МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Веб-сервіс із прогнозування процесів на фондовому ринку

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Прикладне програмування»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи ПП - 41
Горохов Д.В.
(прізвище та ініціали)

Керівник: Зосімов В.В.
(прізвище та ініціали)
д.т.н., доцент
(науковий ступінь, звання)

Унікальність тексту - 97%

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *прикладних інформаційних систем*
Протокол № 14 від 23 травня 2023р.
зав. кафедри Плескач В.Л.

Київ – 2023

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№з/п	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	14.10.2022	
2.	Видача завдання кваліфікаційної роботи бакалавра	24.10.2022	
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	31.10.2022	
4.	Затвердження плану кваліфікаційної роботи бакалавра	01.11.2022	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	08.11.2022	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	21.12.2022	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2023	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	30.03.2023	
9.	Подання роботи у першому варіанті	28.04.2023	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2023	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2023**	

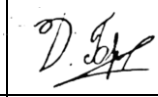
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	26.05.2023	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	12.06.2023	
14.	Захист кваліфікаційної роботи бакалавра	26.06.2023	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини кваліфікаційної роботи бакалавра	Обсяг, арк.
Титульний аркуш	1
Календарний план кваліфікаційної роботи бакалавра	2
Відомість кваліфікаційної роботи бакалавра	1
Анотація	2
Анотація (іноземною мовою-англійською)	1
Зміст	2
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	15
2	25
3	8
Висновок	2
Список використаних джерел	4
Додатки	8

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Горохов Д.В.			Відомість кваліфікаційної роботи бакалавра	Лист	Листів
Керівн.	Зосімов В.В.					
Н/контр.	Макаренко С.А.		26.05.2023			
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Дипломна робота: 75 с., 13 рис., 3 табл., 32 джерел, 3 дод.

Ця дипломна робота присвячена проектуванню та розробленню веб-сервісу з прогнозування процесів на фондовому ринку.

Метою дипломної роботи є якісне прогнозування процесів на фондовому ринку на основі веб-сервісу з прогностичною моделлю для передбачення ціни акцій у короткостроковій перспективі на фондовому ринку.

Для досягнення поставленої мети треба вирішити такі **завдання**:

- провести аналіз основних процесів фондового ринку та способи їх автоматизації;
- здійснити аналіз способів автоматизації процесів фондового ринку;
- порівняти доступні моделі для прогнозування короткострокової ціни акції;
- спроектувати, реалізувати, впровадити веб-сервіс з прогнозування процесів на фондовому ринку з урахуванням інженерії вимог;

Об’єкт дослідження.

Процеси ціноутворення фондового ринку.

Предмет дослідження.

Програмно-технічні, організаційні засади, принципи, підходи щодо побудови веб-сервісу з прогнозування процесів на фондовому ринку.

Методи дослідження.

Метод порівняння підходів операцій з фондовим ринком, емпіричний аналіз, моделювання програмної системи та аналіз літературних джерел і наукових статей.

Ключові слова: машинне навчання, нейронні мережі, мікросервісна архітектура, фондовий ринок, Vue.js, Node.js.

ABSTRACTS

Thesis: 75 p., 13 fig., 3 tables, 32 sources, 3 appendix.

This thesis is devoted to the design and development of a web service for forecasting processes on the stock market.

The purpose of this thesis is to design and implement a web service with a predictive model for predicting stock prices in the short term on the stock market.

To achieve this goal, **the following tasks** need to be solved analyze the main processes of the stock market and ways to automate them; analyze the ways to automate stock market processes; compare the available models for forecasting the short-term share price; to design, implement, and deploy a web service for forecasting processes on the stock market, taking into account requirements engineering;

Object of research.

Operations and pricing of the stock market.

Subject of research.

Software, technical, organizational principles, principles, approaches to building a web service for forecasting processes on the stock market.

Research methods.

Method of comparing approaches to stock market operations, empirical analysis, modeling of the software system and analysis of literary sources and scientific articles.

Keywords: machine learning, neural networks, microservice architecture, stock market, Vue.js, Node.js.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

AI (artificial intelligence) – Штучний інтелект

ML (machine learning) – Машинне навчання

Фінтех. – фінансові технології

MLP (multilayer perceptron) – багатошарові перцептрони

CNN (convolutional neural network) – конволюційні нейронні мережі

RNN (recurrent neural network) – Рекурентні нейронні мережі

DNN (deep neural networks) – Глибокі нейронні мережі

ВСТУП

Фондовий ринок є одним з найбільш динамічних та складних сегментів світової економіки. Зростаюча складність та непередбачуваність цього ринку вимагають постійного аналізу та прогнозування процесів, що дозволяє інвесторам приймати обґрунтовані рішення та мінімізувати ризики.

У зв'язку з цим, веб-сервіси з прогнозування процесів на фондовому ринку набувають все більшої популярності та значущості. Ці сервіси використовують аналітичні методи, статистичні моделі та машинне навчання для збору та аналізу великих обсягів фінансової інформації з різних джерел, зокрема індикаторів ринку, новин, соціальних мереж та економічних показників, що зумовлює **актуальність кваліфікаційної роботи**.

Основними завданнями дослідження є:

- Провести аналіз основних процесів на фондовому ринку та виявити можливості їх автоматизації з використанням сучасних технологій.
- Дослідити різні способи автоматизації процесів на фондовому ринку.
- Розробити веб-сервіс, що здійснює прогнозування процесів на фондовому ринку, з урахуванням інженерії вимог.
- Реалізувати розроблений веб-сервіс і впровадити його для реального використання.
- Провести тестування та оцінку ефективності розробленого веб-сервісу, здійснити аналіз отриманих результатів і внести необхідні виправлення та удосконалення.

Об’єктом дослідження кваліфікаційної роботи бакалавра є процеси ціноутворення фондового ринку..

Предметом дослідження кваліфікаційної роботи бакалавра є розробка веб-сервісу прогнозування ціни акції на фондовому ринку

Методи дослідження: програмно-технічні, організаційні засади, принципи, підходи щодо побудови веб-сервісу з прогнозування ситуації на фондовому ринку.

Практичне значення даної кваліфікаційної роботи бакалавра полягає у тому, що розроблена система може стати зручним та сучасним інструментом для фахівців у галузі фінансів, інвесторів, трейдерів та повсякденних користувачів.

Структура роботи:

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ФОНДОВОГО РИНКУ

1.1 Огляд та аналіз основних концепцій фондового ринку

Фондовий ринок є складною та динамічною галуззю світової економіки, де здійснюється купівля-продаж цінних паперів, таких як акції та облігації. Дослідження та розуміння фондового ринку має велике значення для інвесторів, компаній та економістів, оскільки допомагає прогнозувати цінову динаміку, оцінювати ризики та приймати обґрунтовані фінансові рішення. Для того щоб отримати певну експертизу у сфері автоматизації процесів фондового ринку потрібно провести детальний огляд та аналіз основних концепцій, а саме: ефективність ринку, цінове формування та роль інвесторів.

1.1.1 Ефективність ринку

Концепція ефективності ринку є однією з найважливіших у фінансовій економіці. Вона висуває припущення, що ринок швидко та адекватно враховує всю публічно доступну інформацію при формуванні цін цінних паперів. Рівень ефективності ринку може бути класифікований в три категорії: слабка, напівсильна та сильна ефективність. У слабкій формі ефективності ринку ціни враховують історичні дані, в напівсильній - публічно доступну інформацію, а в сильній - інформацію, недоступну для широкої публіки. Дослідники використовують різні методи та моделі, включаючи тестування на наявність аномалій та арбітражних можливостей, для оцінки рівня ефективності ринку. Існує гіпотеза, яка була запропонована Юджином Фамою у 1970 році, що саме ціни на фінансовому ринку відображають всю доступну інформацію і саме тому неможливо систематично мати прибуток більше, ніж ринок [1].

1.1.2 Цінове формування

Цінове формування є ключовим аспектом фондового ринку, і вивчення його механізмів є необхідним для розуміння ринкової динаміки та прогнозування цін. Традиційні моделі цінового формування, такі як модель раціональних очікувань та модель збиткового гравця, допомагають пояснити роль пропозиції та попиту на ринку, фактори, що впливають на цінову динаміку та утворення рівноваги. Крім того, інновації в галузі фінансових технологій та алгоритмічного трейдингу значно змінили спосіб, яким формуються ціни на фондовому ринку, і вимагають нових підходів до аналізу.

1.1.2.1 Традиційні моделі цінового формування

З історичної перспективи, модель раціональних очікувань і модель збиткового гравця були двома основними теоретичними рамками для розуміння цінового формування на фондовому ринку. Модель раціональних очікувань висуває ідею, що інвестори формують свої очікування щодо майбутніх цін на акції на основі всієї доступної інформації, і ці очікування потім впливають на поточні ціни на ринку [2]. Прикладом цієї моделі може служити ситуація з акціями Apple у 2007 році. Коли компанія анонсувала випуск першого iPhone, інвестори очікували, що це призведе до зростання продажів і прибутків компанії. В результаті цих очікувань ціна акцій Apple зросла ще до того, як продажі нового продукту розпочалися.

Модель збиткового гравця, з іншого боку, концентрується на ідеї, що інвестори здійснюють торгівлю з метою отримання короткострокового збитку, і це поведінка в свою чергу впливає на динаміку цін [3]. Наприклад, ситуація яка відбулася з акціями компанії GameStop на початку 2021 року. Група інвесторів на платформі Reddit вирішила активно купувати акції GameStop, що були в основному зацікавлені у швидкому продажі, але це призвело до масивного збитку

для тих, хто ставив на зниження ціни акцій. Це, у свою чергу, викликало ще більшу покупку акцій і в результаті - швидке зростання ціни акцій GameStop. Ця ситуація демонструє, як поведінка інвесторів, спрямована на отримання короткострокового збитку, може вплинути на ціни на ринку [3].

Обидві ці моделі пояснюють, як фактори пропозиції та попиту, разом з інформацією, яка доступна інвесторам, можуть впливати на формування цін.

1.1.2.2 Вплив фінансових технологій і алгоритмічної торгівлі

З появою фінансових технологій (фінтех) та алгоритмічної торгівлі, процес цінового формування на фондовому ринку став значно складнішим. Нині комп'ютерні алгоритми можуть виконувати велику кількість торгових операцій за дуже короткий час, що може призвести до стрімких змін цін.

Як приклад, трапляються події які називаються "flash crash" (з англійської – *флеш-крах*), коли ринок швидко падає і потім так само швидко відновлюється, вже стали звичним явищем на сучасних ринках і є прямим результатом впливу алгоритмічної торгівлі [4]. Ці нові феномени вимагають розробки нових підходів до аналізу цінового формування на фондовому ринку.

1.1.3 Роль інвесторів і методи оцінювання ризику на фондовому ринку

Інвестори грають важливу роль у фондовому ринку. Вони приймають рішення щодо купівлі, продажу та утримання цінних паперів, враховуючи рівень ризику та потенційний дохід. Використовуючи концепції теорії портфеля або теорії очікуваної корисності, інвестори отримують краще бачення ризиків та потенційних доходів, які пов’язані з інвестиційною можливістю.

Теорія портфеля, яку розробив Гаррі Марковіц в 1950-х роках, пропонує спосіб мінімізації ризику через диверсифікацію інвестицій. Як приклад використання цієї теорії можна взяти інвестора, який хоче здійснити інвестиції в

акції технологічних компаній. Замість того, щоб купити акції лише однієї компанії, він може купити акції декількох компаній з різних секторів технологічної промисловості, щоб зменшити потенційний ризик.

Теорія очікуваної корисності, в свою чергу, є методом оцінки вибору інвестора з урахуванням його ставлення до ризику. Згідно з цією теорією, інвестори вибирають інвестиційні можливості, що максимізують їхню очікувану корисність. Цю теорію можна ілюструвати на прикладі інвестора, який вирішує між двома можливими інвестиціями. Одна інвестиція пропонує високий потенційний дохід, але з високим рівнем ризику, а інша пропонує нижчий потенційний дохід, але з нижчим рівнем ризику. Інвестор може використовувати теорію очікуваної корисності для визначення, яка інвестиція максимізує його очікувану корисність, враховуючи його ставлення до ризику [5] [6].

Огляд та аналіз основних концепцій фондового ринку підтверджують його складність та значущість у глобальній економіці. Ефективність ринку, цінове формування та роль інвесторів є важливими аспектами, які впливають на динаміку фондового ринку та інвестиційні рішення. Дослідження та розуміння цих концепцій допомагають інвесторам, компаніям та економістам приймати обґрунтовані рішення та управляти ризиками на фондовому ринку.

1.2 Аналіз процесів та операцій на фондовому ринку

Центральне місце в розумінні операцій з фінансовим ринком займає процес арбітражу. Арбітраж - це торговельна політика, яка спрямована на використання цінових розбіжностей між активами. Більш цікавими, ніж сама політика, є її непередбачувані наслідки, а саме: який вплив вони мають для прогнозування цін на активи у передбачувані моделі [7]. Іншими процесами, які використовуються на фондовому ринку, є механізми торгівлі, вплив інвестицій та роль економічних та політичних факторів.

1.2.1 Визначення арбітражу

Арбітраж є важливою складовою фондового ринку та його економіки, бо він відіграє значущу функцію в забезпеченні ефективності та ліквідності ринку. Основна ідея арбітражу полягає в тому, що існують ринкові ситуації, коли один і той же актив може бути проданий або куплений за різними цінами на різних біржах або ринках. Арбітражники виявляють такі відмінності цін і виконують відповідні операції для отримання безризикового прибутку. Це може включати одночасну покупку та продаж активу або використання похідних інструментів для здійснення операцій арбітражу. Роль арбітражу полягає в розвитку ефективності ринку, оскільки він вирівнює ціни на різних ринках, що призводить до зменшення арбітражних можливостей [8]. Коли арбітражні можливості з'являються, арбітражники виконують операції, що призводять до вирівнювання цін та зменшення невикористаних можливостей прибутку. Це сприяє підтримці ефективності ринку, забезпечує рівновагу між попитом та пропозицією активів та сприяє швидкій реакції ринку на нову інформацію.

1.2.2 Роль економічних та політичних факторів

Визначення ролі політики у міжнародних фінансових ринках є викликом, оскільки держава має обмежені можливості контролювати або спрямовувати потоки капіталу без значних втрат. Саме тому політичний фактор має суттєвий вплив на рівень ризику, прибутковісті та ліквідністі ринку. Зміни у фінансовому регулюванні, податкові політики, торгові війни, політична нестабільність та конфлікти можуть викликати коливання на ринку та впливати на інвестиційні рішення.

Економічні фактори, такі як стан макроекономіки, економічний зріст, рівень безробіття, інфляція та процентні ставки, теж впливають на фондовий ринок.

Наприклад, позитивні макроекономічні показники можуть підвищити довіру інвесторів, що може сприяти зростанню цін акцій. Зворотній вплив може спостерігатися у випадку негативних економічних показників. Крім того, зміни в процентних ставках можуть впливати на інвестиційні рішення, зокрема вибір між акціями та іншими фінансовими інструментами.

1.3 Визначення потреби в автоматизації процесів фондового ринку

На сучасному фондовому ринку автоматизація вирішує численні проблеми, які пов'язані із швидкістю обробки інформації, прозорістю і ефективністю угод. Цей процес може включати алгоритмічну торгівлю, електронні платформи для виконання операцій, автоматизовані системи управління ризиками. Системи такого типу можуть обробляти величезні обсяги даних, реагувати на зміни ринкових умов в миттєвому режимі та підвищувати ефективність фондового ринку. Окрім цього, автоматизація допомагає знижувати ризики, пов'язані з людськими помилками, та поліпшує відповідність стандартам регулювання. Наприклад, у 2021 році обсяг світового ринку алгоритмічної торгівлі оцінювався в 15,55 мільярда доларів США, і очікується, що в період з 2022 по 2030 рік він зростатиме зі середньорічним темпом зростання (CAGR) на 12,2% [10]. Це дає нам зрозуміти, що ефективність автоматизації, особливо прояв її у вигляді алгоритмічної торгівлі, є доведеною.

Однак, не дивлячись на численні переваги, автоматизація створює нові виклики та ризики. Зокрема, розробка, тестування та використання алгоритмічних стратегій вимагає високої кваліфікації та досвіду, а дисфункції в цих системах можуть швидко поширитися і призвести до значних ринкових наслідків [4].

1.4 Аналіз існуючих технологій та методологій автоматизації на фондовому ринку

В сучасному світі, автоматизація є ключовим інструментом для управління великими обсягами даних, особливо в такій області, як фондовий ринок. Використання автоматизованих систем дозволяє аналізувати динаміку цін на цінні папери, розраховувати ризики, визначати вигідні інвестиції та виконувати інші важливі завдання.

1.3.1 Технології автоматизації

Алгоритмічна торгівля або торгівля з використанням комп'ютерних алгоритмів стала широко розповсюдженою на фондовому ринку. Такі системи використовують складні математичні моделі для виконання операцій купівлі та продажу на фондових біржах. Вони здатні реагувати на зміни ринку швидше за людину, що дозволяє збільшити прибуток та знизити ризики. Штучний інтелект та машинне навчання також використовуються для автоматизації фондового ринку. Ці технології дозволяють створювати прогнози на основі аналізу великих обсягів історичних даних. Штучний інтелект може виявляти закономірності та тенденції, які можуть бути непомітні для людини.

1.3.1.1 Алгоритмічна торгівля

Алгоритмічна торгівля, відома також як автоматизована або квантова торгівля, використовує передові математичні моделі для визначення оптимального моменту для входу та виходу з ринку. Алгоритми бувають різноманітними і враховують ряд параметрів, включаючи ціну, час, обсяг та історичні дані. Основні переваги алгоритмічної торгівлі включають:

— швидкість виконання операцій: комп'ютерні алгоритми можуть виконувати торговельні операції значно швидше людей, що дозволяє використовувати короткочасні ринкові можливості.

— ефективність: алгоритмічна торгівля зменшує вплив людської помилки, покращуючи точність торговельних операцій.

— систематичність: застосування алгоритмічної торгівлі дозволяє уникнути емоційних рішень, які можуть негативно вплинути на результат торгівлі.

1.3.1.2 Штучний інтелект та машинне навчання

Штучний інтелект (AI) та машинне навчання (ML) дедалі активніше використовуються у сфері фінансів для автоматизації різних процесів, включаючи аналіз даних, прогнозування ринку та алгоритмічну торгівлю. AI та ML можуть аналізувати великі набори даних та визначати складні шаблони та тенденції, що недоступні для людського сприйняття. Завдяки цьому, AI та ML можуть використовуватись для побудови прогнозних моделей, які можуть передбачати майбутню поведінку ринку на основі історичних даних. Деякі AI/ML алгоритми, які використовуються в фінансах, включають нейронні мережі, алгоритми посиленого навчання, алгоритми класифікації, регресії та інші. Ці алгоритми можуть бути навчені розпізнавати та використовувати складні шаблони в даних для виконання різноманітних завдань, від прогнозування цін на акції до визначення кредитоспроможності клієнтів. Але, як і в будь-якій технології, AI та ML мають свої обмеження та потенційні ризики. Наприклад, моделі AI/ML можуть навчитися не тільки корисних шаблонів, але й шуму в даних, що може призвести до перенавчання моделі та погіршити її загальну здатність передбачати. Також, AI та ML моделі можуть бути вразливими до маніпуляцій з даними, що може призвести до неточних прогнозів.

1.3.2 Методології автоматизації

Однією з основних методологій автоматизації на фондовому ринку є технічний аналіз. Він включає в себе використання статистичних індикаторів для аналізу торгових даних, таких як ціна та обсяг.

Технічний аналіз допомагає визначити тренди та можливі точки обертання на ринку [11]. Він зосереджується на вивченні історичних даних про ціни та обсяги торгів для прогнозування майбутніх цінових трендів. Технічний аналіз застосовує статистичні індикатори, такі як середні ціни за певний період, рівні підтримки та опору, а також вивчення графіків цін для визначення можливих сценаріїв руху цін.

Фундаментальний аналіз є іншою важливою методологією, яка використовується для автоматизації на фондовому ринку. Він зосереджений на вивченні економічних показників, таких як прибуток компанії, ставки проценту, політичні події, які можуть вплинути на вартість акцій. Фундаментальний аналіз використовується для визначення внутрішньої вартості акцій та визначення чи є акція переоціненою або недооціненою на ринку. Здійснюючи глибоке дослідження фундаментального аналізу, науковці з Гарвардського університету визначили, що стратегії, засновані на вивченні ключових економічних показників - таких як поточні запаси, дебіторська заборгованість, валовий прибуток, витрати на збут, капітальні інвестиції, ефективні податкові ставки, методи інвентаризації, кваліфікація аудиторів та продуктивність праці - можуть призвести до відносно високої аномальної дохідності. Після створення відповідних портфелів, вони дійшли до висновку, що середня кумулятивна аномальна дохідність, скоригована за розміром, становила 13,2 відсотка протягом 12-місячного періоду [12].

Системи автоматизованого управління портфелем акцій використовують обидва види аналізу для створення і виконання інвестиційних стратегій. Вони

можуть автоматично купувати та продавати акції на основі заданих параметрів, враховуючи ризики та потенційні прибутки.

1.4 Застосування машинного навчання та інших технологій штучного інтелекту для автоматизації процесів фондового ринку

У сфері фондового ринку використання систем штучного інтелекту зростає з кожним роком. Ці машинні системи, залежно від рівня автономності, можуть аналізувати великі обсяги різноманітних даних, включаючи "великі дані", та здійснювати прогнозування, рекомендації та прийняття рішень відповідно до заданих цілей. Вони базуються на моделях ML, які можуть самостійно навчатися на великих обсягах даних, не потребуючи явного програмування. Застосування штучного інтелекту на фондовому ринку допомагає автоматизувати процеси і поліпшити прийняття рішень. Наприклад, в управлінні активами використовуються системи штучного інтелекту для аналізу даних про ринкові тренди та показники компаній з метою прийняття обґрунтованих інвестиційних рішень.

Алгоритмічна торгівля використовує моделі машинного навчання для автоматичного виконання торгових операцій на основі аналізу ринкової динаміки та показників. Кредитний андеррайтинг також може бути автоматизованим за допомогою систем штучного інтелекту, які швидко аналізують кредитну історію та фінансові показники для оцінки ризиків.

Розвиток технологій штучного інтелекту, доступність великих обсягів даних та зростання обчислювальних потужностей стимулюють впровадження цих систем у фінансові послуги на фондовому ринку. Вони дозволяють ефективніше використовувати дані та робити точніші прогнози, що сприяє покращенню якості прийняття рішень та зменшенню ризиків.

1.4.1 Застосування машинного навчання та штучного інтелекту в алгоритмічній торгівлі

Одним з важливих застосувань AI і машинного навчання на фондовому ринку є алгоритмічна торгівля. Цей підхід до торгівлі використовує програмні алгоритми для виконання торгових операцій на основі заданих параметрів, таких як ціна, час або обсяг. Це дозволяє трейдерам встановити конкретні правила для входу та виходу з угод, що максимізує їх прибутки та мінімізує ризики. Машинне навчання та AI дозволяють створювати складні алгоритми, які можуть враховувати велику кількість різних факторів і швидко адаптуватися до змінюваних ринкових умов. Наприклад, алгоритми машинного навчання можуть бути навчені аналізувати історичні дані та визначати залежності між різними змінними. Вони можуть використовувати цю інформацію для передбачення майбутніх трендів ринку і визначення найбільш вигідного моменту для входу або виходу з угоди. Популярним прикладом застосування AI та машинного навчання в алгоритмічній торгівлі є використання нейронних мереж. Нейронні мережі можуть моделювати складні шаблони і взаємозв'язки в даних, що дозволяє їм передбачати майбутні тренди та цінові зміни. Вони можуть бути навчені визначати кореляції між ціною акції та різними макроекономічними індикаторами, щоб передбачати, як ціна акції зміниться у відповідь на зміни в економіці.

1.4.1.1 Типи нейронних мереж та їх характеристики

Нейронні мережі – це моделі, що імітують спосіб роботи людського мозку, щоб ідентифікувати шаблони та взаємозв'язки в даних. Є різні типи нейронних мереж, кожна з яких має свої властивості та використання.

1. Перцептрони: Це найпростіший тип нейронної мережі, який складається з одного або кількох шарів нейронів, з'єднаних в одному напрямку.

Перцептрони можуть вирішувати прості задачі, що не вимагають великої кількості вхідних даних або складних вирахувань. Наприклад, вони можуть бути використані для прогнозування цін акцій на основі набору макроекономічних індикаторів.

2. Багатошарові перцептрони (MLP): Вони мають один або кілька прихованих шарів між вхідним та вихідним шарами, що дозволяє їм моделювати складніші залежності. MLP також широко використовуються для розв'язання складних задач прогнозування та класифікації.
3. Конволюційні нейронні мережі (CNN): Цей тип нейронної мережі спеціально розроблений для обробки зображень. CNN можуть виявляти ключові особливості зображення та використовувати їх для ідентифікації шаблонів. Хоча вони найчастіше використовуються в комп'ютерному зору, CNN також можуть бути корисними для аналізу часових рядів або будь-яких інших даних, які мають просторову структуру.
4. Рекурентні нейронні мережі (RNN): Вони мають унікальну здатність "запам'ятовувати" попередні вхідні дані, що робить їх особливо корисними для аналізу часових рядів або текстових даних. Наприклад, RNN можуть використовуватися для прогнозування змін цін акцій на основі історичних даних про ціни та торговий обсяг.
5. Глибокі нейронні мережі (DNN): Це тип нейронних мереж, який включає два або більше прихованих шарів нейронів. Ці мережі здатні виконувати складні обчислення та виявляти важкі для виявлення шаблони в даних. Вони часто використовуються для розв'язання складних проблем в області машинного навчання, таких як класифікація зображень, розпізнавання мови та природний обробка мови.

1.4.2 Виклики та обмеження використання штучного інтелекту на фондовому ринку

При використанні штучного інтелекту і технологій машинного навчання на фондовому ринку виникають значні виклики та обмеження. Одним з них є складність інтерпретації результатів моделей машинного навчання, особливо глибокого навчання. Багато моделей AI, таких як глибокі нейронні мережі, часто відомі своєю "чорною коробкою" - природою, коли точні процеси в мережі незрозумілі для користувача [13]. Це створює проблему, коли треба розуміти, чому певна модель приймає ті чи інші рішення, особливо в контексті високоризикових сфер, таких як фондовий ринок.

Крім того, використання AI та машинного навчання потребує великої кількості даних для навчання моделей. Збір, обробка та зберігання цих даних може бути дорогою та складною задачею, особливо в світлі строгих регулятивних вимог щодо конфіденційності даних. Зокрема, можливість перенавчання, коли модель надто добре вивчає тренірovочний набір даних і погано справляється з новими, невідомими даними, є ще одним значним обмеженням машинного навчання в цій області [14].

Існує також проблема змінності фондового ринку. Несприятливі глобальні економічні події можуть вплинути на поведінку ринку, а історичні дані можуть виявитися менш корисними для прогнозування майбутніх трендів. Це часто вимагає перегляду та регулярного оновлення моделей машинного навчання, що може бути трудомістким процесом.

Особливо існує ризик занадто великої залежності від AI та автоматизації. У випадку помилок або непередбачених обставин, системи, які повністю залежать від AI, можуть призвести до значних втрат. До цього можна додати етичні питання

щодо автоматизації та втрати робочих місць, які виникають зі збільшенням застосування AI.

1.4.3 Майбутнє штучного інтелекту в контексті фондового ринку

Штучний інтелект і технології машинного навчання вже привнесли значні зміни на фондовий ринок, але їх потенціал далекий від вичерпного.

У майбутньому, використання AI може стати ще більш впливовим, особливо з розвитком нових методів машинного навчання і збільшенням обчислювальної потужності. Зокрема, можливість AI аналізувати соціальні медіа та інші джерела неструктурованих даних також може мати значний вплив на фондовий ринок. Вже зараз існують технології, які використовують AI для аналізу настроїв у соціальних медіа, що може бути корисним для прогнозування поведінки ринку. Що стосується регулятивних змін, з'являються нові можливості для використання AI для підтримки справедливості і прозорості на фондовому ринку. Наприклад, регулятори можуть використовувати AI для виявлення шахрайства або зловживання на ринку.

Таким чином, у першому розділі ми провели детальний аналіз застосування машинного навчання та штучного інтелекту у сфері фондового ринку, а також їх потенційного впливу на автоматизацію його процесів. Було проаналізовано ключові види машинного навчання, включаючи нейронні мережі та підсилювальне навчання, та їх застосування у контексті фінансових ринків.

Також ми розглянули можливі виклики та обмеження, з якими може зіткнутися використання машинного навчання та штучного інтелекту в цій сфері. Особливу увагу було приділено ризикам перенавчання, нестабільності та непередбачуваності моделей. Попри це, перспективи штучного інтелекту та машинного навчання у сфері фондового ринку є великими, зокрема у контексті

автоматизації і оптимізації торгівельних стратегій, зменшення ризику, покращення точності прогнозування та забезпечення більшої ефективності операцій.

У наступному розділі ми розглянемо процес проектування та реалізації веб-сервісу, що використовує ці технології для передбачування поведінки фондового ринку.

РОЗДІЛ 2 ПРОЕКТУВАННЯ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ ДЛЯ ПЕРЕДБАЧЕННЯ ПОВЕДІНКИ ФОНДОВОГО РИНКУ

2.1 Опис та аналіз потреб користувачів

Опис та аналіз потреб користувачів є невід'ємною частиною процесу проектування веб-сервісу і відіграє важливу роль у досягненні успіху проекту. Аналіз потреб користувачів дозволяє зрозуміти їх вимоги, очікування та проблеми, що вони стикаються. Це відкриває можливість створити веб-сервіс, який відповідає реальним потребам користувачів і забезпечує їм зручність, ефективність та задоволення. Аналіз потреб користувачів дозволяє виявити ключові функціональні вимоги, визначити пріоритетність функцій та визначити оптимальні шляхи взаємодії з користувачами. Цей процес допомагає зменшити ризик невдачі проекту та сприяє створенню веб-сервісу, який буде успішно використовуватися та задовольняти потреби своїх користувачів.

2.1.1 Планування та аналіз потреб користувачів: методи і важливість в процесі створення веб-сервісу

Методи дослідження потреб користувачів можуть бути різними. Вони включають як кількісні методи, такі як анкетування і аналіз даних про користувачів, так і якісні методи, такі як інтерв'ю, спостереження і аналіз використання. Анкетування є широко використовуваним методом дослідження, який дозволяє швидко зібрати інформацію від великої кількості людей [16]. За допомогою анкет можна дізнатися про інтереси, вподобання, проблеми та очікування користувачів. Інтерв'ю дозволяють отримати більш глибоке розуміння потреб користувачів [17]. Вони включають безпосередню комунікацію з користувачами і дозволяють поставити додаткові питання, уточнити відповіді та дізнатися про контекст використання веб-сервісу. Спостереження за користувачами дозволяють дізнатися про їх реальну поведінку при використанні

веб-сервісу. Це може включати в себе аналіз журналів використання, проведення тестування з використанням і так далі.

Процес планування є основою для визначення вимог до веб-сервісу і проектування його інтерфейсу. Він допомагає забезпечити, що веб-сервіс буде відповідати потребам користувачів і забезпечувати їм високу якість обслуговування. Також він допомагає забезпечити, що веб-сервіс буде зручним у використанні і допоможе користувачам ефективно виконувати їх задачі. В результаті аналізу потреб користувачів повинні бути визначені конкретні вимоги до веб-сервісу, такі як вимоги до функціональності, зручності використання, доступності, ефективності, надійності і так далі.

2.1.2 Визначення потреб користувачів

Після визначення значущості процесу планування ми виявили такі ключові потреби користувачів:

1. Швидкий доступ до інформації: користувачам потрібен сервіс, який надасть швидкий доступ до ринкової інформації.
2. Передбачення ринку: користувач очікує від веб-сервісу здатність передбачення короткострокової ціни акції.
3. Простота використання: веб-сервіс має бути дружнім до користувача та простим у навігації. Інтерфейс повинен бути інтуїтивно зрозумілим.
4. Відслідковування інвестицій: користувач очікує, що веб-сервіс предоставить можливість для створення свого власного “портфоліо”, що надасть можливість відслідковувати кількість свої інвестицій.

Задля ілюстрації потреб користувачів та їх взаємозв’язку з функціональністю веб-сервісу, можна навести User-Flow діаграму.

User-flow діаграма - це візуалізація шляху, який користувач проходить при використанні веб-сервісу, від початкової точки контакту до завершення обраного

завдання. Вона зазвичай складається з ряду кроків, які відповідають діям користувача, та вузлів, які відповідають ключовим моментам взаємодії [18].

Цей інструмент є важливим у процесі планування веб-сервісу, оскільки він допомагає дизайнерам та розробникам краще розуміти, як користувачі взаємодіють з сервісом. Він також дозволяє виявити можливі проблеми в інтерфейсі або структурі сервісу на ранніх стадіях розробки.

Із допомогою user-flow діаграми можна визначити критичні точки в інтерфейсі, які вимагають особливої уваги, та оцінити, наскільки ефективно користувач може досягти своїх цілей. Особливо це стає актуальним для веб-сервісів в сфері фондового ринку, де швидкість та ефективність можуть бути критично важливими для успішної взаємодії.

На наступному зображенні представлена user-flow діаграма для веб-сервісу з прогнозуванні цін на фондовому ринку.

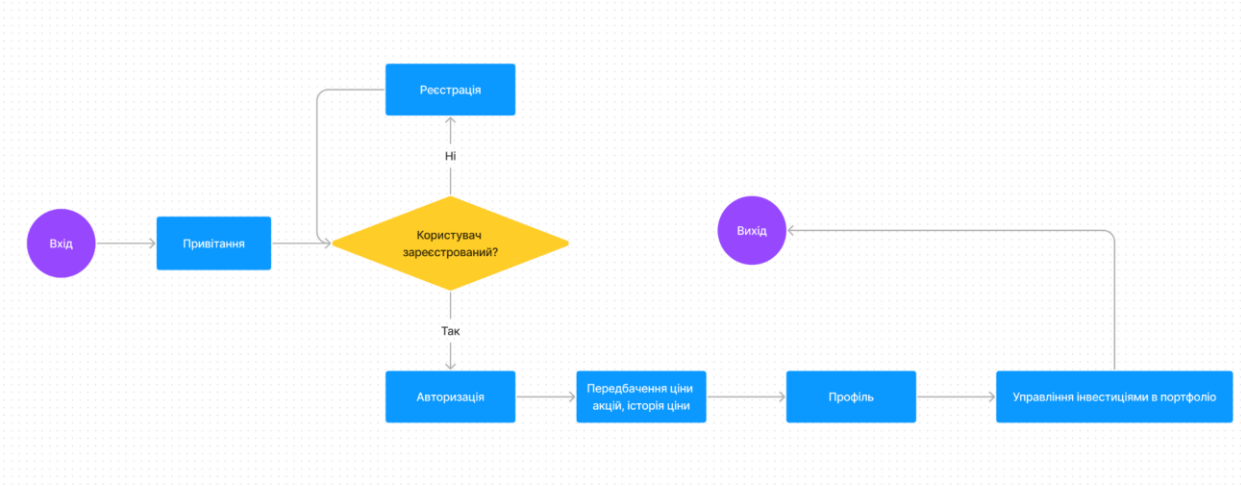


Рисунок 2.1 — User-flow діаграма

2.2 Вибір та обґрунтування використання технологій фронт-енду та бек-енду

У сучасному світі вибір технологій для фронт-енду та бек-енду є критично важливим етапом розробки веб-сервісі. Правильний вибір технологій має значний вплив на продуктивність, масштабованість, безпеку та зручність використання

веб-додатків. Оптимальні технології дозволяють створити потужні та ефективні системи зі швидким завантаженням, зручним інтерфейсом та високою надійністю. Крім того, правильно обрані технології сприяють масштабуванню проекту та легкій інтеграції з іншими системами. Враховуючи це, важливо ретельно досліджувати та обґрунтовано вибирати технології, що відповідають конкретним вимогам та цілям проекту.

2.2.1 Аналіз сучасних технологій фронт-енду

Фронт-енд – це частина веб-сервісу, яку безпосередньо бачить і з якою взаємодіє користувач. Останніми роками з'явилось багато різних технологій для розробки фронт-енду, які мають свої переваги та недоліки. Розглянемо найпопулярніші з них.

1. HTML/CSS/JavaScript: це основні технології, на яких базується будь-який веб-сервіс. HTML використовується для створення структури сторінки, CSS - для стилізації, а JavaScript - для динамічної взаємодії з користувачем.
2. React.js [19]: це JavaScript-бібліотека для побудови користувацьких інтерфейсів, розроблена Facebook. React.js використовує віртуальний DOM, що дозволяє збільшити продуктивність за допомогою оптимізованих перерендерів.
3. Vue.js [20]: це прогресивний JavaScript-фреймворк для створення користувацьких інтерфейсів. Vue.js відрізняється своєю простотою та гнучкістю, він дозволяє легко інтегруватись в існуючі проекти.
4. Angular [21]: це повноцінний фреймворк для розробки веб-застосунків від Google. Angular використовує TypeScript як основну мову програмування і має потужну систему для роботи з формами і валідації даних.
5. Svelte [22]: це новий JavaScript-фреймворк, який перетворює компоненти в ефективний імперативний код, що взаємодіє безпосередньо з DOM.

6. Webpack: це потужний інструмент для збирання та упаковки ресурсів веб-сайтів. Він дозволяє розробникам створювати складні фронтенд додатки, які складаються з різних типів файлів, таких як JavaScript, CSS, зображення та інші.
7. Vite.js: це сучасна технологія для розробки веб-інтерфейсів та логіки застосунку, яка надає швидку, ефективну та прогресивну розробку фронтенду. Вона відрізняється від традиційних інструментів збирання, таких як Webpack, завдяки своїй швидкості та швидкій перезавантаженні сторінок під час розробки. Vite.js використовує передачу модулів на етапі розробки, що полегшує швидкість розробки та дозволяє бачити зміни в реальному часі без необхідності повного перезавантаження сторінки.

2.2.2 Аналіз архітектурних рішень фронт-енду

Під час розробки веб-сервісу важливим є не тільки вибір конкретних технологій, але і розробка ефективної архітектури, яка визначає структуру проекту, взаємодію між компонентами і загальний підхід до розробки.

Розглянемо вплив правильно обраної архітектури на різні аспекти веб-сервісу у таблиці наведеній нижче:

Таблиця 2.1 — Фактори та вплив правильно обраної архітектури

Фактори	**Вплив**
Спрощення розробки	Правильна архітектура створює чітку структуру проекту, що дозволяє команді розробників більш ефективно взаємодіяти і розробляти продукт.
Підтримка та масштабованість	Система, що добре структурована і має відповідну архітектуру, легше адаптується до змін, включаючи додавання нових функціональних можливостей, а також підтримує

Продовження таблиці 2.1

	масштабування продукту.
Продуктивність	Архітектура впливає на продуктивність веб-сайту, як для розробників, так і для кінцевих користувачів.
Розробка і тестування	Коректна архітектура дозволяє краще ізолювати компоненти для юніт-тестування і спрощує процес розробки, оскільки розробники можуть працювати над різними компонентами системи паралельно.
Безпека	Вибір технології та архітектури може вплинути на безпеку продукту.

Визначивши важливість архітектури розглянемо декілька ключових способів реалізації:

— Компонентний підхід: На сьогоднішній день майже всі сучасні фреймворки використовують компонентний підхід для розробки інтерфейсів. Компонент - це самостійна одиниця інтерфейсу, яка має власний стан, логіку та представлення. Велика програма складається з багатьох компонентів, які можуть взаємодіяти між собою.

— Стейт-менеджмент: Для управління станом компонентів та їх взаємодії використовуються спеціальні бібліотеки, такі як Redux, Pinia або MobX. Вони дозволяють створити централізоване сховище даних, до якого можуть звертатися всі компоненти.

— Маршрутизація: Для навігації між різними сторінками або розділами веб-сервісу використовуються роутери. Сучасні фреймворки надають вбудовані рішення для маршрутизації, такі як react-router для React.js або vue-router для Vue.js.

— Асинхронні операції: Для виконання асинхронних операцій, таких як запити до сервера, використовуються спеціальні методики та інструменти. Наприклад, в JavaScript для цього використовуються проміси і async/await.

2.2.3 Обґрунтування вибору технологій фронт-енду

Розробка фронт-енду веб-додатку вимагає обережного вибору технологій, що максимально відповідають потребам проекту та забезпечують необхідну продуктивність та зручність для користувачів. Вибір технологій фронт-енду для цього проекту був зроблений з урахуванням їх ефективності, гнучкості, відкритості та масштабованості:

— Vue.js дозволяє розробити високопродуктивний веб-застосунок з компонентною архітектурою. Це означає, що кожен елемент інтерфейсу користувача може бути віддільним компонентом, який можна повторно використовувати та тестувати незалежно від інших. Крім того, Vue.js має вбудовану підтримку реактивності, що означає, що інтерфейс користувача автоматично оновлюється при зміні даних.

— Pinia використовується як інструмент для управління станом у Vue.js проєктах. Він пропонує простий та інтуїтивно зрозумілий API для управління станом сайту, включаючи зберігання даних, керування станом та діями. Використання Pinia допоможе забезпечити ефективне управління даними і підвищить продуктивність додатка.

— Vue-router – це офіційний маршрутизатор для Vue.js. Він дозволить легко створювати односторінкові веб-сайти, використовуючи знайомі концепції URL, такі як шляхи та параметри. Vue-router інтегрується безпосередньо з Vue.js, що спрощує розробку.

— Axios допомагає робити HTTP-запити до сервера. У контексті нашого проекту, Axios буде використовуватися для взаємодії з бек-ендом та отримання даних про ринок, а також для відправки даних про дії користувачів.

— Tailwind CSS – це утилітарний CSS-фреймворк, який надає низькорівневі утиліти для створення замісних стилів напряму у розмітці. Це допоможе уникнути непотрібного CSS-коду та спростить процес створення відповідного дизайну.

— Vite – це інструмент для побудови, який створений для надання швидкої та легкої розробки. Він використовує ESBuild для надзвичайно швидкої перебудови та HMR (Hot Module Replacement). Це зробить процес розробки швидким і ефективним, що є важливим для проекту.

Обрання фреймворку Vue.js як основи для розробки веб-сервісу має суттєвий вплив на архітектурні рішення. Зокрема, однією з ключових особливостей Vue.js є його компонентний підхід до створення інтерфейсу користувача.

Це означає, що весь інтерфейс будується з використанням модульних, повторно використовуваних компонентів. Кожен компонент визначає свій власний HTML, CSS, і JavaScript, який використовується для реалізації його функціональності. Компоненти можуть включати інші компоненти, створюючи таким чином ієрархію.

Компонентний підхід дозволяє уніфікувати код, робити його більш зрозумілим та легким для підтримки. Компоненти можна використовувати багатократно в різних частинах додатка, що підвищує ефективність розробки.

Особливо це стає важливим у випадку складних проектів, як наш веб-сервіс, де існує значна кількість різних елементів інтерфейсу, що потребують спільної логіки. Тому, з урахуванням цих переваг, було прийнято рішення про використання Vue.js та його компонентної архітектури.

2.2.4 Аналіз сучасних технології бек-енду

Вибір бек-енд технологій є не менш важливим етапом під час проектування веб-сервісу, ніж вибір фронт-енд технологій. Це стосується не тільки серверної платформи, але й вибору фреймворків, бібліотек та інших інструментів, які будуть використані для реалізації серверної частини додатку. Сучасні технології бек-енду включають в себе такі рішення як Node.js, Fastify та bcrypt:

— Node.js є популярною платформою, що використовує JavaScript для серверної розробки. Використання Node.js пропонує ряд переваг. Зокрема, це дозволяє забезпечити високу продуктивність та масштабованість за рахунок асинхронної природи Node.js. Крім того, розробникам, вже знайомим з JavaScript на стороні клієнта, буде легше освоїти JavaScript на стороні сервера.

— Fastify [23] – це високопродуктивний фреймворк веб-додатків для Node.js, що забезпечує швидке маршрутизування, підтримку плагінів та низькі накладні витрати. Fastify розроблений для підтримки HTTP/2 та HTTP/3, а також має декларативну підтримку JSON Schema, що дозволяє використовувати стандартизований формат для опису структури JSON даних.

— bcrypt – це бібліотека для Node.js, яка дозволяє забезпечувати безпечне хешування паролів. Вона використовує алгоритм bcrypt для створення криптографічно стійких хешів паролів, що забезпечує високий рівень захисту від атак на основі підбору паролей.

— Express.js – це один з найпопулярніших фреймворків для розробки веб-додатків на Node.js. Express.js пропонує простий та ефективний спосіб створення серверних додатків, забезпечуючи зручну маршрутизацію, обробку запитів та підтримку середовища розробки.

— Django – це високорівневий фреймворк розробки веб-додатків на мові програмування Python. Django надає вбудовану адміністративну панель, готові

компоненти для роботи з базами даних, автентифікацію, маршрутизацію та інші важливі функціональні можливості [24].

— Spring Boot – це фреймворк для розробки веб-додатків на мові програмування Java [25]. Spring Boot пропонує швидкий старт та простоту використання, дозволяючи розробникам швидко створювати ефективні серверні додатки з використанням Spring Framework.

— Ruby on Rails – це фреймворк розробки веб-додатків на мові програмування Ruby [26]. Ruby on Rails надає зручний спосіб створення динамічних та масштабованих серверних додатків, використовуючи конвенції перед налаштуваннями.

— Flask – це мінімалістичний фреймворк для розробки веб-додатків на мові програмування Python. Flask дозволяє швидко створювати прості серверні додатки та API, забезпечуючи гнучкість та легкість використання.

2.2.5 Архітектурні підходи до реалізації бек-енду

У реалізації бек-енду веб-сервісу використовуються різні архітектурні підходи, серед яких основні - це мікросервісна архітектура та монолітна архітектура. Мікросервісна архітектура передбачає розбиття веб-сервісу на набір незалежних, самостійно функціонуючих компонентів (мікросервісів), які взаємодіють між собою за допомогою API. Кожен мікросервіс може бути реалізований та масштабований окремо, що забезпечує гнучкість та швидкість розробки. З іншого боку, монолітна архітектура полягає у розгортанні веб-сервісу як єдиного, великого модуля, в якому всі компоненти взаємодіють між собою без використання зовнішніх інтерфейсів. Вибір між цими двома архітектурними підходами залежить від потреб проекту та вимог щодо масштабованості, швидкості розробки, підтримки та інші фактори.

2.2.5.1 Монолітна архітектура

Монолітна архітектура є традиційним підходом до розробки бек-енд додатків, в якому весь функціонал зосереджений в одному монолітному додатку. Вона має свої переваги та недоліки.

Переваги:

— Простота у використанні та розробці. Монолітний додаток має одну кодову базу, що спрощує його розгортання та підтримку.

— Висока продуктивність. Монолітна архітектура дозволяє ефективно використовувати ресурси сервера, оскільки весь функціонал працює в одному процесі.

— Простота масштабування. При збільшенні навантаження можна просто збільшити обсяг ресурсів сервера.

Недоліки:

— Низька модульність. Монолітна архітектура не дозволяє легко розділити функціонал на окремі модулі, що може ускладнити розробку та тестування.

— Обмежена масштабованість. При зростанні обсягу функціоналу або навантаження, можуть виникати проблеми з продуктивністю та масштабованістю монолітного додатку.

2.2.5.2 Мікросервісна архітектура

Мікросервісна архітектура є сучасним підходом до розробки бек-енд додатків, в якому функціонал розбивається на набір невеликих та незалежних сервісів, що працюють разом через API. Кожен мікросервіс відповідає за певну функціональну область.

Переваги:

— Висока модульність. Мікросервісна архітектура дозволяє розділити функціонал на окремі сервіси, що спрощує розробку, тестування та підтримку.

— Гнучкість та швидкість розгортання. Кожен сервіс може розгортатись незалежно, що дозволяє швидко впроваджувати зміни та вдосконалення.

— Легка масштабованість. Можливість горизонтального масштабування окремих сервісів дозволяє збільшувати навантаження на певні компоненти системи.

Недоліки:

— Складність управління. Мікросервісна архітектура вимагає високого рівня управління та координації сервісами, що може бути викликом для команди розробників.

— Збільшений обсяг комунікації. Застосування мікросервісної архітектури призводить до збільшення обсягу комунікації між сервісами, що може вплинути на продуктивність та час відповіді системи.

Таблиця 2.2 — Порівняння мікросервісної та монолітної архітектури

Параметер	Монолітна архітектура	Мікросервісна архітектура
Модульність	Низька	Висока
Масштабованість	Проста	Гнучка
Управління	Просте	Складне
Комунікація	Мінімальна	Збільшена

2.2.6 Обґрунтування вибору технологій бек-енду

В процесі розробки веб-сервісу для передбачення поведінки фондового ринку були обрані наступні технології: Node.js, Fastify та Bcrypt. Кожна з цих технологій принесе певні користі та внесе вагомий внесок у розробку та функціональність проекту. Розглянемо їх детальніше.

Node.js є потужною платформою для розробки серверних додатків, що базуються на JavaScript. Вибір Node.js для бек-енду нашого веб-сервісу дозволить забезпечити наступні переваги:

— Висока продуктивність: Node.js працює на однопотоковому та подієвому принципі, що дозволить ефективно обробляти багато запитів одночасно і забезпечувати швидку відповідь користувачам.

— Широкі можливості співпраці з фронтендом: Використання JavaScript як на бек-енді, так і на фронтенді, спростить комунікацію між клієнтом та сервером, спільне використання коду і забезпечить однорідний досвід розробки.

Fastify – це легкий, ефективний і швидкий фреймворк для створення серверних додатків на Node.js. Вибір Fastify має наступні переваги для нашого проекту:

— Висока продуктивність: Fastify пропонує оптимізовану обробку запитів і відповідей, що дозволить забезпечити високу швидкість роботи сервера.

— Легкість використання: Fastify має простий та інтуїтивний API, що спростить розробку і підтримку коду.

— Масштабованість: Fastify дозволить легко масштабувати додаток та розширювати його функціональність, що є важливим для майбутнього розвитку веб-сервісу.

Bcrypt є криптографічною бібліотекою, яка використовується для шифрування та хешування паролів користувачів. Вибір Bcrypt принесе наступні переваги для нашого проекту:

— Безпека: Bcrypt використовує солі та медлені алгоритми хешування, що зробить паролі більш стійкими до розкриття в разі витоку бази даних.

— Легкість використання: Bcrypt має простий API, що дозволяє легко застосовувати шифрування паролів у додатку.

Firestore є розподіленою базою даних в реальному часі, розробленою компанією Google, яка надає потужні можливості для зберігання та синхронізації даних в хмарному середовищі. Вона побудована на базі документ-орієнтованої моделі даних, що дозволяє зберігати дані в колекціях документів, які містять поля та значення. Firestore надає високу доступність та масштабованість, що робить його ефективним інструментом для розробки веб-додатків з великим обсягом даних та вимогами до швидкості.

Основні переваги Firestore включають:

— Реальний час: Firestore підтримує миттєву синхронізацію даних між клієнтськими пристроями та сервером, що дозволить користувачам бачити оновлення даних в режимі реального часу без необхідності оновлення сторінки.

— Гнучкість: Firestore дозволяє легко організувати дані в колекції та документи, а також використовувати різні типи даних, такі як рядки, числа, масиви, об'єкти тощо. Це робить його придатним для різноманітних веб-додатків з різними потребами.

— Швидкість: Firestore володіє високою продуктивністю завдяки своїй розподіленій архітектурі та оптимізації запитів. Це дозволить ефективно взаємодіяти з великим обсягом даних та отримувати результати запитів миттєво.

— Масштабованість: Firestore автоматично масштабується для задоволення вимог високих завантажень та обсягу даних. Це дасть змогу зберігати мільйони документів та ефективно обробляти запити.

— Безпека: Firestore надає можливості для налаштування прав доступу до даних на рівні колекцій та документів. Ви можете контролювати, які користувачі мають доступ до яких даних, забезпечуючи високий рівень конфіденційності та безпеки.

Для реалізації мікросервісу, відповідального за передбачення ціни, було обрано фреймворк Flask.

Flask є легким і простим у використанні фреймворком для розробки веб-додатків на мові програмування Python. Його вибір обґрунтовується кількома факторами та перевагами:

— Простота використання: Flask має чистий та простий синтаксис, що дозволить швидко розробити веб-сервер, який буде використовуватись як незалежний мікросервіс.

— Гнучкість: Flask надає велику свободу в організації структури та архітектури свого проекту, що дозволить зосередитись саме на реалізації нейронної мережі і якісного її використання.

Додатково до Flask, також використовується Docker для ізоляції мікросервісу передбачення ціни. Docker дозволить упакувати сервіс та всі його залежності в контейнер, що забезпечить консистентне та незалежне від середовище виконання. Використання Docker спростить розгортання, масштабування та управління сервісом, а також забезпечить його незалежність від конкретної операційної системи та серверного середовища.

2.3 Проектування архітектури веб-сервісу

Веб-сервіс є складною системою, і важливим етапом його розробки є проектування архітектури. Архітектура веб-сервісу визначає його структуру, компоненти, способи взаємодії, розподіл функцій та інші аспекти, які визначають його працездатність і ефективність. Проектування архітектури має велике значення, оскільки воно впливає на подальшу розробку, підтримку та розширення веб-сервісу.

Мікросервісна архітектура була обраною для проектування архітектури веб-сервісу з метою досягнення гнучкості, швидкості розробки та масштабованості. В цій архітектурі функціональність веб-сервісу розділена на невеликі, незалежні мікросервіси, які можуть бути розгорнуті та масштабовані окремо [26][27].

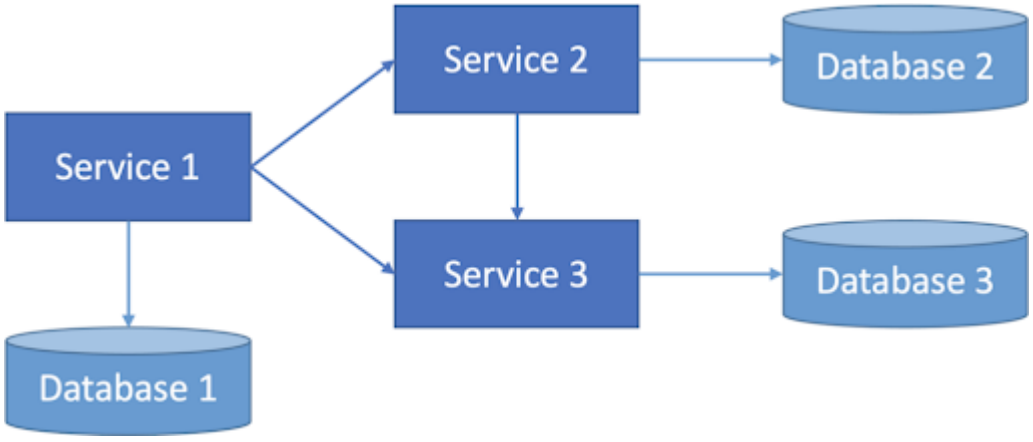


Рисунок 2.2 — Приклад мікросервісної архітектури

Основна ідея мікросервісної архітектури полягає в тому, щоб кожен мікросервіс виконував лише одну конкретну функцію або послугу. Кожен мікросервіс може бути розгорнутий і масштабований окремо від інших, що дозволяє швидше впровадження змін, незалежну розробку та масштабування лише необхідних компонентів системи. Через використання мікросервісної архітектури веб-сервіс стає більш модульним, що полегшує розробку та тестування. Кожен мікросервіс може бути написаний мовою програмування або використовувати технології, які найкраще підходять для виконання його конкретної функції. Це забезпечує гнучкість та швидкість в розробці нових функцій та можливість оновлення окремих компонентів без впливу на решту системи.

Архітектура веб-сервісу включає такі компоненти:

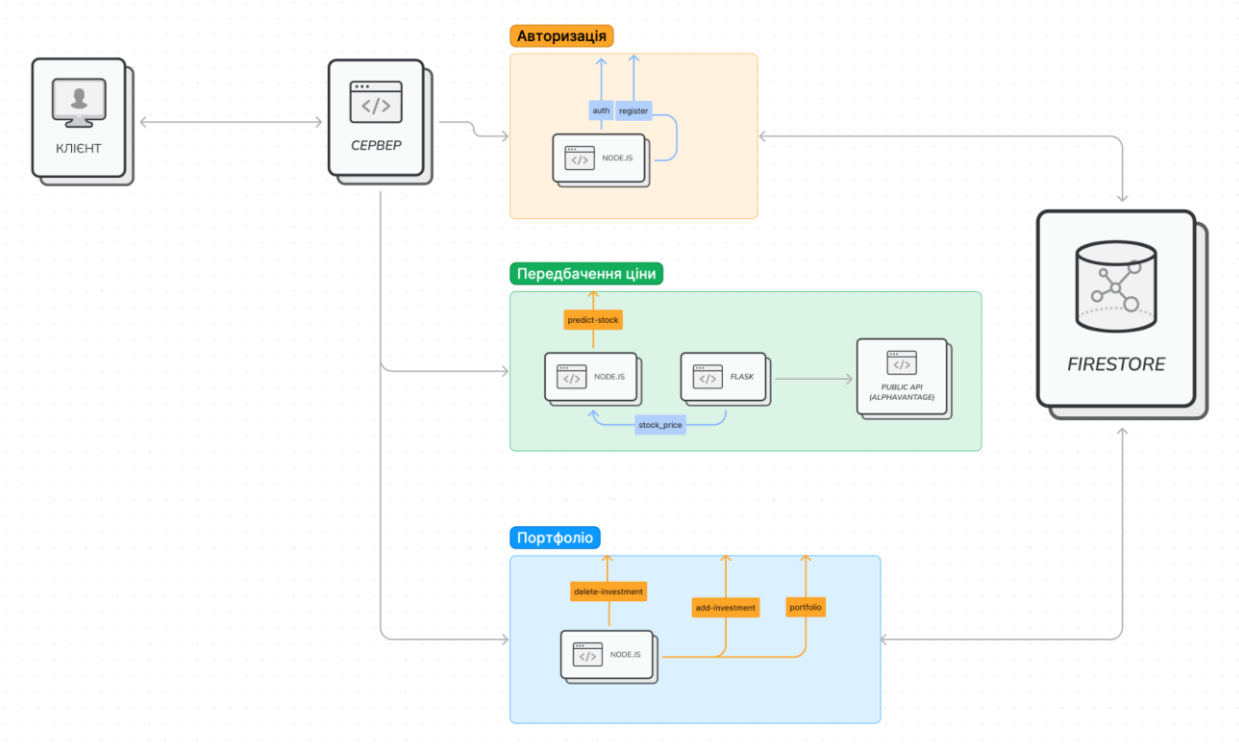
1. Клієнтський інтерфейс: Для реалізації клієнтського інтерфейсу використовується фреймворк Vue.js, який забезпечує швидкий та ефективний рендеринг сторінок, компонентну структуру та простоту в розробці. Для стилізації та оформлення інтерфейсу використовується бібліотека Tailwind CSS.

2. Серверний шар: Для реалізації серверної частини використовується Node.js, який забезпечує швидке та ефективне виконання серверного коду. В

якості фреймворку для розробки API використовується Fastify, який забезпечує високу продуктивність та низьку витрату пам'яті.

3. База даних: Для зберігання та управління даними використовується Firestore. Вона надає гнучкість, швидкодію та масштабованість для потреб веб-сервісу.

Нижче наведена діаграма архітектури веб-сервісу, яка ілюструє взаємодію



компонентів та поток даних:

Рисунок 2.3 — Діаграма мікросервісної архітектури веб-сервісу

Ця архітектура веб-сервісу забезпечує модульність, масштабованість та надійність системи. Вона дозволяє розділити функціональність на окремі компоненти, які можуть працювати незалежно один від одного, спрощуючи розробку та підтримку системи. Крім того, використання потужних технологій,

таких як Node.js, Fastify, Flask та Docker, забезпечує ефективну роботу та швидкість веб-сервісу.

2.4 Проектування моделі машинного навчання для передбачення цін акцій фондового ринку

Проектування такого мікросервісу є нетривіальною задачею, яке вимагає уваги до деталей та правильного підходу. Одна з основних особливостей полягає у виборі правильної моделі, яка відповідатиме поставленій задачі та має високу точність прогнозування. Вибір моделі залежить від різних факторів, таких як тип даних, доступність навчальної вибірки, обсяг даних та складність задачі. При проектуванні моделі також важливо враховувати можливі перешкоди, такі як висока волатильність ринку або непередбачуваність факторів, що впливають на ціни акцій. Всі ці аспекти потребують внимання та глибокого аналізу для досягнення надійних та точних результатів в передбаченні цін акцій фондового ринку.

2.4.1 Вибір типу моделі машинного навчання

Вибір конкретного типу моделі має велике значення, оскільки різні типи моделей можуть мати різні характеристики та ефективність в залежності від задачі та характеристик даних. Для обґрунтування вибору типу моделі проведено дослідження та аналіз різних моделей машинного навчання, зокрема лінійних моделей, дерев'яних моделей та нейронних мереж [28].

2.4.1.1 Аналіз типів моделей машинного навчання

1. Перший тип моделей, що розглядається - лінійні моделі, базуються на лінійній комбінації ознак і використовуються для простих задач. Вони мають просту структуру та ефективність, але їх потужності обмежені лінійною залежністю між ознаками. Враховуючи це, лінійні моделі можуть

бути підходящим варіантом для задач з невеликою кількістю ознак та простими залежностями між ними.

2. Другий тип моделей - дерев'яні моделі, засновані на деревовидній структурі розгалужень . Вони відмінно підходять для задач класифікації та регресії, оскільки можуть враховувати нелінійні залежності між ознаками. Дерев'яні моделі здатні розбити простір ознак на підділи та приймати рішення, використовуючи поріги та правила. Ці моделі можуть бути ефективними, коли маємо багато ознак та складніші залежності між ними [29].
3. Третій тип моделей - нейронні мережі, які є найбільш потужними та гнучкими моделями машинного навчання. Вони складаються зі штучних нейронів, з'єднаних у шари, та використовуються для вирішення складних задач з великою кількістю даних. Нейронні мережі можуть автоматично виявляти складні залежності між ознаками та адаптуватись до нових даних. Вони мають широкі можливості, але вимагають більше обчислювальних ресурсів та даних для тренування.

Таблиця 2.3 — Порівняння типів моделей

Тип моделі	Сфери реалізації
Лінійна	Використовуються для апроксимації лінійних залежностей між ознаками та ціною акцій. Вони показують позитивні результати, коли залежність між ознаками та ціною є прямолінійною. Проте, при наявності складних нелінійних залежностей лінійні моделі можуть бути менш ефективними.

Продовження таблиці 2.3

Дерев’яна	Рішучі дерева та випадкові ліси, добре справляються зі складними нелінійними залежностями. Вони вирішують проблему лінійної неперервності шляхом розбиття простору ознак на різні регіони. Кожен регіон має своє правило прийняття рішень, що дозволяє дерев'яним моделям добре узгоджуватись зі складними нелінійними залежностями. Однак, вони також можуть бути схильними до перенавчання, особливо якщо надто глибокі та складні.
Нейронні мережі	Здатні автоматично виявляти та узагальнювати складні нелінійні залежності між ознаками та ціною акцій. Застосування нейронних мереж дозволяє створити глибоку архітектуру з багатьма шарами, що забезпечує додаткові можливості у моделюванні залежностей. Недоліком нейронних мереж є, нейронні мережі вимагають багато

Кінець таблиці 2.3

	даних для тренування та велику обчислювальну потужність для їхнього тренування та використання.

Зробивши детальний аналіз можливих моделей машинного навчання та способів досягнення мети передбачення цін акцій фондового ринку можна зробити вибір щодо моделі, яка буде використовуватись у застосунку.

2.4.1.2 Обґрунтування вибору моделі

Було обрано модель типу Sequential з використанням LSTM (Long-Short Term Memory) архітектури. Вибір цієї моделі зумовлений кількома факторами, які роблять її найбільш відповідною для потреб даної задачі [30].

1. Модель типу Sequential: Це є одним з основних типів моделей у глибокому навчанні, який використовується для побудови нейронних мереж. Вона представляє собою послідовну структуру шарів, де вихід одного шару передається як вхід до наступного шару. Це означає, що інформація проходить через модель від початкового вхідного шару до кінцевого вихідного шару без будь-яких зворотних зв'язків або перехресних зв'язків між шарами.
2. Обробка послідовних даних: LSTM архітектура є варіантом рекурентної нейронної мережі, спеціально розробленої для роботи з послідовними даними, такими як часові ряди. Це дозволяє моделі ефективно аналізувати та моделювати динаміку зміни вартості акцій в залежності від попередніх спостережень.
3. Здатність до виявлення складних залежностей: LSTM алгоритм володіє здатністю до моделювання довготривалих залежностей між даними. В

контексті фондового ринку, де ціни на акції можуть бути під впливом різних факторів та тенденцій, важливо мати модель, яка може виявити та уважно аналізувати ці складні залежності.

4. Збереження контексту: LSTM мережі мають внутрішні структури, що дозволяють зберігати контекстну інформацію протягом тривалого періоду часу. Це дозволяє моделі враховувати важливість попередніх спостережень на різних відрізках часу та використовувати їх для прогнозування майбутньої вартості акцій [31].
5. Ефективність та швидкодія: Модель типу Sequential заснована на простій послідовній структурі, що дозволяє їй працювати швидко та ефективно. Це особливо важливо для роботи з великими обсягами даних на фондовому ринку, де швидка обробка та аналіз є критичними факторами.
6. Переваги збереження даних навчання: Обрана модель LSTM забезпечує збереження даних навчання, що дає можливість використовувати їх без необхідності повторного навчання моделі. Це значно економить час та ресурси, особливо в умовах змінності та швидкого оновлення даних на фондовому ринку.

На графічному зображенні нижче показана загальна архітектура моделі Sequential для прогнозування цін акцій фондового ринку:

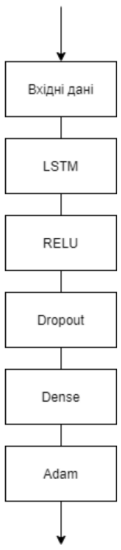


Рисунок 2.4 — Архітектура розробленої моделі

2.4.2 Побудова моделі машинного навчання

Для правильної розробки моделі машинного навчання потрібно обрати інструменти, які підходять для досягнення поставленої цілі. Важливим аспектом є використання навчальної вибірки для тренування моделі [32].

Тестова вибірка, яка використовується після тренування, дозволяє налаштувати гіперпараметри моделі і порівняти їх з передбачуваною точністю.

Навчальну вибірку можна розглядати як набір даних, за допомогою якого будуються ваги моделі. Це дозволяє точно налаштувати параметри або архітектуру моделі. Перевага полягає в змозі повторно порівнювати різні параметри з одними і тими ж даними та вагами, щоб визначити, як зміни параметрів впливають на прогнозувану здатність моделі. Тестова вибірка використовується тільки для оцінки точності прогнозування навченої моделі на нових, невидимих даних після налаштування параметрів і архітектури за допомогою наборів даних для навчання та перевірки.

Для ефективного навчання моделі необхідно мати достатню кількість даних, щоб мережа могла виявити ключові моменти, необхідні для передбачення подальшого руху ціни. Тому найкраще використовувати дані компаній, які належать до фондового ринку протягом тривалого періоду часу. Важливо, щоб дані про ціну не містили різких спадів, що можуть бути пов'язані зі зрізанням ціни акцій компанією, що дає можливість доступу до них більш широкому колу людей. Приклад графіку з роздрібленням ціни показаний нижче:

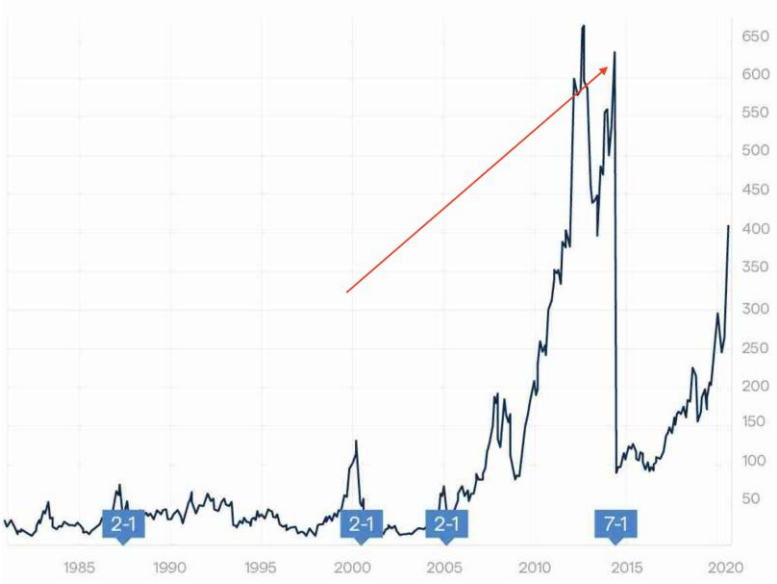


Рисунок 2.5 — Період роздріблення акцій на графіку, компанія APPLE

Саме тому, для побудови моделі необхідно врахувати особливості часових рядів, використовуючи дані і встановлюючи кількість днів для обчислень. Додатково, важливо забезпечити, що дані, використані для навчання, не використовуються в тестовій вибірці.

2.4.2.1 Інструменти реалізації моделі машинного навчання

Вибір правильних інструментів для реалізації машинного навчання визначає успіх проекту, оскільки це впливає на продуктивність, ефективність та точність моделі. Відповідно обрані інструменти забезпечують зручну розробку, розширені

можливості та надійну реалізацію алгоритмів машинного навчання. Для побудови та навчання моделі були використані такі інструменти:

— TensorFlow: Це потужна відкрита бібліотека машинного навчання, яка надає широкі можливості для побудови та тренування моделей нейронних мереж. TensorFlow забезпечує гнучкість та ефективність обчислень, що дозволяє розробникам реалізовувати складні алгоритми машинного навчання.

— Keras: Це високорівневий інтерфейс над TensorFlow, який спрощує процес побудови, навчання та оцінки моделей. Keras надає інтуїтивно зрозумілі API для швидкої розробки моделей машинного навчання, що робить його популярним вибором для початківців та досвідчених розробників.

— Python: Це мова програмування, яка використовується для реалізації моделі та обробки даних. Python має простий синтаксис, багатий екосистему бібліотек та підтримку для наукових обчислень, що робить його популярним вибором для реалізації моделей машинного навчання.

— Pandas: Це бібліотека для обробки та аналізу даних. Pandas надає потужні інструменти для завантаження, очищення, маніпулювання та аналізу даних, що робить його важливим компонентом у роботі з даними для моделі машинного навчання.

— NumPy: Це бібліотека для обчислення наукових обчислень у Python. NumPy надає підтримку для багатовимірних масивів та математичних функцій, що дозволяє ефективно працювати з даними та виконувати матричні обчислення.

2.4.2.2 Реалізація моделі машинного навчання

Реалізація моделі машинного навчання включає побудову архітектури LSTM-моделі з одним шаром LSTM, дотримання щільності до 50 нейронів і використання функції активації RELU предоставить достатній рівень прогнозування та оптимізації моделі. У моделі також використовується шар

регуляризації Dropout для уникнення перенавчання. Архітектура моделі була представлена на рисунку 2.4. Також, додано контрольну точку, яка допомагає мінімізувати втрати на наборі перевірки. Кожна модель, яка має менші втрати під час перевірки порівняно з попередніми моделями, зберігається. Далі потрібно налаштувати навчання нашої моделі, вказавши аргументи про кількість проходів та максимальну можливість зберігання навчених даних. Код для навчання моделі мовою Python буде виглядати так:

```
checkpoint = ModelCheckpoint(filepath = filepath,
    monitor = 'val_loss',
    verbose = 1,
    save_best_only = False,
    mode ='min')
history = model.fit(X_train, Y_train, epochs = 100, batch_size = 20,
            validation_data = (X_test, Y_test),
            callbacks = [checkpoint],
            verbose = 1,
            shuffle = False)

model.summary()
```

Рисунок 2.6 — Код навчання моделі LSTM

У цьому розділі було проведено проектування архітектури веб-сервісу для передбачення цін акцій фондового ринку. Було вибрано компоненти для клієнтського інтерфейсу, серверного шару та бази даних, що відповідають вимогам швидкості, продуктивності та масштабованості. Для реалізації клієнтського інтерфейсу було обрано фреймворк Vue.js та бібліотеку Tailwind CSS. Ці технології забезпечують швидкий доступ до інформації, гнучкість в розробці та привабливий вигляд інтерфейсу. Серверний шар буде реалізовано за допомогою Node.js та фреймворку Fastify. Для зберігання та управління даними

використовується база даних Firestore, яка забезпечує гнучкість та масштабованість.

Також було використано технологію Flask для реалізації мікросервісу передбачення ціни акцій та Docker для його ізоляції. Ці технології дозволяють ефективно прогнозувати ціни акцій та забезпечити безпеку та незалежність мікросервісу.

Проектування моделі машинного навчання для прогнозування цін акцій було підтримане детальним аналізом різних типів моделей. У розділі про модель було викладено розгорнутий опис вибраної моделі типу Sequential з використанням LSTM алгоритму. Ця модель дозволяє побудувати ефективний сервіс прогнозування вартості акцій для інвесторів. Особлива увага приділялася збереженню даних навчання, що дозволяє використовувати їх без необхідності повторного навчання моделі. Комбінація моделі типу Sequential та використання LSTM алгоритму дозволяє отримати точні прогнози цін акцій з високою швидкодією та надійністю. У процесі реалізації моделі, були використані такі технології як TensorFlow та Keras. TensorFlow є потужною відкритою бібліотекою для машинного навчання та глибинного навчання, яка забезпечує широкий спектр функцій для побудови та тренування моделей. Keras, з іншого боку, є високорівневим API для TensorFlow, що спрощує розробку нейронних мереж.

Додатково, у розробці моделі були використані такі інструменти як Pandas та NumPy. Pandas та NumPy - це бібліотеки для аналізу та обробки даних, що надають зручні структури даних та функції для ефективної маніпуляції даними.

Всі обрані технології та архітектурні рішення допоможуть реалізувати веб-сервіс з прогнозування цін акцій фондового ринку, що буде забезпечувати швидкий доступ до інформації та точність передбачення.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ ДЛЯ ПРОГНОЗУВАННЯ ЦІН НА ФОНДОВОМУ РИНКУ

3.1 Інструкція користувача

Розробка інтерфейсу користувача є невід'ємною частиною процесу створення веб-сервісу передбачення цін акцій фондового ринку. Інтерфейс виступає важливим елементом взаємодії з користувачами. Правильно розроблений інтерфейс дозволить користувачам легко здійснювати авторизацію та реєстрацію, отримувати доступ до передбачень ціни акцій та відстежувати свої інвестиційні портфелі. В цьому розділі ми детально розглянемо процес розробки інтерфейсу та презентуємо ключові аспекти, що стосуються кожної з вказаних сторінок.

3.1.1. Авторизація

На сторінці авторизації користувач може увійти до свого облікового запису, використовуючи зареєстрований логін та пароль. При правильному введенні даних, користувачу надається доступ до інших функціональних сторінок сервісу.

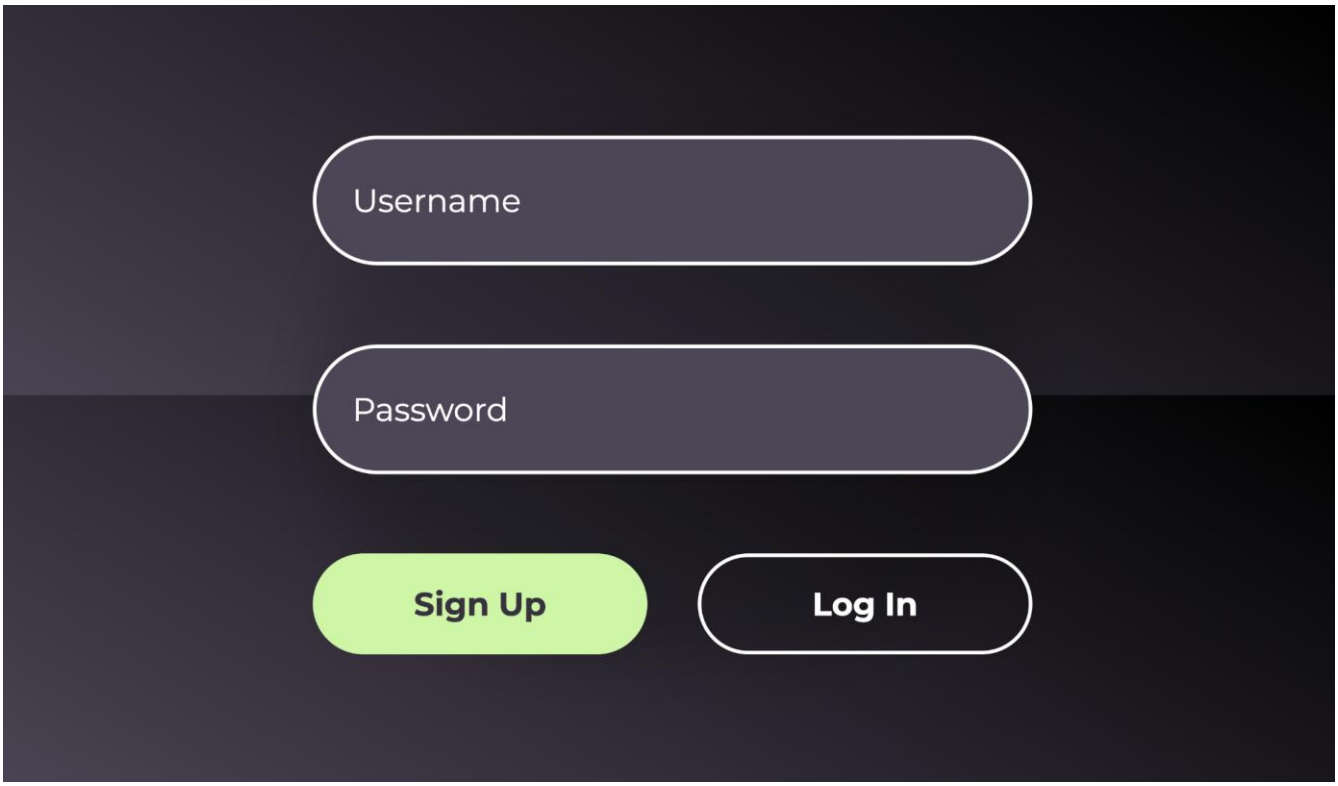


Рисунок 3.1 — Екран авторизації

3.1.2. Реєстрація

Сторінка реєстрації дозволяє новим користувачам створити обліковий запис в системі. Користувач повинен ввести необхідну інформацію, таку як ім'я, пароль та підтвердження паролю. Після успішної реєстрації, користувачу надається можливість авторизуватися та використовувати функціональність сервісу.

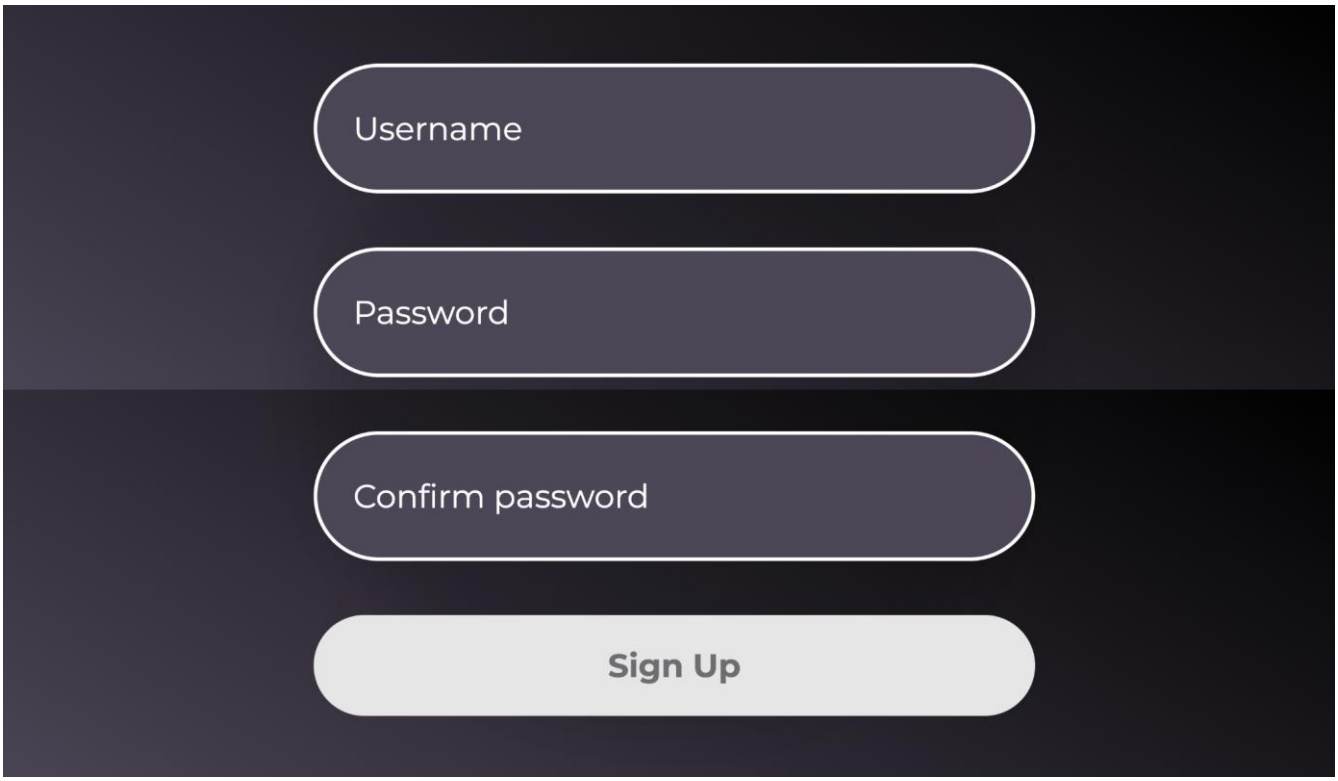
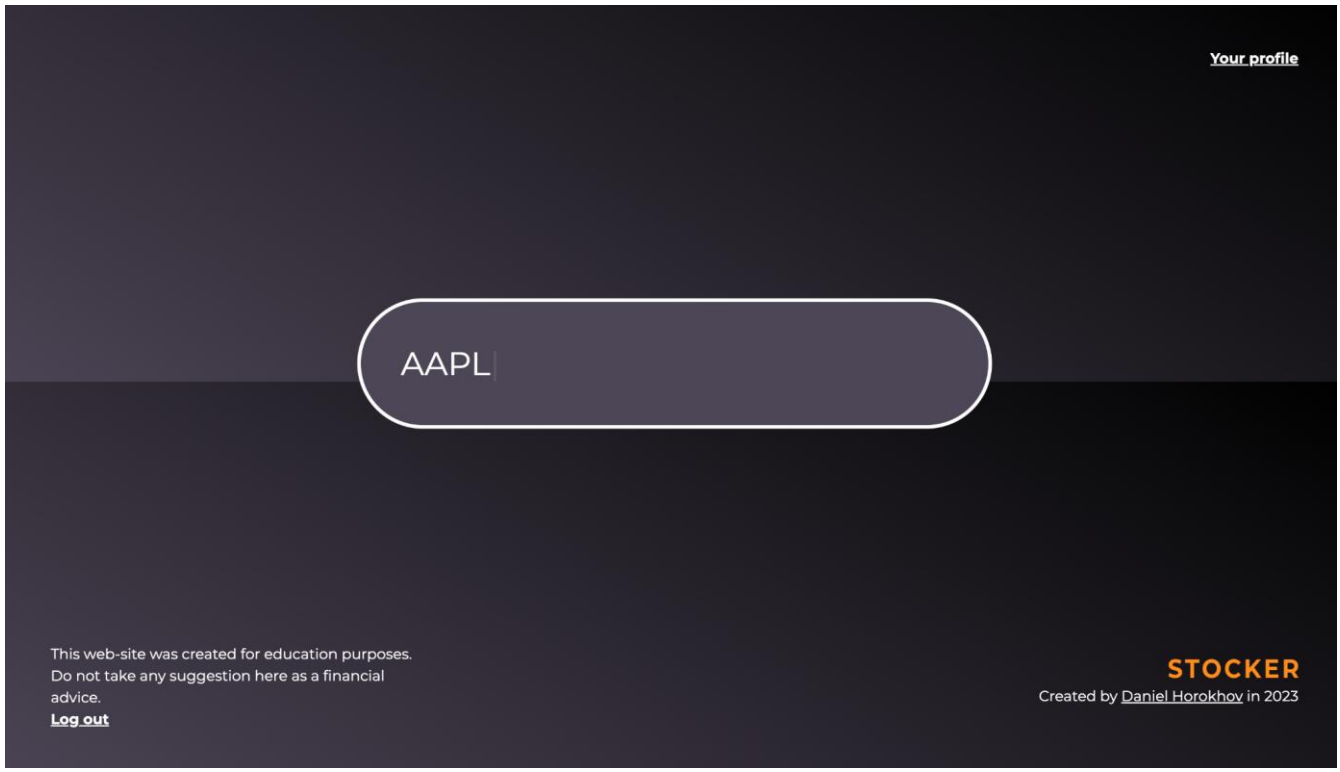


Рисунок 3.2 — Екран реєстрації

3.1.3. Сторінка Передбачення ціни акції

Ця сторінка дозволяє користувачам отримати прогнозовану ціну акції на основі навчаних данних нейронної мережі. Користувач може ввести необхідну інформацію, таку як символ акції. Після обробки даних, система відображає прогнозовану та попередні ціни акції.

Рисунок 3.3 — Головний екран з полем для вводу назви акцій і отримання



передбачення ціни

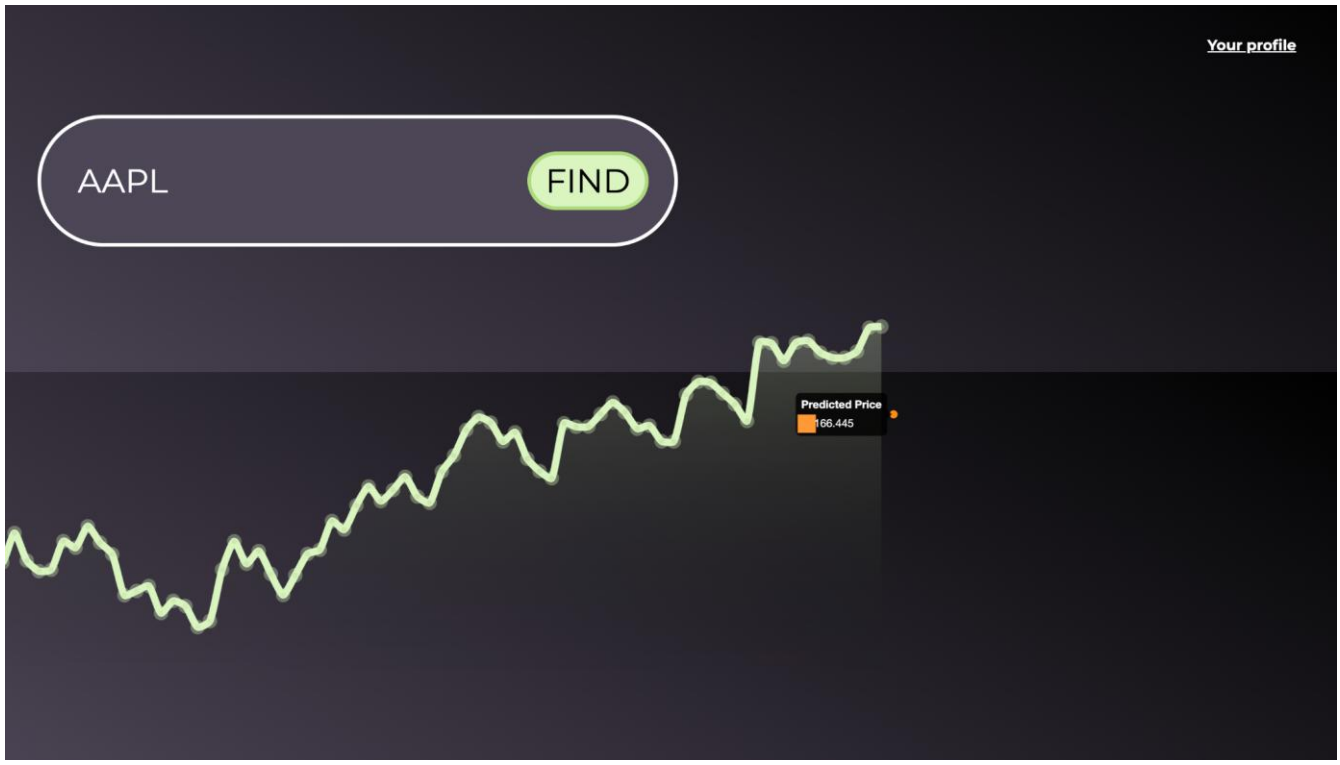


Рисунок 3.4 — Головний екран з графіком цін та передбаченною ціною

3.1.4 Профіль користувача

Ця сторінка містить інвестиційний портфель користувача. Користувач може додавати свої інвестиції, вказувати кількість акцій, які були куплені та відслідковувати результат диверсифікації на діаграмі.

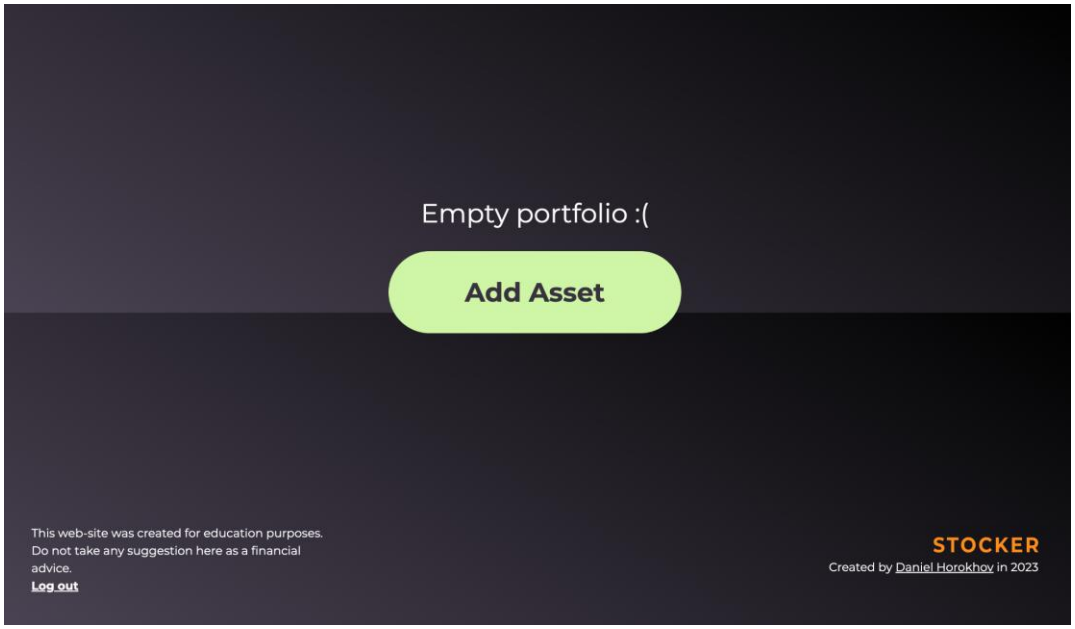
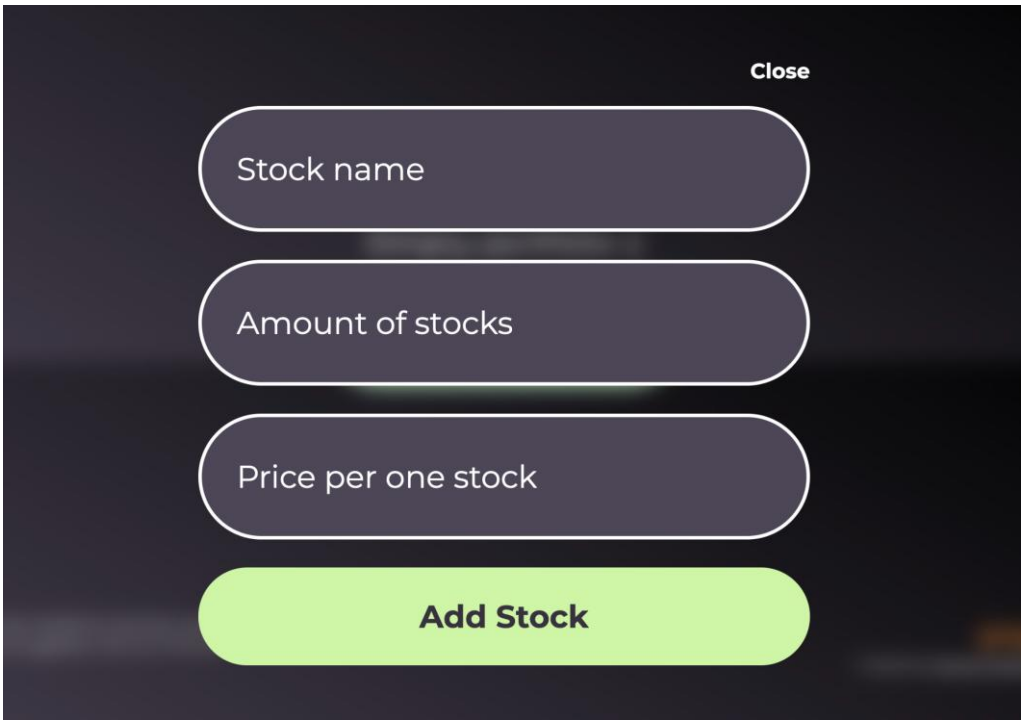


Рисунок 3.5 — Екран профілю користувача без наявних інвестиції

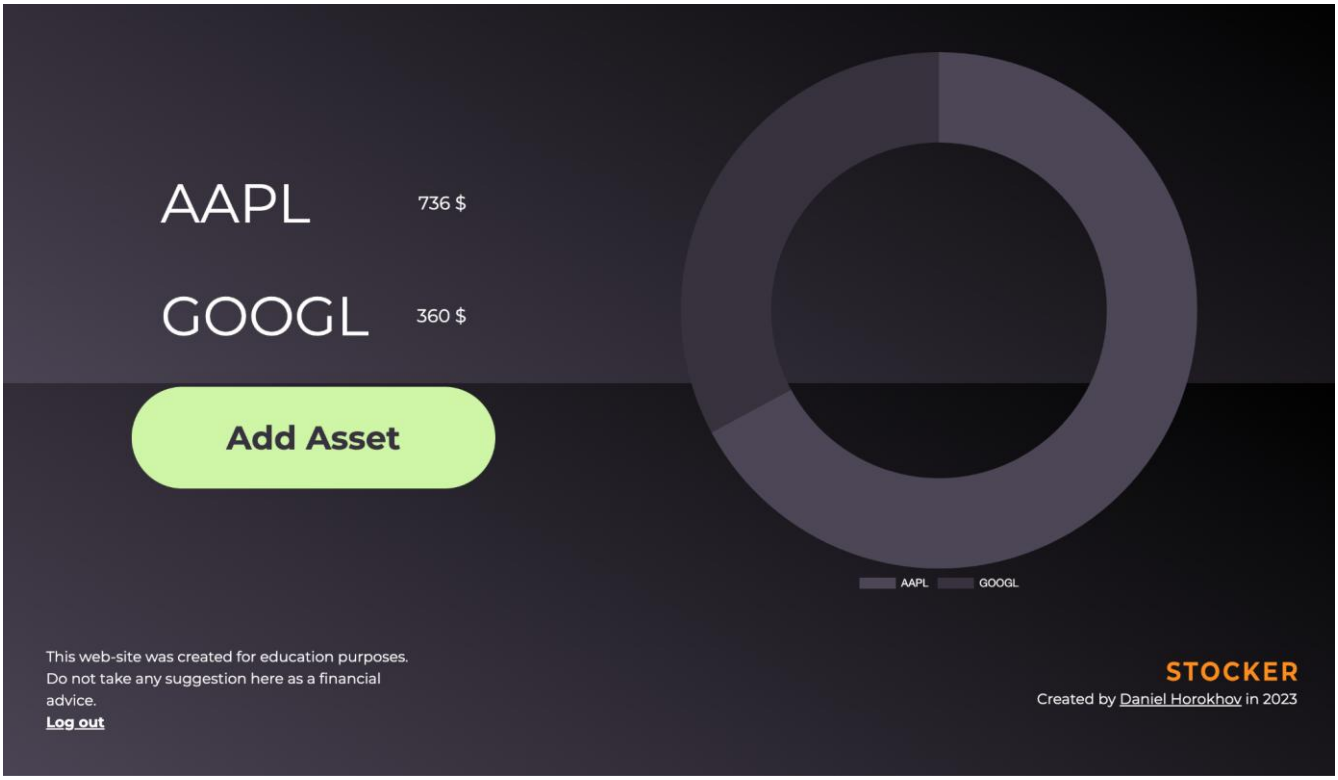
Так як наразі наш користувач не має ніяких інвестицій у своєму портфоліо,



їх потрібно додати.

Рисунок 3.6 — Форма додавання інвестицій до портфоліо користувача

Додавши відповідну кількість акцій у інвестиційний портфель, користувач буде мати можливість побачити увесь список інвестицій які було зрублені, суму кожної проінвестованої акції та діаграму, яка показує як сильно диверсифікований



портфель користувача.

Рисунок 3.7 — Портфоліо користувача зі списком проінвестованих акцій та діаграмою диверсифікації портфелю

Розробка інтерфейсу користувача виконувалась з використанням сучасних технологій, таких як HTML, CSS та JavaScript. Для забезпечення привабливого та зручного інтерфейсу була використана бібліотека Vue.js, яка дозволяє розробляти компонентну структуру та динамічні елементи веб-сторінок. Також була використана бібліотека Tailwind CSS для стилізації та оформлення інтерфейсу.

Результатом розробки інтерфейсу є зручний та інтуїтивно зрозумілий веб-

сервіс, який дозволяє користувачам авторизуватися, реєструватися, отримувати прогнози цін акцій та керувати своїм портфелем інвестицій.

3.2 Планування та реалізація подальших удосконалень та оновлень веб-сервісу

Під час програмної реалізації веб-сервісу з прогнозування цін акцій, з'ясувалося, що є кілька шляхів покращення точності моделі. Один з них - експериментування з параметрами та використання якісних наборів даних для кожної нової компанії, з метою покращення якості прогнозування. Поточна модель передбачення цін акцій використовує ряд особливостей, таких як дата відкриття та вартість. Однак, для дальшого покращення моделі можна розглянути додавання інших особливостей, таких як кількість акцій, ціна/прибуток та дивіденди. Правильне балансування використовуваних особливостей є важливим, оскільки їх надмірне використання може негативно вплинути на точність прогнозування. Зокрема, для веб-сервісу прогнозування цін акцій, варто зосередитися на пошуку оптимального набору особливостей, які забезпечать найкращу точність прогнозу.

Також можна зосередитися на покращенні процесу авторизації. Одним з можливих підходів є використання технології fastify-jwt, яка надає зручні та безпечні механізми для аутентифікації користувачів за допомогою токенів. Це дозволить забезпечити високий рівень безпеки під час доступу до ресурсів веб-сервісу.

Крім того, можна розширити функціональність профілю користувача, додавши нові можливості. Наприклад, відслідковування прибутку інвестицій може бути корисною функцією, яка дозволить користувачам переглядати та аналізувати результати своїх інвестиційних операцій. Також, додавання можливості

відстеження дивідентності дозволить користувачам переглядати та аналізувати отримані дивіденди зі своїх інвестицій.

Крім вищезазначених покращень, однією зі значущих можливостей для розширення функціоналу та покращення користувацького досвіду є мобільна адаптація додатку. У сучасному світі мобільні пристрої стають все більш популярними та широко використовуваними засобами доступу до інтернет-сервісів. Тому важливо забезпечити зручну та оптимізовану версію веб-додатку для мобільних пристроїв.

Мобільна адаптація додатку може включати такі аспекти:

— Респонсивний (responsive) дизайн: Пристрої з різними розмірами екранів потребують адаптивного дизайну, який забезпечує оптимальне відображення контенту та зручну навігацію на будь-якому пристрої.

— Оптимізовані функціональність та швидкодія: Мобільні пристрої мають свої особливості та обмеження щодо обробки даних та швидкодії. Для оптимальної роботи додатку на мобільних пристроях важливо оптимізувати його функціонал та швидкодію.

Мобільна адаптація додатку підвищить доступність та зручність для користувачів, дозволяючи їм з легкістю використовувати додаток на своїх мобільних пристроях, отримувати вчасно інформацію та керувати своїми інвестиціями з будь-якого місця та в будь-який час.

У цьому розділі була розглянута програмна реалізація веб-сервісу для прогнозування цін на фондовому ринку.

Додатково, розглянуто можливості покращення авторизації за допомогою fastify-jwt, що дозволить забезпечити безпеку та захист персональних даних користувачів. В профілі користувача розглянуто можливість відслідковування прибутку інвестицій та дивідентність, що надасть користувачам зручні інструменти для керування своїми активами. З метою поліпшення веб-сервісу

розглянуто планування та реалізацію подальших удосконалень. Це включає розширення функціоналу, оптимізацію алгоритмів передбачення, розробку мобільної адаптації додатку для забезпечення зручного користування на мобільних пристроях. В цілому, програмна реалізація веб-сервісу для прогнозування цін на фондовому ринку була успішно здійснена з використанням сучасних технологій та методів машинного навчання. Результатом є потужний інструмент, що надає користувачам можливість отримувати прогнози цін акцій та ефективно керувати своїми інвестиціями.

ВИСНОВОК

Таким чином, у результаті виконання кваліфікаційної роботи бакалавра розроблено веб-сервіс для прогнозування короткострокової ціни акцій фондового ринку, що обумовлює прогнозування процесів, зокрема:

— було проведено аналіз основних процесів фондового ринку та способи їх автоматизації;

— було здійснено аналіз способів автоматизації процесів фондового ринку;

— були проведені порівняння доступних моделей для прогнозування короткострокової ціни акції;

— було спроектовано та реалізовано веб-сервіс з прогнозування ситуації на фондовому ринку з урахуванням інженерії вимог.

Під час розробки було проведено аналіз ринку та обрано найбільш ефективні та підходящі технології, а також розроблено алгоритм передбачення цін на акції. Проектування та розробка веб-сервісу включала розробку серверного та клієнтського шару, інтеграцію бази даних та розробку інтерфейсу користувача.

Було використано Fastify фреймворк для реалізації серверної частини, який забезпечив зручне API для взаємодії з клієнтами. Для реалізації мікросервісу для передбачень ціни акцій було використано Flask, який надав потрібну функціональність та забезпечив ефективну роботу веб-сервісу. Додатково, для забезпечення безпеки, масштабованості та ефективності роботи веб-сервісу використовувалися інші технології, такі як Firestore та Docker. Одним із основних аспектів роботи було розроблення та застосування моделі машинного навчання для передбачення цін акцій. Було обрано LSTM алгоритм, який виконав поставлену ціль на достатньому рівні. Додатково, було розглянуто можливості покращення моделі шляхом додавання нових особливостей та параметрів. У результаті роботи було розроблено веб-сервіс, який надає користувачам зручність,

ефективність та надійність при прогнозуванні цін акцій. Веб-сервіс може бути використаний як інструмент для інвесторів, трейдерів та всіх зацікавлених осіб у вирішенні фінансових задач та прийнятті рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сутормiна В.М. Фінанси зарубіжних корпорацій: Підручник. Київ: КНЕУ, 2004. 566с.
2. Malkiel, Burton G. The Efficient Market Hypothesis and Its Critics. Journal of Economic Perspectives, 17(1), с. 59-82, 2003..
3. Hull, J.C. Options, Futures and Other Derivatives. 9th ed., Pearson Education, 2012.
4. Kirilenko та інші. "The Flash Crash: High-Frequency Trading in an Electronic Market." Journal of Finance, с. 967-998, 2017.
5. Теорія ризику інвестицій Гарі Марковіца. Wikipedia: веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D1%80%D0%B8%D0%B7%D0%B8%D0%BA%D1%83_%D1%96%D0%BD%D0%B2%D0%B5%D1%81%D1%82%D0%B8%D1%86%D1%96%D0%B9_%D0%93%D0%B0%D1%80%D1%96_%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D1%96%D1%86%D0%B0 (дата звернення: 10.03.2023)
6. В. Автономова, О.Ананьіна, Н. Макашевой. Історія економічних вчень: підручник. ИНФРА-М, 2001, 784 с.
7. Roy E. Bailey. The Economics of Financial Markets: Cambridge University Press, New York, 2005, 14 с.
8. Dybvig, P.H., Ross, S.A. Finance: London, Palgrave Macmillan. 1989, 290 с.
9. Geoffrey R.D. Underhill. European Journal of Political Research, 1991, с. 197-225.
10. Algorithmic Trading Market Size, Share & Trends Analysis Report By Solution, By Service, By Deployment, By Trading Types, By Types Of Traders, By Region, And Segment Forecasts: *Grandview research:* веб-сайт URL:

https://www.grandviewresearch.com/industry-analysis/algorithmic-trading-market-report (дата звернення 17.03.2023)

11. Murphy, J.J. Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications: New York, New York Institute of Finance, 1999, 576 с.
12. Jeffery S. Abarbanell and Brian J. Bushee. Abnormal Returns to a Fundamental Analysis Strategy: American Accounting Association, с. 19-45, 1998.
13. Lou Blouin. AI's mysterious ‘black box’ problem explained: *Dearborn:* веб-сайт URL: https://umdearborn.edu/news/ais-mysterious-black-box-problem-explained (дата звернення 17.03.2023)
14. IBM. Overfitting: *IBM*: веб-сайт URL: https://www.ibm.com/topics/overfitting (дата звернення: 20.03.2023)
15. ASQ. Learn about quality: ASQ: веб-сайт URL: https://asq.org/quality-resources/survey (дата звернення: 20.03.2023)
16. Titus Utibe Monday. Impacts of Interview as Research Instrument of Data Collection in Social Sciences: Titus Utibe Monday: стаття URL: https://ics.events/wp-content/uploads/2020/10/Article2-JDAH-1.1-DOI.pdf (дата звернення 21.03.2023)
17. Adobe. User flow diagram — what it is, why it’s important, and how to create one: Adobe: веб-сайт URL: https://business.adobe.com/blog/basics/how-to-make-a-user-flow-diagram (дата звернення 22.03.2023)
18. React: React: веб-сайт URL: https://react.dev/ (дата звернення: 23.03.2023)
19. Vue.js: Vue.js: веб-сайт URL: https://vuejs.org/guide/introduction.html (дата звернення 23.03.2023)
20. Angular: Angular: веб-сайт URL: https://angular.io/guide/what-is-angular (дата звернення: 23.03.2023)
21. Svelte: Svelte: веб-сайт URL: https://svelte.dev/ (дата звернення: 23.03.2023)

22. Fastify: Fastify: веб-сайт URL: https://www.fastify.io/ (дата звернення: 23.03.2023)
23. Django: Django: веб-сайт URL: https://docs.djangoproject.com/en/4.2/ref/contrib/admin/ (дата звернення: 24.03.2023)
24. Spring Boot: Spring: веб-сайт URL: https://spring.io/projects/spring-boot (дата звернення: 24.03.2023)
25. Ruby on Rails: Ruby on Rails: веб-сайт URL: https://rubyonrails.org/ (дата звернення: 24.03.2023)
26. GlobalLogic. Мікросервісна архітектура для початківців. Частина I: Globallogic: стаття URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/ (дата звернення 27.03.2023)
27. Chandler Harris. Microservices vs. monolithic architecture: Atlassian: веб-сайт URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith (дата звернення 28.03.2023)
28. Commonly used Machine Learning Algorithms: Analytics Vidhya: веб-сайт URL: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ (дата звернення: 30.03.2023)
29. Tree-Based Machine Learning Algorithms Explained: Medium: веб-сайт URL: https://medium.com/analytics-vidhya/tree-based-machine-learning-algorithms-explained-b50937d3cf8e (дата звернення: 03.04.2023)
30. Sequence Models & Recurrent Neural Networks (RNNs): Towards data science: веб-сайт URL: https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1 (дата звернення: 03.04.2023)
31. LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past: Towards data science: веб-сайт URL: https://towardsdatascience.com/lstm-

recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e (дата звернення: 05.04.2023)

32. Understanding the Importance of Training Data in Machine Learning: Medium: веб-сайт URL: https://medium.com/the-ai-technology/understanding-the-importance-of-training-data-in-machine-learning-da4235332904 (дата звернення: 06.04.2023)

ДОДАТКИ

```
const User = require('../models/user');

exports.register = async (request, reply) => {
    const { body } = request;
    const {
        username,
        password
    } = body;

    const isUserExist = await User.findOne(username);

    if (isUserExist) return reply.code(409).send({ msg: 'User exists!' });

    const user = new User(username);
    const {
        salt,
        encryptedPassword
    } = await User.encryptUserPassword(password, 10);

    await user.addToDatabase(encryptedPassword, salt);
    reply.code(200).send({ username, msg: 'User was created!' });
}

exports.auth = async (request, reply) => {
    const { headers } = request;
    const { authorization } = headers;

    const { username, password } = User.retrieveUserData(authorization.replace('Bearer ', ''));
    const user = new User(username);

    const userFromDatabase = await user.getUserFromDatabase();

    if (!userFromDatabase || !Object.keys(userFromDatabase).length)
        return reply.code(401).send({ msg: 'Wrong credentials!' });

    const isPasswordsSame = await user.comparePasswords(userFromDatabase.password, password);

    if (!isPasswordsSame)
        return reply.code(401).send({ msg: 'Wrong password!' });

    reply.code(200).send({ username, msg: 'Auth success!' });
}
```

ДОДАТОК А. Мікросервіс авторизації та розмітка

```
<template>
  <section class="h-full">
    <div class="container h-full flex items-center justify-center">
      <form @submit.prevent class="relative grow md:max-w-[800px] flex flex-col md:gap-y-14 block-shadow-layer z-10">
        <BaseInput
          name="login"
          type="text"
          placeholder="Username"

          v-model="login"
        />

        <BaseInput
          name="password"
          type="password"
          placeholder="Password"

          v-model="password"
        />

        <div class="flex justify-between md:gap-x-14 w-full">
          <BaseButton
            color="green"
            hoverClasses="hover:bg-brand-700 hover:border-brand-700 hover:text-blue-700"

            :link="{ name: 'signUp' }"
          >
            Sign Up
          </BaseButton>

          <BaseButton
            color="white"
            hoverClasses="hover:text-blue-700 fade-in-bg-to-right before:bg-white relative overflow-hidden z-10"
            :outlined="true"

            :loading="isLoading"

            @click="submitForm"
          >
            Log In
          </BaseButton>
        </div>
      </form>
    </div>
  </section>
</template>
```

```
export const useAuthStore = defineStore('auth', () => {
  const isLoggedIn = ref(false);
  const token = ref('');
  const username = ref('');

  const isAuthenticated = computed(() => isLoggedIn);

  function setUsername(inputUsername) {
    username.value = inputUsername;
  }

  function authenticateApp() {
    isLoggedIn.value = true;
  }

  async function registerUser(username, password) {
      return await _axios.request({
        method: 'POST',
        url: '/sign-up',
        data: {
          username,
          password,
        },
      })
  }

  async function logInUser(username, password) {
    const encodedData =
      [encodeToBase64(username), encodeToBase64(password)];
    return await _axios.request({
      method: 'GET',
      url: '/auth',
      headers: {
        'Authorization': 'Bearer ' + encodedData.join('.'),
      }
    })
  }

  function logOut() {
    isLoggedIn.value = false;
    username.value = '';
    token.value = '';
  }

  return {
    isAuthenticated,
    token,
    username,
```

```
const User = require('../models/user');

exports.register = async (request, reply) => {
    const { body } = request;
    const {
        username,
        password
    } = body;

    const isUserExist = await User.findOne(username);

    if (isUserExist) return reply.code(409).send({ msg: 'User exists!' });

    const user = new User(username);
    const {
        salt,
        encryptedPassword
    } = await User.encryptUserPassword(password, 10);

    await user.addToDatabase(encryptedPassword, salt);
    reply.code(200).send({ username, msg: 'User was created!' });
}

exports.auth = async (request, reply) => {
    const { headers } = request;
    const { authorization } = headers;

    const { username, password } = User.retrieveUserData(authorization.replace('Bearer ', ''));
    const user = new User(username);

    const userFromDatabase = await user.getUserFromDatabase();

    if (!userFromDatabase || !Object.keys(userFromDatabase).length)
        return reply.code(401).send({ msg: 'Wrong credentials!' });

    const isPasswordsSame = await user.comparePasswords(userFromDatabase.password, password);

    if (!isPasswordsSame)
        return reply.code(401).send({ msg: 'Wrong password!' });

    reply.code(200).send({ username, msg: 'Auth success!' });
}
```

```
import numpy as np
import tensorflow as tf
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import urllib.request, json
import datetime
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from sklearn.metrics import mean_squared_error

API_KEY='7NYLN5LMJVH1TEUR'
SOURCE = 'alphavantage'

def tickerfromuser(userTicker):
    if SOURCE == 'alphavantage':
        ticker = userTicker
        STOCKS_API_ENDPOINT="https://www.alphavantage.co/query?function=TIME_SERIES_DAILY_ADJUSTED&symbol=%s&outputsize=full&apikey=%s"
%(ticker,API_KEY)
        file_to_save = 'stock_market_data-%s.csv'%ticker

    with urllib.request.urlopen(STOCKS_API_ENDPOINT) as url:
        data = json.loads(url.read().decode())
        data = data['Time Series (Daily)']
        df = pd.DataFrame(columns=['Date','Low','High','Close','Open'])

        for k,v in data.items():
            date = datetime.datetime.strptime(k, '%Y-%m-%d')
            data_row = [date.date(),float(v['3. low']),float(v['2. high']),
                    float(v['4. close']),float(v['1. open'])]
            df.loc[-1,:] = data_row
            df.index = df.index + 1
        print('Data saved to : %s'%file_to_save)
        df.to_csv(file_to_save)

    ticker = pd.read_csv('stock_market_data-%s.csv'%ticker)
    ticker = ticker[['Date', 'Close']]
    ticker.Date = pd.to_datetime(ticker.Date, format ='%Y-%m-%d')
    ticker_length = len(ticker)

    new_ticker = ticker.loc[0:ticker_length//2]
    new_ticker = new_ticker.drop('Date', axis = 1)
    new_ticker = new_ticker.reset_index(drop = True)
```

```
T = new_ticker.values
T = T.astype('float32')
T = np.reshape(T, (-1, 1))


scaler = MinMaxScaler(feature_range = (0, 1))
T = scaler.fit_transform(T)
train_size = int(len(T) * 0.80)
test_size = int(len(T) - train_size)
train, test = T[0:train_size,:], T[train_size:len(T),:]

def create_features(data, window_size):
    X, Y = [], []
    for i in range(len(data) - window_size - 1):
        window = data[i:(i + window_size), 0]
        X.append(window)
        Y.append(data[i + window_size, 0])
    return np.array(X), np.array(Y)
```

ДОДАТОК Б. Мікросервіс передбачення ціни

```
window_size = 20
X_train, Y_train = create_features(train, window_size)
X_test, Y_test = create_features(test, window_size)
X_train = np.reshape(X_train, (X_train.shape[0], 1,
X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1,
X_test.shape[1]))
T_shape = T.shape
train_shape = train.shape
test_shape = test.shape

def isLeak(T_shape, train_shape, test_shape):
    return not(T_shape[0] == (train_shape[0] + test_shape[0]))

print(isLeak(T_shape, train_shape, test_shape))
tf.random.set_seed(11)
np.random.seed(11)
model = Sequential()
model.add(LSTM(units = 50, activation = 'relu',
               input_shape = (X_train.shape[1], window_size)))
model.add(Dropout(0.2))
model.add(Dense(1, activation = 'linear'))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
filepath = 'saved_models/model_epoch_{epoch:02d}.hdf5'
checkpoint = ModelCheckpoint(filepath = filepath,
    monitor = 'val_loss',
    verbose = 1,
    save_best_only = False,
    mode ='min')
history = model.fit(X_train, Y_train, epochs = 100, batch_size = 20,
            validation_data = (X_test, Y_test),
            callbacks = [checkpoint],
            verbose = 1,
            shuffle = False)

model.summary()
best_model = load_model('saved_models/model_epoch_93.hdf5')
train_predict = best_model.predict(X_train)
Y_hat_train = scaler.inverse_transform(train_predict)
test_predict = best_model.predict(X_test)
Y_hat_test = scaler.inverse_transform(test_predict)
Y_test = scaler.inverse_transform([Y_test])
Y_train = scaler.inverse_transform([Y_train])
Y_hat_train = np.reshape(Y_hat_train, newshape = len(Y_hat_train))
Y_hat_test = np.reshape(Y_hat_test, newshape = len(Y_hat_test))
Y_train = np.reshape(Y_train, newshape = len(Y_hat_train))
Y_test = np.reshape(Y_test, newshape = len(Y_hat_test))
train_RMSE = np.sqrt(mean_squared_error(Y_train, Y_hat_train))
test_RMSE = np.sqrt(mean_squared_error(Y_test, Y_hat_test))
```

```
model.summary()
best_model = load_model('saved_models/model_epoch_93.hdf5')
train_predict = best_model.predict(X_train)
Y_hat_train = scaler.inverse_transform(train_predict)
test_predict = best_model.predict(X_test)
Y_hat_test = scaler.inverse_transform(test_predict)
Y_test = scaler.inverse_transform([Y_test])
Y_train = scaler.inverse_transform([Y_train])
Y_hat_train = np.reshape(Y_hat_train, newshape = len(Y_hat_train))
Y_hat_test = np.reshape(Y_hat_test, newshape = len(Y_hat_test))
Y_train = np.reshape(Y_train, newshape = len(Y_hat_train))
Y_test = np.reshape(Y_test, newshape = len(Y_hat_test))
train_RMSE = np.sqrt(mean_squared_error(Y_train, Y_hat_train))
test_RMSE = np.sqrt(mean_squared_error(Y_test, Y_hat_test))
print('Train RMSE is: ')
print(train_RMSE, '\n')
print('Test RMSE is: ')
print(test_RMSE)

Y = np.append(Y_train, Y_test)
Y_hat = np.append(Y_hat_train, Y_hat_test)
result_df = pd.DataFrame()
result_df['Actual_Y'] = Y
result_df['Predicted_Y'] = Y_hat
return round(result_df['Predicted_Y'][0], 3)
```

```
const Stock = require('../models/stock');

const stockModel = new Stock();

exports.predictStock = async (request, reply) => {
    const { body } = request;
    const { stock } = body;

    try {
        const predictionResponse = await stockModel.predictStock(stock);

        reply.code(200).send({ data: predictionResponse });
    } catch(e) {
        reply.code(400).send({ msg: 'Prediction error!' });
    }
}

exports.retrieveStockHistory = async (request, reply) => {
    const { body } = request;
    const { stock } = body;

    try {
        const stockHistory = await stockModel.getStockPriceHistory(stock);

        reply.code(200).send({ data: stockHistory })
    } catch(e) {
        reply.code(400).send({ msg: 'Stock history error.' })
    }
}
```

```
const Stock = require('../models/stock');

const stockModel = new Stock();

exports.addStockToPortfolio = async (request, reply) => {
    const { username, stock, amount, averagePrice } = request.body;

    try {
        const response = await stockModel.addToPortfolio(username, stock, amount, averagePrice);

        reply.code(200).send({ msg: 'Stock was successfully added', data: response });
    } catch(e) {
        reply.code(400).send({ msg: 'Error with adding investment to portfolio.' });
    }
}

exports.removeStockFromPortfolio = async (request, reply) => {
    const { username, stock } = request.body;

    try {
        const response = await stockModel.deleteStockFromPortfolio(username, stock);

        reply.code(200).send({ data: response });
    } catch (e) {
        reply.code(400).send({ msg: 'Error with removing stock from portfolio' });
    }
}

exports.getPortfolio = async (request, reply) => {
    const { username } = request.body;

    try {
        const response = await stockModel.getPortfolio(username);

        reply.code(200).send({ data: response });
    } catch (e) {
        reply.code(400).send({ msg: e.message });
    }
}
```

ДОДАТОК В. Мікросервіс портфоліо