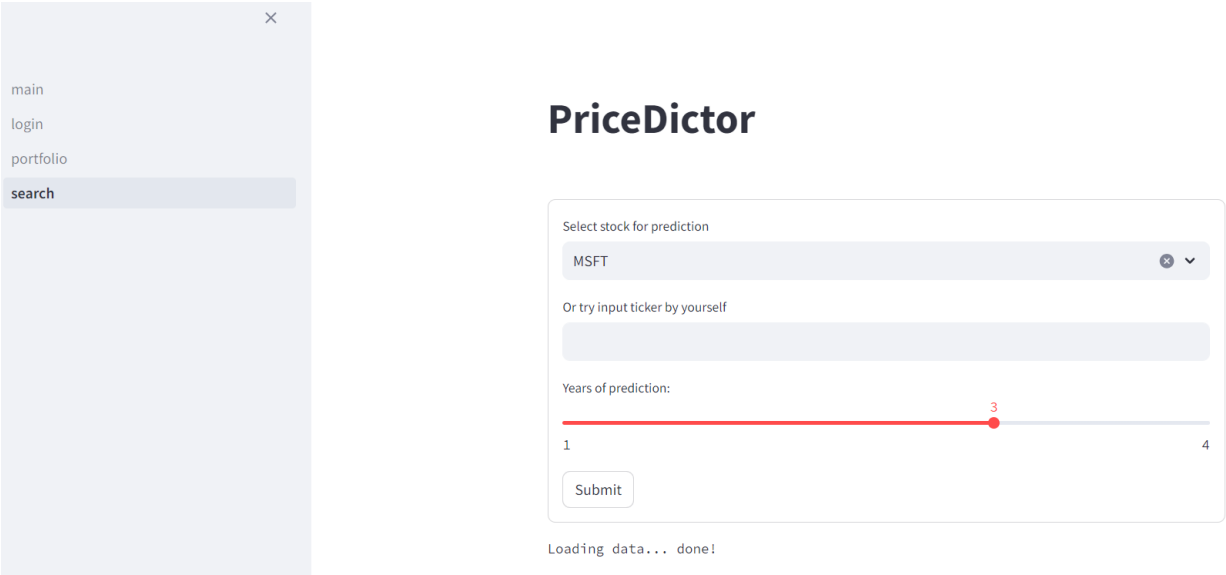

Рисунок 4.3 – Сторінка пошуку з формою

На рисунку 4.3 наведено готовий результат представлення сторінки пошуку а саме частина з формою. На рисунку 4.4 представлено ту ж сторінку проте вже з готовим прогнозом ціни на зазначений період у вигляді графіку.

Графік інтерактивний та надає можливість як змінити масштаб чи межі графіку так і детально отримати інформацію про ціну в певну дату.

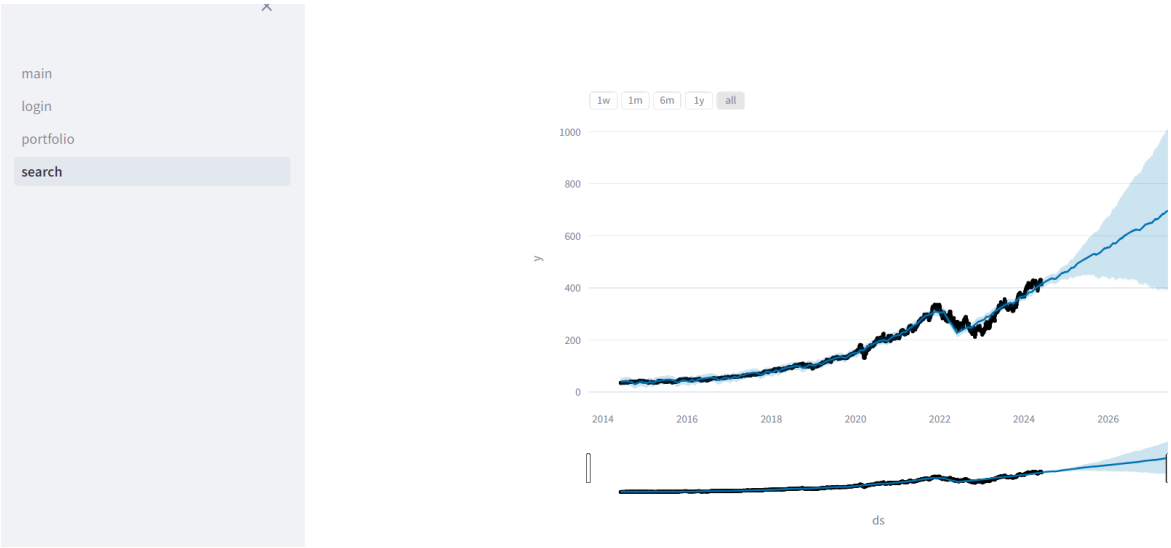

Рисунок 4.4 – Сторінка пошуку з готовим прогнозом

Наступною було розроблено головну сторінку застосунку що надає користувачу можливість отримати прогноз на популярні акції.

```
st.title('PriceDictor')

st.write("This button will show prices for popular stocks in a year")

generate_button = st.button("Generate forecast")

if generate_button:
    data_load_state = st.text('Loading data...')
    data = productManager.load_top_products()
    data_load_state.text('Loading data... done!')

    forecast = pricePredictor.predict_popular_prices(data)

    list = ChartManager.generateList(forecast)

    st.write(list)
```

Основний функціонал реалізовано досить просто та зрозуміло, адже використано мінімальну кількість віджетів та як наслідок коду. Користувач отримує усю необхідну інформацію через інформаційний віджет st.write() та у

разі натискання кнопки застосунок починає генерувати прогноз для каталогу акцій. Отож було створено головну сторінку застосунку (рис. 4.5).

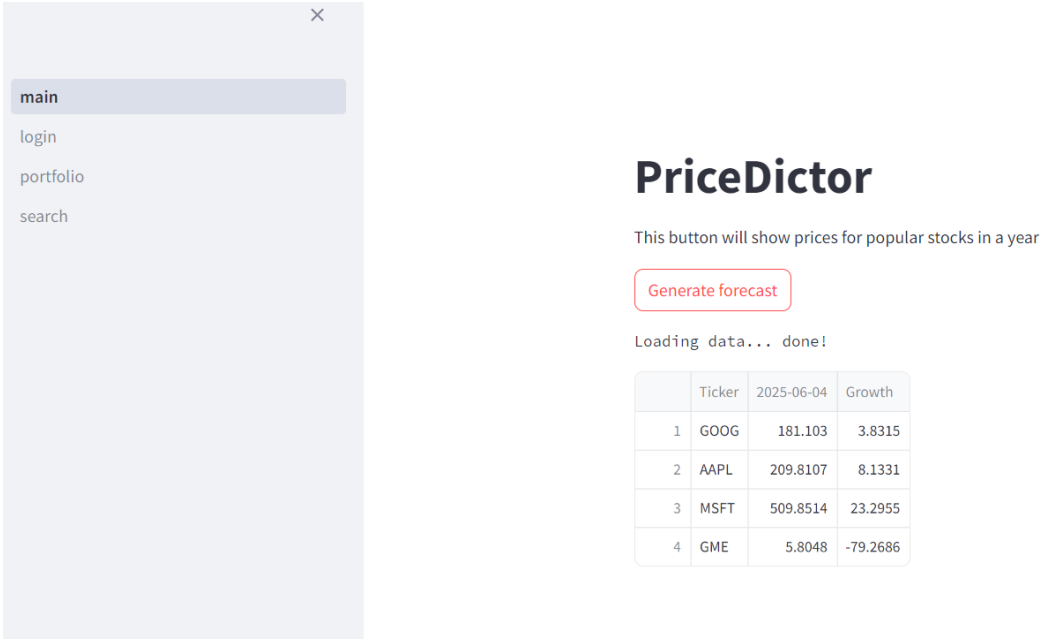

Рисунок 4.5 – Головна сторінка з популярними прогнозами

Наступна сторінка – Login повинна надавати користувачу можливість автентифікуватися та потрапити до власного профілю. Отож на цьому етапі було досліджено та виявлено бібліотеку для автентифікації в застосунках streamlit, а саме streamlit_authenticator [21]. Ця бібліотека надає такі можливості, як:

– вхід користувача: streamlit Authenticator створює форму входу для користувачів, якій дозволяє їм ввести свій логін та пароль;

– перевірка автентифікації: бібліотека перевіряє введені дані користувача використовуючи попередньо визначені облікові данні;

– авторизація: якщо автентифікація успішна, користувачеві надається доступ до визначеного правилами функціоналу Streamlit-застосунку;

– конфігурація: надається можливість налаштувати Streamlit Authenticator, визначивши користувачів, паролі, назву файлу cookie для зберігання стану сеансу тощо.

```
with open('auth.yaml') as file:
    config = yaml.load(file, Loader=SafeLoader)

authenticator = stauth.Authenticate(
    config['credentials'],
    config['cookie']['name'],
    config['cookie']['key'],
    config['cookie']['expiry_days'],
    config['pre-authorized']
)

authenticator.login()

if st.session_state["authentication_status"]:

    authenticator.logout(location='sidebar')
    st.title(f'Welcome {st.session_state["name"]}!')

elif st.session_state["authentication_status"] is False:

    st.error('Username/password is incorrect')

elif st.session_state["authentication_status"] is None:

    st.warning('Please enter your username and password')
```

Наведений лістинг коду сторінки Login чітко ілюструє основні кроки використання бібліотеки для автентифікації користувачів. Спочатку за допомогою завантажувальника конфігураційних файлів `yaml.loader` копіюється зміст основного конфіг файлу. Наступним кроком створюється необхідний об’єкт автентифікації та викликається його метод login() що автоматично створює просту форму автентифікації. Наступні кроки вже в свою чергу визначені для повідомлення користувача про помилку автентифікації чи про її успіх.

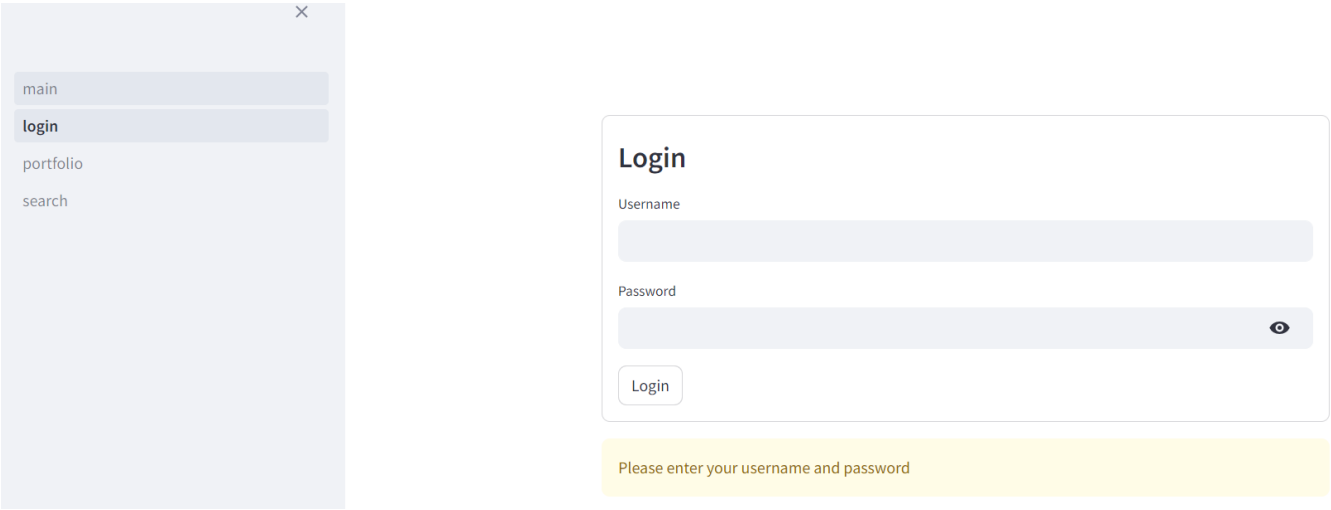

Рисунок 4.6 – Сторінка логіну

Отож остання сторінка яку було розроблено – Portfolio. Вона має містити данні щодо портфелю конкретного користувача та надавати різноманітні можливості.

```
if st.button("Add to portfolio"):
    add_to_portfolio_dialog()
st.divider()
for product in portfolioManager.products:
    row = st.columns(3)

    with row[0]:
        st.write(product)

    with row[1]:
        st.button('Forecast', key= f'{product}_forecast')

    with row[2]:
        if st.button("Delete"
                     #, on_click=delete_from_portfolio(product)
                     , key=f'{product}_delete'
                     , type='primary'):
            portfolioManager.delete_from_portfolio(product)
    st.divider()
```

Із зазначеного лістингу слід виділити використання компонентів st.columns та st.row що добре підходять для позіціонування списку елементів на сторінці. Отож було розроблено сторінку портфелю (рис. 4.7).

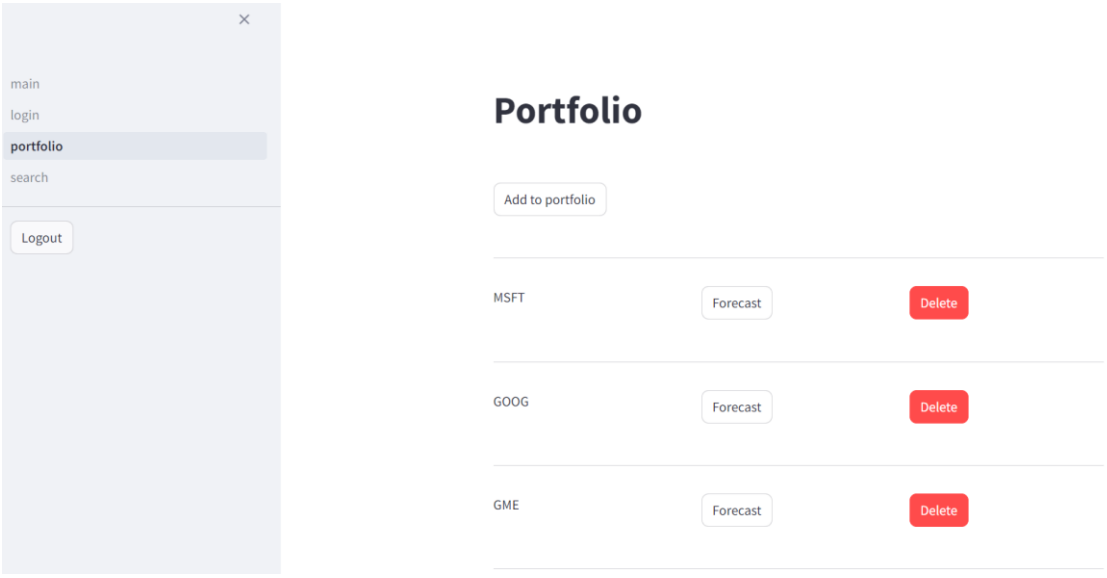

Рисунок 4.7 – Сторінка портфелю користувача

Висновки до розділу 4

В ході виконання четвертого розділу покроково описано усі етапи програмної реалізації застосунку прогнозування цін. А саме налаштування середовища розробки, реалізацію основних класів та реалізацію представлення застосунку. Окрім вербального опису зазначених кроків також було задіяно лістинги коду та зображення готових сторінок веб-застосунку.

Також у розділі було більш детально описано деякі інструменти та бібліотеки, їх основні моделі виконання та функціонал що прихований за досить простими з першого погляду методами. Зокрема описано потік виконання Streamlit та статистичні методи що використовуються для вирішення завдання прогнозування часових рядів.

Отож результатом виконання четвертого розділу стало реалізоване програмне забезпечення прогнозування цін на основі штучного інтелекту. Програмне забезпечення представляє з себе веб-застосунок з монолітною архітектурою. ПЗ відповідає усім зазначеним в першому розділі вимогам, та створено відповідно зі спроєктованими моделями розробленими у другому та третьому розділі.

ВИСНОВКИ

При підготовці кваліфікаційної роботи бакалавра досягнуто основну мету роботи - розробка ПЗ для прогнозування цін на основі ШІ, що спростить процес керування портфелем акцій. Для досягнення поставленої мети виконано ряд завдань пов’язаних з основними етапами створення ПЗ.

В ході виконання кваліфікаційної роботи була проаналізована предметна область та проведено порівняння аналогічних рішень з виділенням переваг та недоліків. Також було сформовано основні функціїї та виявлено конкретні специфікації вимог до розроблюваного програмного забезпечення.

Під час вирішення задач пов’язаних з моделюванням та проєктуванням було розроблено низку проєктних рішень. Насамперед розроблено сценарії використання що описують взаємодію між користувачем та системою в контексті основних функцій застосунку. Також створено графічні моделі, а саме діаграми послідовності та діаграми діяльності що описують потік виконання програми. Було визначено архітектуру та сформовано моделі основних класів системи. Досліджено технічні рішення та інструменти та обрано технологічний стек. Також було побудовано макети користувацького інтерфейсу. Останнім кроком було реалізовано усі необхідні функціональні модулі та представлення застосунку.

Перевагою розробленого застосунку є простота та доступність у використанні. Звичайний користувач незалежно від знань чи наявності підписки може в декілька натискань отримати достатньо швидкий та точний прогноз цін для бажаної акції. В той же час користувач може отримати доступ до функціоналу введення власного портфелю.

Також важливою перевагою слід визначити використання сучасних інструментів та технічних рішень, що роблять розробку веб-застосунків пов’язаних з аналізом даних достатньо простою та в той же час надають можливості для розширення функціоналу та масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fowler M. UML distilled: a brief guide to the standard object modeling language. Pearson Education, Limited, 2018. 208 p.

2. Gomaa H. Software modeling and design: UML, use cases, patterns, and software architectures. Cambridge University Press, 2010. 578 p.

3. Hetler A. Mobile website vs. app: what's the difference?. *WhatIs*. URL: https://www.techtarget.com/whatis/feature/Mobile-website-vs-app-Whats-the-difference (date of access: 08.05.2024).

4. Leffingwell D., Widrig D. Managing software requirements: a use case approach. Pearson Education, Limited, 2003. 544 p.

5. Richards M., Ford N. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020. 432 p.

6. Wiegers K. E., Beatty J. Software requirements (3rd edition) (developer best practices). Microsoft Press, 2013. 672 p.

7. Gollwitzer Z. What is Software Architecture? Beginner Explanation. Full Stack Foundations. URL: https://www.fullstackfoundations.com/blog/what-is-software-architecture (date of access: 04.06.2024).

8. The UML 2 class diagram. IBM Developer. URL: https://developer.ibm.com/articles/the-class-diagram/ (date of access: 04.06.2024).

9. The component diagram. IBM Developer. URL: https://developer.ibm.com/articles/the-component-diagram/ (date of access: 04.06.2024).

10. Uzayr S. b. Mastering UI Mockups and Frameworks: A Beginner's Guide. Taylor & Francis Group, 2022. 230 p.

11. How to Create a Website Mockup – Complete Guide & Examples. Visme Blog. URL: https://visme.co/blog/website-mockup/ (date of access: 04.06.2024).

12. How to Choose a Technology Stack for Web App in 2023. WEZOM: Business Software Development Company. URL: https://wezom.com/blog/how-to-choose-a-technology-stack-for-web-app-in-2023 (date of access: 04.06.2024).

13. 7 Reasons to Learn Python for Data Science. University of San Diego Online Degrees. URL: https://onlinedegrees.sandiego.edu/python-for-data-science/ (date of access: 04.06.2024).

14. Streamlit A faster way to build and share data apps. Streamlit. URL: https://streamlit.io/ (date of access: 04.06.2024).

15. Quick Start. Prophet. URL: https://facebook.github.io/prophet/docs/quick_start.html#python-api (date of access: 04.06.2024).

16. API reference – pandas 2.2.2 documentation. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/docs/reference/index.html (date of access: 04.06.2024).

17. Plotly. Plotly: Low-Code Data App Development. URL: https://plotly.com/python/ (date of access: 04.06.2024).

18. yfinance. PyPI. URL: https://pypi.org/project/yfinance/#quick-start (date of access: 04.06.2024).

19. venv Creation of virtual environments. Python documentation. URL: https://docs.python.org/3/library/venv.html (date of access: 04.06.2024).

20. Streamlit Docs. Streamlit documentation. URL: https://docs.streamlit.io/ (date of access: 04.06.2024).

21. GitHub - mkhorasani/Streamlit-Authenticator: A secure authentication module to validate user credentials in a Streamlit application. GitHub. URL: https://github.com/mkhorasani/Streamlit-Authenticator (date of access: 04.06.2024).