МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ЧАСОВИХ РЯДІВ»**

Виконав: студент 2 курсу, групи 6.1213-пі-с3
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія(зі скороченим терміном навчання)
(назва освітньої програми)
О.А. Наврось
(ініціали та прізвище)
Керівник професор кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія (зі скороченим терміном навчання)

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Навросю Олександру Андрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи аналізу часових рядів

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Типи та джерела вихідних даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи аналізу часових рядів у фінансових даних.
2. Архітектура та моделювання інформаційної системи.
3. Програмна реалізація та тестування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	13.01.2025	
2.	Збір вихідних даних.	03.02.2025	
3.	Обробка методичних та теоретичних джерел.	14.03.2025	
4.	Розробка першого та другого розділу.	28.04.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	20.06.2025	

Студент _____ О.А. Наврось
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи аналізу часових рядів»: 42 с., 30 рис., 1 табл., 24 джерела.

АЛГОРИТМИ, АНАЛІЗ ЧАСОВИХ РЯДІВ, ЕКСТРАПОЛЯЦІЯ, ІНФОРМАЦІЙНІ СИСТЕМИ, МЕТОДИ ОБРОБКИ ДАНИХ, ОБРОБКА ДАНИХ, ПРОГНОЗУВАННЯ, РЕГРЕСІЙНИЙ АНАЛІЗ, СТАТИСТИЧНІ МЕТОДИ, ЧАСОВІ СЕРІЇ.

Об’єкт дослідження: інформаційна система для аналізу і прогнозування часових рядів.

Мета роботи: розробка інформаційної системи для обробки та аналізу часових рядів з використанням статистичних і машинних методів для прогнозування на основі вхідних даних.

Метод дослідження: аналіз теоретичних джерел, методи статистичного прогнозування, математичні моделі для обробки часових рядів, розробка програмних засобів для реалізації системи.

У ході роботи було проведено аналіз предметної області, досліджено особливості часових рядів та підходи до їх обробки. Розглянуто сучасні статистичні та математичні методи прогнозування, зокрема регресійний аналіз, ковзне середнє, метод екстраполяції. На основі отриманих знань розроблено інформаційну систему, що дозволяє завантажувати вхідні дані, обробляти їх, будувати графіки, здійснювати аналіз і формувати прогнози.

SUMMARY

Bachelor's qualifying paper "Development of an Information System for the Time Series Analysis": 42 pages, 30 figures, 1 table, 24 references.

ALGORITHMS, DATA PROCESSING, DATA PROCESSING METHODS, EXTRAPOLATION, FORECASTING, INFORMATION SYSTEMS, REGRESSION ANALYSIS, STATISTICAL METHODS, TIME SERIES, TIME SERIES ANALYSIS.

The object of the study is the information system for time series analysis and forecasting.

The aim of the study is to develop an information system for processing and analyzing time series using statistical and machine learning methods for forecasting based on input data.

The method of research is the analysis of theoretical sources, statistical forecasting methods, mathematical models for processing time series, and the development of software tools for implementing the system.

In the course of the work, an analysis of the subject area was carried out, the characteristics of time series and approaches to their processing were studied. Modern statistical and mathematical forecasting methods were considered, including regression analysis, moving averages, and extrapolation methods. Based on the acquired knowledge, an information system was developed that allows users to upload input data, process it, generate graphs, perform analysis, and create forecasts.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 7
1 Теоретичні основи аналізу часових рядів у фінансових даних 9
1.1 Особливості фінансових часових рядів та технічного аналізу 9
1.2 Огляд доступних API для фінансових даних 10
1.3 Інструменти та технології реалізації системи прогнозування 13
1.3.1 Платформи для роботи з даними 13
1.3.2 Фреймворки машинного навчання 15
1.3.3 Інструменти візуалізації 18
1.4 Аналіз вимог розробки 21
2 Архітектура та моделювання інформаційної системи 23
2.1 Архітектура системи 23
2.2 Діаграми проєктування 26
2.3 Логіка роботи системи та сценарій взаємодії з користувачем 28
3 Програмна реалізація та тестування системи 30
3.1 Архітектурні рішення та ключові алгоритми 30
3.2 Стратегія тестування інформаційної системи 35
3.3 Реалізація функціональних компонентів 37
Висновки 39
Перелік посилань 41

ВСТУП

У сучасному світі інформаційні технології відіграють ключову роль у збиранні, обробці та аналізі даних у різноманітних сферах діяльності – від фінансових ринків до систем охорони здоров'я. Одним із типів даних, які мають особливу цінність для прогнозування та прийняття рішень, є часові ряди – послідовності спостережень, упорядкованих у часі. Аналіз таких даних дозволяє виявляти приховані закономірності, сезонні коливання, тенденції, а також здійснювати точні прогнози майбутніх значень на основі історичної інформації.

З розвитком цифрових технологій обсяг доступних даних зростає експоненціально, тому постає завдання створення ефективних та зручних інструментів для їх аналізу. Традиційні підходи, які базуються на ручному аналізі та простих інтерпретаціях, більше не відповідають потребам сучасного користувача. У зв'язку з цим розробка автоматизованих інформаційних систем, що здатні виконувати повноцінний аналіз часових рядів, є актуальним завданням прикладної інформатики.

Актуальність теми полягає в необхідності створення універсального інструменту для аналізу часових рядів, який поєднує зручність використання, візуалізацію даних, підтримку популярних методів статистичного прогнозування та можливість масштабування. Така система може знайти застосування в бізнес-аналітиці, фінансовому плануванні, енергетиці, охороні здоров'я, екології та інших сферах, де важливо аналізувати зміни у часі та прогнозувати майбутні події або показники.

Метою даної кваліфікаційної роботи є розробка інформаційної системи аналізу часових рядів, яка дозволяє здійснювати повний цикл обробки даних – від імпорту вхідної інформації до побудови моделі прогнозування та візуального представлення результатів. Особлива увага приділяється реалізації методів аналізу трендів, сезонності, стаціонарності, а також

побудові моделей типу LSTM, експоненційного згладжування та інших сучасних підходів.

Об’єктом дослідження є процес розробки програмного забезпечення прогнозування фінансових часових рядів.

Методи дослідження, використані в роботі, включають аналіз літературних джерел, порівняння математичних моделей, методи статистичної обробки, експериментальне моделювання та розробку програмного забезпечення.

Робота складається з трьох розділів. У першому розділі розглянуто теоретичні основи аналізу часових рядів та технічного аналізу. У другому – розглянуто архітектура та моделювання інформаційної системи. Третій розділ присвячений програмній реалізації та тестуванню системи.

Результатом роботи є створення прототипу інформаційної системи, яка дозволяє швидко та зручно виконувати аналіз часових рядів і прогнозування з використанням актуальних математичних методів.

1 ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ЧАСОВИХ РЯДІВ У ФІНАНСОВИХ ДАНИХ

1.1 Особливості фінансових часових рядів та технічного аналізу

Фінансові часові ряди є важливим джерелом інформації для аналізу ринкової динаміки та прийняття інвестиційних рішень. Такі ряди – це послідовності значень цін активів (наприклад, акцій або криптовалют), впорядкованих у часі. Їх аналіз дає змогу виявляти закономірності, оцінювати ринкові умови та будувати прогнози.

Ключовими характеристиками таких рядів є висока волатильність, наявність трендів і сезонності, наявність шуму та спекулятивних коливань.

У технічному аналізі використовуються індикатори, які базуються на історичних даних.

У розроблюваній системі використовуються такі основні індикатори:

- SMA (Simple Moving Average) – середнє значення ціни за певний період (наприклад, 50 або 200 днів);
- EMA (Exponential Moving Average) – згладжене середнє з більшою вагою для останніх значень;
- RSI (Relative Strength Index) – індикатор перекупленості/перепроданості активу;
- MACD (Moving Average Convergence Divergence) – співвідношення двох EMA для визначення моменту зміни тренду;
- Bollinger Bands – відхилення ціни від середнього, що вказує на потенційні екстремальні значення;
- ATR (Average True Range) – вимірює волатильність активу.

Ці індикатори дозволяють краще інтерпретувати поведінку ринку та слугують основою для подальшого прогнозування. Вони також є ключовими елементами при побудові автоматизованих торгових стратегій.

1.2 Огляд доступних API для фінансових даних

Сучасні системи аналізу фінансових даних потребують надійних джерел інформації. Розглянемо ключові API, які зазвичай використовуються у фінансових додатках.

Yahoo Finance (посилання: https://finance.yahoo.com/) [7]. Цей інструмент є одним з найпопулярніших серед розробників через свою доступність та широкий спектр підтримуваних активів. Основна перевага – безкоштовний доступ до історичних даних. Однак варто враховувати, що це неофіційний API, який може змінюватися без попередження, а також має певні обмеження на кількість запитів. Для роботи з ним використовується бібліотека yfinance (див. рис. 1.1), на рисунку 1.2 можемо побачити приклад результату.

example.py > ...

```
1  import yfinance as yf
2  data = yf.download("AAPL", start="2024-01-02", end="2024-01-05")
3  print(data.head())
```

Рисунок 1.1 – Приклад запиту Yahoo Finance

Price	Close	High	Low	Open	Volume
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL
Date					
2024-01-02	184.290421	187.070068	182.553143	185.789438	82488700
2024-01-03	182.910522	184.528677	182.096477	182.880742	58414500
2024-01-04	180.587540	181.758954	179.565029	180.825785	71983600

Рисунок 1.2 – Приклад результату Yahoo Finance

Alpha Vantage (посилання: https://www.alphavantage.co/) [11]. Відрізняється більш структурованим підходом до надання даних. Безкоштовний тарифний план дозволяє отримувати детальні фінансові показники, але має жорсткі обмеження – лише 5 запитів на хвилину та 500 на день. Це робить його менш придатним для масштабних досліджень без придбання платного пакету. Приклад використання можна побачити на рисунку 1.3. Також на рисунку 1.4 відображено приклад результату.

```
import requests

url = "https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=AAPL&
apikey=DEMO"
response = requests.get(url).json()

print(f"Данні за 02.06.2025:",response["Time Series (Daily)"]["2025-06-02"])
```

Рисунок 1.3 – Приклад запиту Alpha Vantage

```
Данні за 02.06.2025: {'1. open': '200.2800', '2. high': '202.1300', '
3. low': '200.1200', '4. close': '201.7000', '5. volume': '35423294'}
```

Рисунок 1.4 – Приклад результату Alpha Vantage

Twelve Data (посилання: https://twelvedata.com) [12]. Спеціалізується на роботі з даними реального часу, що особливо цінно для трейдингових систем. Підтримує широкий спектр криптовалют, але безкоштовний план суттєво обмежений. Для повноцінної роботи необхідно використовувати WebSocket або REST API, що потребує додаткових витрат на розробку. Приклад запиту можна побачити на рисунку 1.5.

example.py > ...

```
1   import requests
2
3   api_key = "BALL_API_KEY"
4   symbol = "BTC/USD"
5
6   # Отримання поточної ціни
7   response = requests.get(
8       f"https://api.twelvedata.com/price?symbol={symbol}&apikey={api_key}"
9   )
10
11  print(response.json())
```

Рисунок 1.5 – Приклад запиту Twelve Data

Finnhub (посилання: https://finnhub.io) [13]. Являється потужним API для роботи з фінансовими даними, який надає доступ до широкого спектру інформації. Сервіс особливо корисний для аналітиків, що працюють з фундаментальними показниками компаній, оскільки надає дані про P/E,

дивіденди, фінансову звітність та інші ключові метрики. Приклад запиту бібліотеки відображено на рисунку 1.6.

example.py > ...

```python
import requests

api_key = "ВАШ_API_KEY"
symbol = "AAPL"

response = requests.get(
    f"https://finnhub.io/api/v1/stock/metric?symbol={symbol}&metric=all&token={api_key}"
)

print(response.json())
```

Рисунок 1.6 – Приклад запиту IEX Cloud

Binance (посилання: https://binance-docs.github.io/apidocs/) [14]. Оптимальний вибір для роботи з криптовалютами. Забезпечує дані в реальному часі з високою частотою оновлення. Однак сфера застосування обмежена виключно криптовалютним ринком. Приклади запиту і результату наведено відповідно на рисунках 1.7 та 1.8.

example.py > ...

```python
import requests

response = requests.get('https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT')

print(response.json())
```

Рисунок 1.7 – Приклад запиту Binance API

```
{'symbol': 'BTCUSDT', 'price': '105952.27000000'}
```

Рисунок 1.8 – Приклад результату Binance API

Проведений аналіз сучасних API для роботи з фінансовими даними дозволив виявити ключові особливості та відмінності між популярними сервісами. Кожен з розглянутих інструментів має свою спеціалізацію та оптимальну сферу застосування.

Yahoo Finance залишається найдоступнішим рішенням для старту проєктів, проте його неофіційний статус створює певні ризики для

довгострокових систем. Alpha Vantage пропонує більш структурований підхід, але суттєво обмежує кількість запитів у безкоштовному режимі.

Для роботи з даними реального часу особливий інтерес представляють Twelve Data та Binance API, останній є беззаперечним лідером для криптовалютних ринків. Finnhub, у свою чергу, виділяється комплексною підтримкою фундаментальних показників.

Важливим аспектом при виборі API є не лише технічні можливості, але й економічна доцільність їх використання. Для промислових рішень варто розглядати комерційні пропозиції, тоді як для дослідницьких цілей часто достатньо можливостей безкоштовних тарифів.

Архітектура сучасної аналітичної системи повинна враховувати можливість інтеграції різних джерел даних, що дозволить компенсувати обмеження окремих API та забезпечити стабільність роботи навіть при тимчасовій недоступності одного з джерел. Оптимальним підходом є реалізація модульної системи з єдиним інтерфейсом взаємодії з різними API.

1.3 Інструменти та технології реалізації системи прогнозування

Сучасні інструменти аналізу часових рядів ґрунтуються на потужних технологічних рішеннях, які дозволяють ефективно обробляти великі обсяги даних та будувати точні прогностичні моделі. У даній роботі використано комплексний підхід, що поєднує перевірені бібліотеки для роботи з даними, передові фреймворки машинного навчання та інструменти візуалізації.

1.3.1 Платформи для роботи з даними

Для забезпечення ефективного збору фінансової інформації в системі реалізовано підтримку двох API: Yahoo Finance та Alpha Vantage.

Обидва джерела надають доступ до історичних даних фондового ринку, проте мають суттєві відмінності за рівнем структурованості, частотою оновлення, форматом відповіді та обмеженнями за кількістю запитів.

Yahoo Finance є неофіційним, але широко використовуваним джерелом фінансових даних, зручним для швидкої інтеграції за допомогою Python-бібліотеки yfinance. Основною перевагою є відсутність необхідності авторизації через API-ключ, що спрощує роботу у прототипах та невеликих проєктах. Проте, неофіційний статус API створює ризики втрати сумісності з майбутніми оновленнями сервісу. Також можуть спостерігатися обмеження за кількістю запитів у короткий період [7].

Alpha Vantage, на відміну від Yahoo Finance, є повністю офіційним API з чіткою документацією, розмежуванням функціональності та ключовими методами для доступу до часових рядів, технічних індикаторів і фундаментальних показників компаній. Доступ надається через API-ключ, що дозволяє контролювати навантаження, але також встановлює обмеження – не більше п’яти запитів за хвилину на безкоштовному тарифі. Цей API підтримує формат JSON і дозволяє легко збирати дані в аналітичному модулі системи. У разі перевищення ліміту сервер повертає повідомлення про помилку, яке обробляється через відповідні винятки у коді [11].

Обидва джерела були реалізовані у системі через окремі модулі, Абстрактний метод BaseDataSource дозволяє динамічно використовувати джерело залежно від параметрів користувача. Така архітектура сприяє масштабованості та подальшому розширенню системи на нові джерела (наприклад, Binance чи Finnhub).

Таким чином, використання двох джерел забезпечує баланс між стабільністю та гнучкістю.

Для технічного аналізу застосовується TA-Lib (див. рис. 1.9), яка дозволяє ефективно обчислювати індикатори. Наприклад, розрахунок ковзного середнього [8].

```
import talib

data['SMA_20'] = talib.SMA(data['close'], timeperiod=20)
```

Рисунок 1.9 – Приклад використання TA-lib

1.3.2 Фреймворки машинного навчання

У задачах прогнозування фінансових часових рядів важливим є вибір відповідного фреймворку машинного навчання, який дозволяє ефективно обробляти послідовні дані. У цьому підрозділі розглянуто найбільш поширені фреймворки, які застосовуються для моделювання таких задач.

TensorFlow – один з найпопулярніших фреймворків для побудови моделей глибокого навчання (див. рис. 1.10), розроблений Google. Його високорівневе API Keras забезпечує простий та інтуїтивно зрозумілий спосіб побудови моделей. TensorFlow підтримує GPU-прискорення, масштабування, та багату екосистему [9].

Для задач прогнозування використовуються рекурентні нейронні мережі, зокрема LSTM.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    LSTM(64, input_shape=(60, 1), return_sequences=True),
    LSTM(64),
    Dense(1)
])
```

Рисунок 1.10 – Приклад побудови LSTM у TensorFlow

TensorFlow з Keras є масштабованим та активно підтримуваним фреймворком, який вирізняється простотою створення моделей завдяки високорівневому API, однак його використання на низькому рівні може бути складним, а сам фреймворк доволі вимогливий до ресурсів.

PyTorch – конкурент TensorFlow, розроблений Meta. Відомий своєю гнучкістю, зручністю для досліджень і прозорим дебагом. Його динамічні обчислювальні графи роблять його особливо зручним для прототипування. Приклад використання фреймворку зображено на рисунку 1.11.

```
import torch.nn as nn

class LSTMModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.lstm = nn.LSTM(input_size=1, hidden_size=64, num_layers=2,
                            batch_first=True)
        self.fc = nn.Linear(64, 1)

    def forward(self, x):
        out, _ = self.lstm(x)
        return self.fc(out[:, -1, :])
```

Рисунок 1.11 – Приклад побудови LSTM у PyTorch

PyTorch надає гнучкий контроль над архітектурою моделей і добре підходить для експериментів завдяки прозорості та підтримці динамічних обчислювальних графів, однак у порівнянні з TensorFlow він менш абстрагований, що робить розробку складнішою для новачків [23].

Scikit-learn – фреймворк зосереджений на класичних методах машинного навчання: регресії, класифікації, кластеризації. Він не підтримує послідовні моделі напряму, однак широко використовується для задач регресії з фіксованими ознаками, у прикладі на рисунку 1.12 можна побачити будування моделі випадкового лісу за допомогою фреймворку.

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
model.fit(X_train, Y_train)
```

Рисунок 1.12 – Приклад моделі випадкового лісу за допомогою Scikit-learn

Scikit-learn вирізняється простотою у використанні, високою швидкістю роботи та стабільністю, однак не підтримує послідовні дані без попередньої

ручної трансформації, що обмежує його застосування в задачах прогнозування часових рядів [24].

XGBoost та LightGBM – високопродуктивні фреймворки градієнтного бустингу, які часто демонструють відмінні результати в задачах з табличними даними. Можуть бути адаптовані для часових рядів за допомогою фічерної інженерії.

Отже забезпечують високу точність і швидкість завдяки ефективній реалізації градієнтного бустингу, однак для задач із часовими рядами вимагають ручної інженерії ознак, оскільки не працюють з послідовностями напряму [18, 21].

Facebook Prophet – бібліотека для часових рядів, орієнтована на бізнес-користувачів. Простий синтаксис, автоматична обробка трендів та сезонності.

Бібліотека вирізняється простим синтаксисом і зручністю для нетехнічних користувачів, автоматично враховуючи сезонність і тренди, однак має обмежену гнучкість у налаштуванні моделей і погано підходить для аналізу високочастотних фінансових даних [22].

Таким чином, вибір фреймворку для задач прогнозування фінансових часових рядів значною мірою залежить від складності моделі, типу вхідних даних та вимог до продуктивності.

TensorFlow/Keras демонструє високу ефективність у побудові глибоких моделей з підтримкою часових залежностей, що робить його оптимальним вибором для систем, орієнтованих на LSTM [5].

PyTorch є гнучкою альтернативою з потенціалом для експериментальних досліджень, тоді як Scikit-learn, XGBoost і Prophet доцільно використовувати у задачах з табличними ознаками, де часовий контекст явно задано або може бути врахований опосередковано [5].

У даній роботі було обґрунтовано використання TensorFlow через його здатність працювати з послідовними даними, масштабованість, наявність підтримки LSTM-архітектур та зручність інтеграції з іншими компонентами системи [5].

1.3.3 Інструменти візуалізації

У задачах, пов’язаних з фінансовим аналізом та прогнозуванням часових рядів, візуалізація відіграє важливу роль. Вона дозволяє не лише перевірити гіпотези щодо поведінки ринку, а й виявити приховані закономірності, порушення тренду, коливання волатильності тощо. Нижче наведено аналіз найпоширеніших бібліотек візуалізації, які застосовуються в Python для роботи з фінансовими даними.

Matplotlib – це універсальна бібліотека для побудови графіків у Python, яка забезпечує широкі можливості для статичної візуалізації. Завдяки своїй низькорівневій структурі, вона дозволяє точно контролювати елементи графіку: осі, підписи, кольори, стилі ліній тощо.

У фінансовому аналізі Matplotlib часто використовується для побудови базових графіків зміни ціни активу, накладання технічних індикаторів, візуалізації сигналів купівлі/продажу. Приклад використання бібліотеки зображено на рисунку 1.13.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(example_data['Date'], example_data['Close'], label='Actual')
plt.plot(example_data['Date'], example_data['Prediction'], label='Predicted')
plt.title('Порівняння прогнозованих та фактичних значень')
plt.legend()
plt.show()
```

Рисунок 1.13 – Приклад використання Matplotlib

Matplotlib забезпечує повний контроль над виглядом графіків, має надійну інтеграцію з pandas і є перевіреним інструментом для побудови візуалізацій, однак не підтримує інтерактивність «з коробки» і вимагає значного обсягу коду для створення складних або динамічних візуалізацій [15].

Plotly – це бібліотека для побудови інтерактивних графіків, яка підтримує масштабування, спливаючі підказки, перемикання між режимами

перегляду. Графіки можна легко інтегрувати у вебдодатки або Jupyter Notebook.

У фінансових задачах Plotly зручний для відображення довгих часових рядів, де користувачу важливо швидко переміщуватись по часовій осі та збільшувати окремі ділянки графіку. На рисунку 1.14 зображено приклад використання бібліотеки.

```
import plotly.express as px

fig = px.line(data, x='Date', y='Close', title='Графік цін')
fig.update_xaxes(rangeslider_visible=True)
fig.show()
```

Рисунок 1.14 – Приклад використання Plotly

Інструмент поєднує сучасний вигляд і високу інтерактивність, легко інтегрується в Jupyter або вебдодатки, проте його налаштування може бути складнішою порівняно з Matplotlib, а для повноцінної роботи потрібен браузерне середовище [16].

Mplfinance – це спеціалізована бібліотека, побудована на базі Matplotlib, призначена для побудови фінансових графіків, таких як японські свічки, бар-чарти, графіки об’ємів торгів, накладання технічних індикаторів. Приклад використання якої зображено на рисунку 1.15.

```
import mplfinance as mpf

mpf.plot(data, type="candle", mav=(20, 50), volume=True, style="yahoo")
```

Рисунок 1.15 – Приклад використання Mplfinance

Вона є найпридатнішою для відображення даних у класичному біржовому форматі. Свічкові графіки дозволяють побачити не лише тренд, а й поведінку ціни протягом кожного інтервалу (відкриття, максимум, мінімум, закриття).

Бібліотеку зручно використовувати завдяки готовим шаблонам для фінансових візуалізацій і простому налаштуванню свічкових графіків, проте вона має обмежену гнучкість у поєднанні елементів і менш придатна для побудови кастомних або комбінованих графіків [17].

Bokeh – бібліотека для створення інтерактивних графіків із підтримкою HTML, JavaScript і інтеграції з браузером. Підтримує векторні та масштабовані графіки з можливістю взаємодії користувача з елементами (наприклад, наведіть курсор – і з'являється значення). Для фінансів використовується рідше, але може бути корисною для побудови унікальних візуальних аналітичних панелей. Приклад використання зображено на рисунку 1.16.

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

output_notebook()

p = figure(title="Графік цін", x_axis_label='Дата', y_axis_label='Ціна закриття')
p.line(data['Date'], data['Close'], line_width=2)
show(p)
```

Рисунок 1.16 – Приклад використання Bokeh

Незважаючи на те, що Bokeh забезпечує глибокі підлаштування та широкий спектр інтерактивних можливостей у вебінтерфейсі, її складність налаштування і менш поширене застосування у фінансовій сфері обмежують її використання в цій галузі [19].

Plotnine – реалізація популярної R-бібліотеки ggplot2 в середовищі Python. Працює на основі концепції «граматики графіки», що робить синтаксис надзвичайно декларативним і чистим.

У фінансовому аналізі її застосовують для дослідження кореляцій, сезонності, порівняння декількох часових рядів. На рисунку 1.17 можна побачити приклад використання.

Plotnine пропонує інтуїтивний та декларативний підхід до побудови графіків, що особливо цінують при роботі з даними у форматі pandas. Проте,

її обмежений функціонал щодо тонкого налаштування візуалізацій і недостатня інтеграція з іншими інструментами можуть ускладнювати реалізацію більш складних аналітичних сценаріїв [20].

```
from plotnine import *

ggplot(data) + aes(x="Date", y="Close") + geom_line() + ggtitle("Графік цін")
```

Рисунок 1.17 – Приклад використання Plotnine

У сучасному середовищі Python існує низка інструментів візуалізації, які можуть ефективно застосовуватись у задачах аналізу фінансових часових рядів. Найбільш універсальним і стабільним є Matplotlib, який широко застосовується у наукових та інженерних середовищах. Для спеціалізованої фінансової графіки найзручнішим є Mplfinance. У випадках, коли необхідна інтерактивність і вебінтерфейси, перевагу варто віддати Plotly або Bokeh. Plotnine стане в пригоді при декларативному підході до аналізу та підготовці статистичних звітів.

1.4 Аналіз вимог розробки

Розробка інформаційної системи для аналізу часових рядів фінансових даних потребує чіткого визначення вимог, що враховують як технічні особливості обробки даних, так і потреби користувачів. На основі аналізу предметної області було сформульовано основні функціональні та нефункціональні вимоги до системи.

Функціональні вимоги охоплюють три основні напрямки: завантаження, аналіз та прогнозування даних. Система повинна забезпечувати:

– отримання фінансових даних із зовнішніх джерел (API);
– можливість вибору активів (акції, валюти, криптовалюти);
– аналіз даних включаючий розрахунок популярних технічних індикаторів, таких як SMA, EMA, RSI, MACD, Bollinger Bands;

– перевірка якості вхідних даних, що дозволяє уникнути помилок при розрахунках;
– прогнозування використовуючи модель типу LSTM, яка добре підходить для роботи з часовими рядами (такий вибір зумовлений здатністю моделі враховувати часові залежності в даних, встановлені вимоги до точності прогнозу (RMSE < 0.06) та часу навчання моделі (до 3 хвилин));
– відображення результатів прогнозування графічно з відповідними позначками рівня достовірності.

Нефункціональні вимоги стосуються продуктивності, стабільності та гнучкості системи. Система має:

– реагувати на типові запити не довше ніж за 5 секунд і підтримувати паралельну обробку кількох активів;
– передбачена оптимізація обчислень для різних апаратних конфігурацій (CPU/GPU).

З метою забезпечення надійності реалізовані механізми обробки помилок, пов’язаних із мережею або некоректними параметрами, а також система резервного копіювання результатів. Модульна архітектура дозволяє розширювати функціональність: додавати нові індикатори, моделі або джерела даних.

У результаті сформульовані вимоги дозволили обґрунтувати використання нейронної мережі LSTM як основного методу прогнозування.

Це рішення забезпечує належний рівень точності при збереженні продуктивності системи. Гнучка архітектура сприяє подальшому розвитку системи та її адаптації до змін у фінансовому середовищі.

2 АРХІТЕКТУРА ТА МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Прогнозування часових рядів є важливим інструментом для аналізу даних, який знаходить широке застосування в різних галузях, зокрема в економіці, фінансах, маркетингу, а також у природничих науках та техніці. Часові ряди – це послідовність спостережень, які змінюються в часі. Метою прогнозування є передбачення значень цих рядів на основі наявних даних, що дозволяє зробити обґрунтовані прогнози та прийняти стратегічні рішення.

Існує безліч підходів до прогнозування часових рядів, кожен з яких має свої переваги та недоліки. Класичні статистичні методи, такі як ARIMA, авторегресивні моделі та ковзні середні, дають можливість знаходити залежності між спостереженнями та здійснювати прогнози на їх основі. Проте з розвитком технологій машинного навчання та глибокого навчання з’явилися нові, більш складні підходи, здатні працювати з великими обсягами даних і виявляти складніші закономірності [3, 5].

У цьому розділі розглядаються основні методи прогнозування часових рядів, а також обґрунтовується вибір найбільш ефективного підходу для реалізації в рамках цієї роботи. Розглянуто можливості використання нейронних мереж, зокрема LSTM, для вирішення задачі прогнозування фінансових часових рядів [1, 6].

2.1 Архітектура системи

Інформаційна система прогнозування фінансових часових рядів реалізована на основі модульної багаторівневої архітектури, яка забезпечує гнучкість, розширюваність та чітке розмежування відповідальності між компонентами. Загальна структура представлена на діаграмі компонентів, де вказано основні логічні блоки системи та характер їх взаємодії. Розглянемо основні компоненти системи.

Компонент збору даних. Цей блок відповідає за інтеграцію із зовнішніми джерелами фінансової інформації – зокрема, API Yahoo Finance та Alpha Vantage. Він реалізує логіку динамічного вибору джерела, обробку HTTP-запитів, валідацію отриманих даних проводить перевірку на відсутність, дублікати, некоректні значення, а також збереження у базу даних. У разі відсутності підключення або перевищення ліміту – реалізовані механізми обробки винятків.

Компонент обробки даних. Цей блок виконує нормалізацію вхідних даних наприклад, за допомогою MinMaxScaler, а також обчислює технічні індикатори, такі як SMA, RSI, MACD, Bollinger Bands. Для цього використовується бібліотека TA-Lib. Також реалізовано формування часових вікон для LSTM-мережі. Підготовка тренувальних та тестових наборів даних.

Компонент моделювання. Цей блок відповідає за навчання та виведення прогнозних моделей. Основна модель реалізована на основі рекурентної нейронної мережі LSTM 2 шари по 50 нейронів, побудованої з використанням TensorFlow/Keras. Також передбачено збереження навченої моделі у форматі .h5, її повторне завантаження та покроковий прогноз на кілька днів уперед. У модуль також можуть бути інтегровані альтернативні моделі для порівняння (RandomForest, XGBoost, тощо).

Компонент візуалізації. Цей блок відповідає за генерацію графіків із відображенням фактичної ціни, індикаторів та прогнозованих значень. Графіки будуються за допомогою Matplotlib. Додатково формується текстова аналітична панель зі значеннями ключових показників.

Компонент бази даних. Для зберігання історичних фінансових даних, результатів моделювання та кешування запитів використовується реляційна база даних MySQL. Структура схеми відображена на рисунку 2.1. Система підтримує створення, оновлення та вибірку даних.

Загальна взаємодія компонентів. Зовнішній користувач ініціює запит наприклад, обирає актив. DataManager перевіряє наявність даних у базі або звертається до зовнішнього API. Після отримання даних вони передаються до процесу обробки, де нормалізуються та збагачуються технічними

індикаторами. Далі підготовлений набір надсилається до процесу створення моделі та для здійснення прогнозу. Результати передаються у модуль візуалізації, де відображаються графіки та формуються аналітичні панелі.

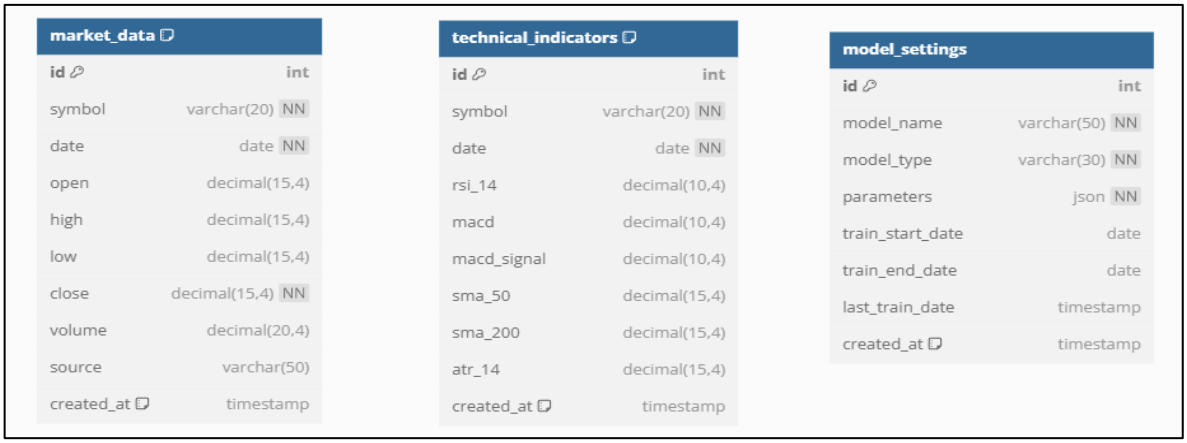


Рисунок 2.1 – Схема бази даних

UML-діаграма компонентів. На рисунку 2.2 наведено діаграму компонентів UML, яка відображає логічну структуру системи та взаємозв'язки між її модулями. Кожен компонент взаємодіє через чітко визначений інтерфейс, що дозволяє легко масштабувати або замінювати окремі частини системи без впливу на інші.

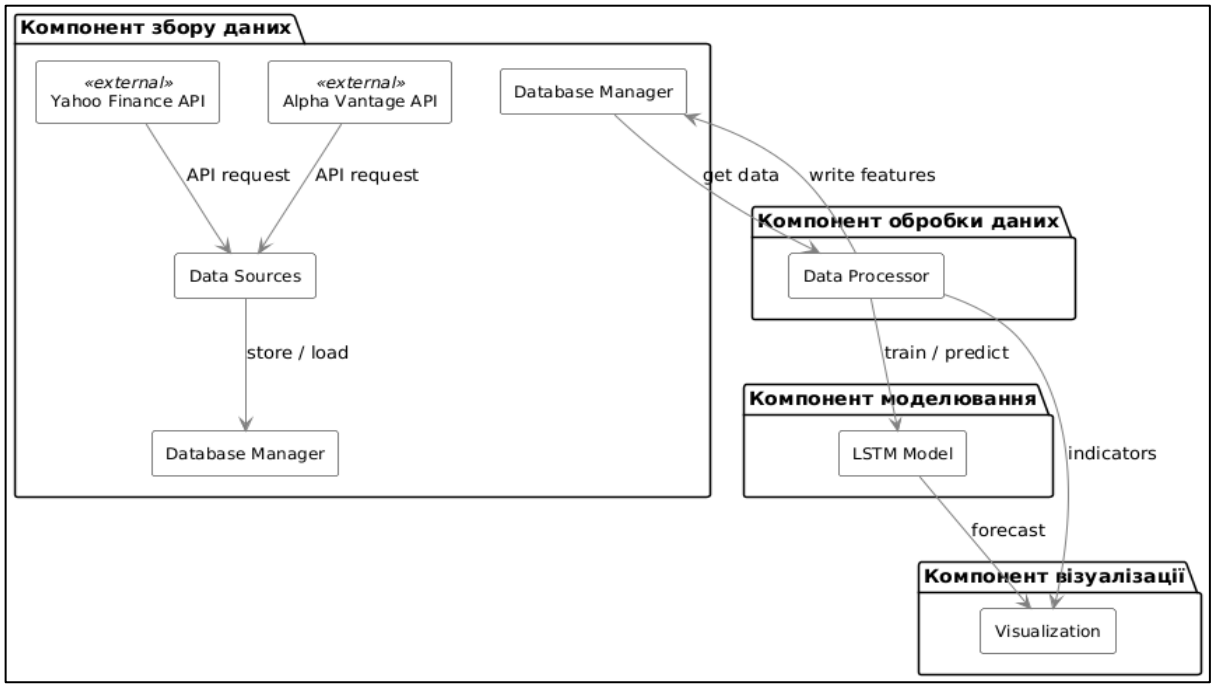


Рисунок 2.2 – UML-Діаграма компонентів

2.2 Діаграми проєктування

Для забезпечення зрозумілості структури програмного забезпечення та процесів взаємодії між його компонентами в системі використано діаграми класів та послідовностей.

Діаграма класів відображає логічну структуру об’єктно-орієнтованих елементів системи. Вона демонструє ключові класи, методи та зв’язки між ними, зокрема, відношення використання, залежності та агрегації. На рисунку 2.3 представлено діаграму класів розроблюваної системи.

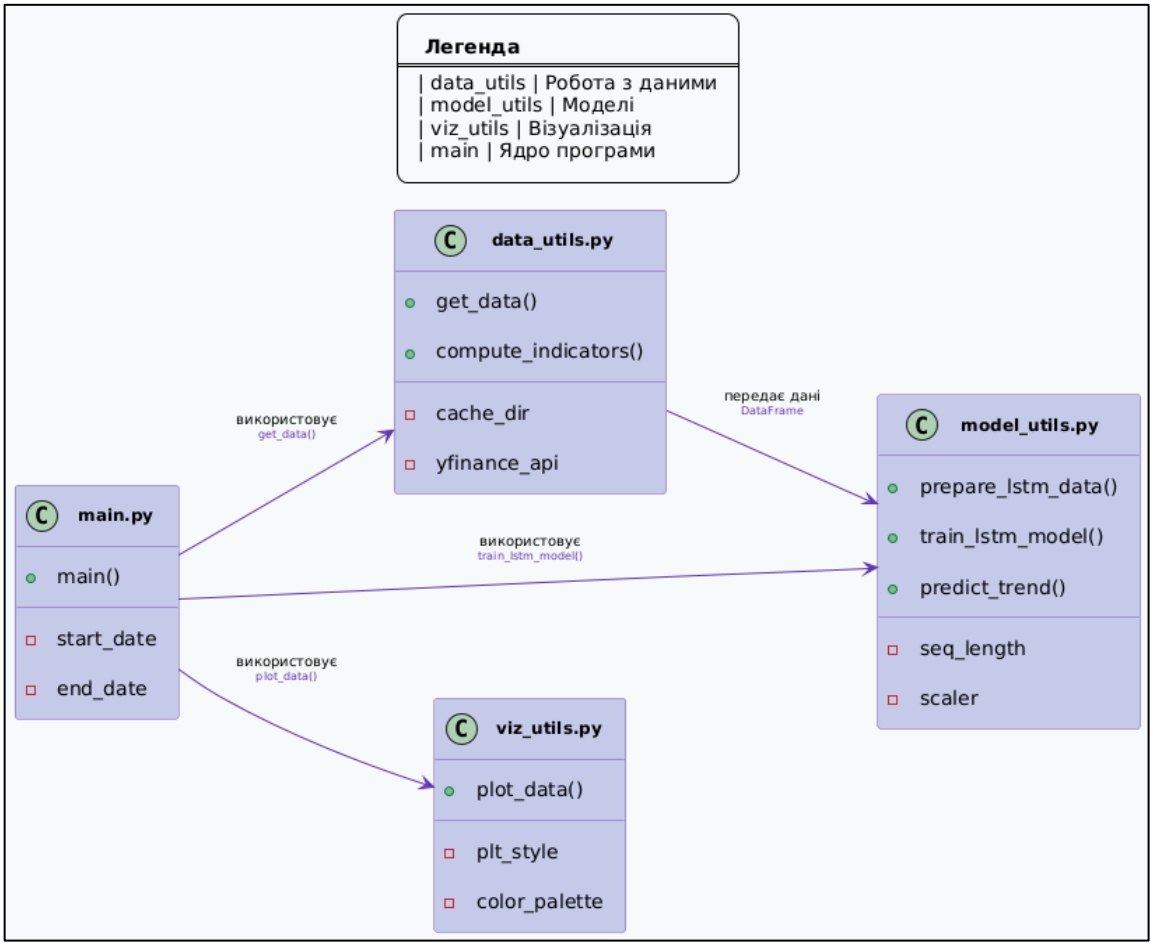


Рисунок 2.3 – Діаграма класів

На діаграмі зображено основні класи, такі як:

– ModelUtils який відповідає за створення та тренування моделі (містить методи prepare_lstm_data(), train_lstm_model(), predict_multiple_days() та змінні конфігурації LSTM);

– DataUtils клас виконує збір, очищення та підготовку даних (реалізує методи get_data(), compute_indicators());
– VizUtils цей клас відповідає за візуалізацію результатів (має метод plot_data() для побудови графіків);
– Main є центральною точкою запуску системи, координує виклики функцій між модулями.

Зв’язки на діаграмі показують, що Main використовує методи з усіх трьох класів (ModelUtils, DataUtils, VizUtils), а ModelUtils безпосередньо взаємодіє з об’єктами, підготовленими DataUtils.

Діаграма послідовностей демонструє динамічну взаємодію між компонентами системи у межах одного сценарію – побудови прогнозу на основі даних, отриманих від зовнішнього API.

Діаграма з рисунку 2.4 демонструє повну логіку роботи системи – від вхідного запиту до отримання результату у вигляді графіків.

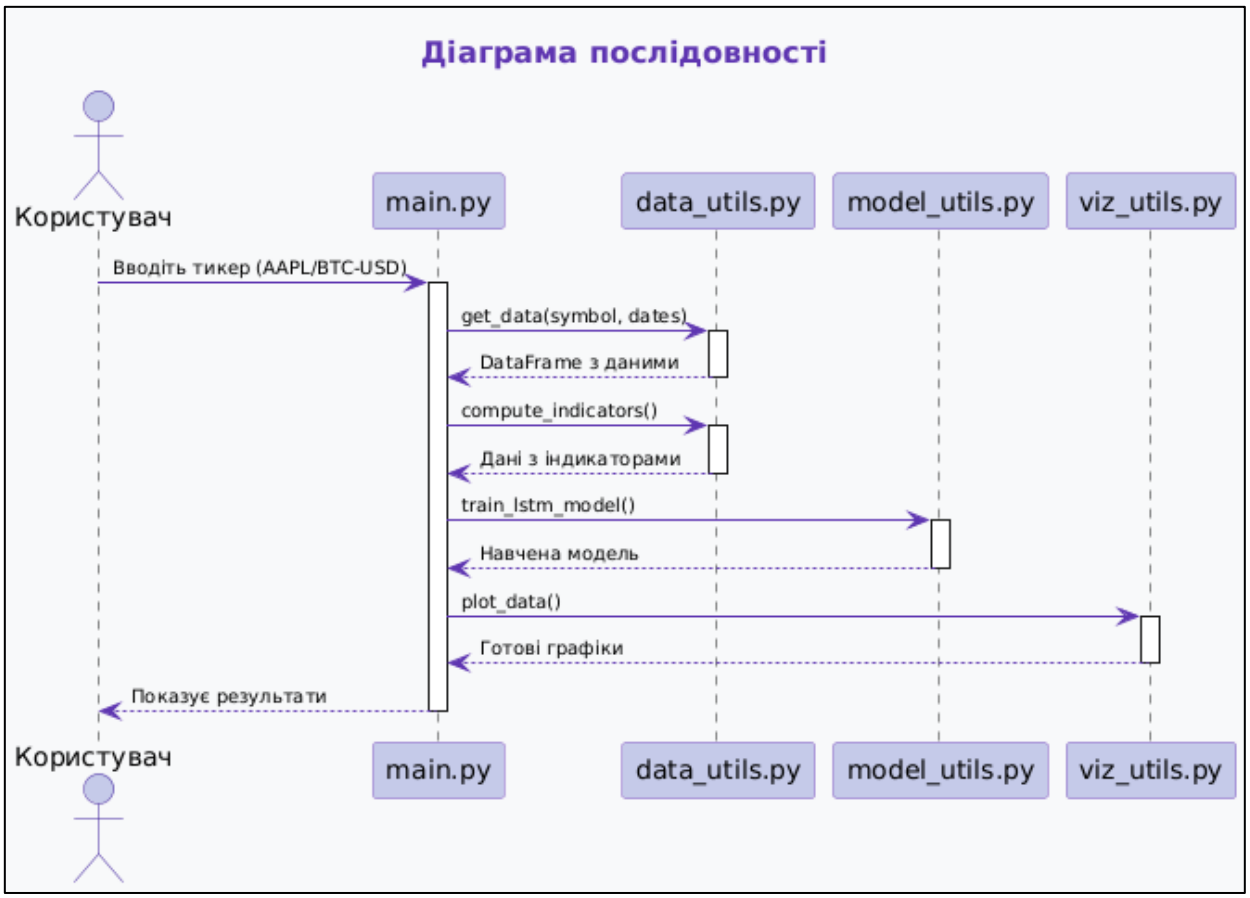


Рисунок 2.4 – Діаграма послідовності

2.3 Логіка роботи системи та сценарій взаємодії з користувачем

Розроблена інформаційна система аналізу часових рядів працює за принципом консольного застосунку, який забезпечує інтерактивну взаємодію з користувачем та проводить технічний аналіз фінансових даних з можливістю прогнозування.

Основна логіка реалізована у модулі main.py, що виконує послідовність дій:

а) ввід користувачем фінансового інструменту – система запитує тикер (наприклад, AAPL, BTC-USD);

б) вибір користувачем джерела даних – система пропонує доступні джерела даних (Yahoo Finance, Alpha Vantage);

в) завантаження історичних даних з обраного користувачем джерела при цьому реалізовано зберігання даних у базу даних для зменшення навантаження на API при повторних запитах;

г) розрахунок технічних індикаторів – застосовується бібліотека TA-Lib для обчислення індикаторів, таких як SMA, EMA, RSI, ATR, MACD та смуги Боллінджера;

д) візуалізація даних – модуль viz_utils.py будує графік цін з нанесеними індикаторами та окрему аналітичну панель, де виводяться останні значення індикаторів;

е) прогнозування – якщо знайдено збережену модель (model_utils.py), система може її завантажити та виконати прогноз; у разі її відсутності, користувачу пропонується навчити модель LSTM на основі останніх даних; після навчання модель зберігається для подальшого використання.

Таким чином, користувач отримує простий і послідовний сценарій взаємодії, що охоплює:

– аналіз поточного стану активу;
– короткостроковий прогноз ціни;
– візуальну оцінку технічних сигналів.

Такий підхід дозволяє ефективно використовувати як класичні методи технічного аналізу, так і сучасні засоби машинного навчання в єдиній інтегрованій системі [3, 5].

У цьому розділі було розглянуто архітектуру, основні компоненти та логіку функціонування інформаційної системи аналізу часових рядів. Система забезпечує повний цикл обробки фінансових даних – від завантаження та технічного аналізу до візуалізації результатів і побудови прогнозу за допомогою нейронної мережі. Простий інтерфейс та модульна структура забезпечують гнучкість та зручність розширення системи в майбутньому.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

У цьому розділі розглядається процес проєктування та поетапної реалізації інформаційної системи, призначеної для аналізу та прогнозування часових рядів. Основна мета створення такої системи – надати користувачеві інструмент для обробки вхідних даних, побудови моделей прогнозування та візуалізації результатів у зручному вигляді.

Особливу увагу приділено вибору інструментів розробки, архітектурі програмного забезпечення, реалізації функціональних компонентів та побудові користувацького інтерфейсу. Було враховано як функціональні вимоги до системи, так і обмеження, пов’язані з обсягами даних, швидкодією та зручністю використання.

Результатом є працездатна інформаційна система, яка реалізує базові та розширювані алгоритми прогнозування, дозволяє зберігати та аналізувати результати, а також має простий та зрозумілий інтерфейс для користувача.

3.1 Архітектурні рішення та ключові алгоритми

Програмна реалізація системи спроєктована з використанням модульного підходу, що дозволяє чітко розділити етапи обробки даних і забезпечити легку масштабованість та модифікацію окремих компонентів. Система складається з кількох основних модулів: завантаження даних, розрахунок технічних індикаторів, підготовка даних для моделювання, побудова та тренування моделі, прогнозування та візуалізація результатів.

Архітектурний підхід. Система побудована на основі pipeline-архітектури, де кожен етап є незалежним модулем, що дозволяє зберігати гнучкість і масштабованість при роботі з великими обсягами даних та різними варіантами використання. Це дає змогу модифікувати або додавати нові етапи обробки без впливу на інші частини системи.

Особливе значення приділено ефективному управлінню даними. Для мінімізації запитів до зовнішнього API і зменшення часу завантаження Для досягнення ефективного управління даними впроваджено реляційну базу даних. Це дозволяє зберігати вже завантажені дані у БД, і при повторних запитах зчитувати їх локально, що значно знижує навантаження на мережу та покращує швидкість роботи.

Обробка даних для моделювання. Підготовка даних є важливим етапом у системі, оскільки нейронні мережі, зокрема LSTM, вимагають специфічного формату вхідних даних. Підготовка даних реалізована в два основних етапи (див. рис. 3.1).

Першим етапом є нормалізація даних, для масштабування даних використовується метод MinMaxScaler, що приводить усі значення до діапазону [0, 1]. Це дозволяє нейронній мережі навчатися більш стабільно і швидко, оскільки у всіх ознак однакова шкала.

Другий етап це побудова ковзних вікон, на основі нормалізованих даних будується ковзаюче вікно, яке враховує контекст попередніх значень для прогнозування майбутніх. Це дозволяє моделі навчатися на часових залежностях.

```python
def prepare_lstm_data(data, seq_length=60):
    if data is None or len(data) < seq_length + 1:
        raise ValueError(f"Недостатньо даних. Потрібно мінімум {seq_length + 1} точок, отримано {len(data) if data is not None else 0}")

    # Витягуємо значення close цін
    close_prices = data[['close']].values if isinstance(data, pd.DataFrame) else data

    # Нормалізація даних
    scaler = MinMaxScaler(feature_range=(0, 1))
    data_scaled = scaler.fit_transform(close_prices)

    # Формування часових вікон
    X, y = [], []
    for i in range(seq_length, len(data_scaled)):
        X.append(data_scaled[i-seq_length:i, 0])
        y.append(data_scaled[i, 0])

    X, y = np.array(X), np.array(y)
```

Рисунок 3.1 – Функція підготовки даних

Архітектура моделі LSTM. Основним елементом прогностичного модуля є модель LSTM, яка є розширенням звичайної RNN і дозволяє зберігати інформацію про попередні стани на довгих часових відрізках.

Модель використовується для передбачення майбутніх значень на основі фінансових часових рядів [2, 4].

Модель складається з таких шарів та має наступну структуру (рис. 3.2):

– вхідний шар: приймає послідовність нормалізованих цін довжиною 60 (shape = (60, 1));
– перший LSTM-шар: 50 нейронів, return_sequences=True, що дозволяє передавати повну послідовність далі;
– dropout-шар: з імовірністю 0.2, для зменшення перенавчання;
– другий LSTM-шар: 50 нейронів;
– batch Normalization: стабілізує навчання, нормалізуючи активації між шарами.

```
model = Sequential()
model.add(Input(shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50, return_sequences=True))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean_squared_error')
```

Рисунок 3.2 – Структура LSTM моделі

Модель навчається за 10 епох при розмірі пакета batch_size=32, з валідаційним розділенням 10%. Функція втрат – mean_squared_error, оптимізатор – Adam, який добре працює для регресійних задач із невеликою кількістю даних.

Дані готуються за допомогою функції prepare_lstm_data(), частину з якої можна побачити на рисунку 3.3 з модуля model_utils.py. На вхід надходить датафрейм з колонкою close.

– нормалізація: масштабування за допомогою MinMaxScaler до [0, 1];
– формування послідовностей: ковзне вікно довжиною 60;
– розділення: 80% – тренувальна вибірка, 20% – тестова;

– форматування для LSTM: перетворення до виду (samples, timesteps, features).

```
# Розділення на тренувальну та тестову вибірки
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Зміна форми даних для LSTM [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

Рисунок 3.3 – Підготовка даних для LSTM моделі

Для оцінки точності моделі було проведено серію експериментів (див. рис. 3.4), результати яких наведено у таблиці 3.1.

Дані взяті з API Yahoo Finance (через yfinance), актив: AAPL, період: 3 роки; крок: 1 день.

Навчання виконувалося на 80% даних. Для оцінки використовували:

– RMSE (Root Mean Squared Error) – точність прогнозу;

– час моделювання – загальний час навчання та прогнозу.

```
# === 4. MA ===
start = time.time()
rolling_mean = pd.Series(y_train).rolling(3).mean().dropna()
pred_ma = np.full(len(y_test), rolling_mean.iloc[-1])
results.append(print_result("MA", y_test, pred_ma, start))

# === 5. Лінійна регресія ===
start = time.time()
lr = LinearRegression()
lr.fit(X_train, y_train)
pred_lr = lr.predict(X_test)
results.append(print_result("Лін.регресія", y_test, pred_lr, start))

# === 6. SVM ===
start = time.time()
svm = SVR()
svm.fit(X_train, y_train)
pred_svm = svm.predict(X_test)
results.append(print result("SVM", y test, pred svm, start))
```

Рисунок 3.4 – Приклад коду для проведення обчислювального експерименту

Таблиця 3.1 – Порівняння методів прогнозування часових рядів

Метод	Переваги	Точність (RMSE)	Час обчислень
ARIMA	Комбінує AR та MA, добре для стаціонарних рядів	0.22	0.06 сек
SARIMA	ARIMA з урахуванням сезонності	0.16	2.04 сек
MA(ковзне середнє)	Проста реалізація, добре для згладжування шуму	0.16	0.01 сек
Лінійна регресія	Швидкий, але не враховує часові залежності	0.016	0.01 сек
SVM	Ефективний для малих вибірок, складний у налаштуванні	0.2	0.01 сек
LSTM	Найкраща якість прогнозу, зберігає довгострокові залежності (обрано для системи)	0.035	3.77 сек
GRU	Спрощена версія LSTM з гарною ефективністю	0.03	4.19 сек
RNN	Враховує послідовності, але має проблему зникаючого градієнта	0.065	2.33 сек

Розглянемо основні алгоритми, що реалізовані у системі.

Алгоритм завантаження даних: за допомогою бібліотеки yfinance відбувається асинхронне завантаження даних за заданими тікерами. Для зменшення кількості запитів реалізовано кешування, що дозволяє повторно використовувати завантажені дані без необхідності кожного разу звертатися до зовнішнього API.

Розрахунок технічних індикаторів: для аналізу фінансових ринків застосовуються стандартні технічні індикатори, такі як RSI, SMA, MACD та інші, які реалізовані через бібліотеку TA-Lib. Це дозволяє швидко і ефективно отримувати необхідні значення для подальшого аналізу.

Прогнозування: нейронна мережа LSTM, навчається на даних з використанням технік глибокого навчання для передбачення майбутніх значень. Для цього використовуються історичні дані, що дозволяють створити точні прогнози на основі часових залежностей.

Цей підхід забезпечує високу ефективність та точність при прогнозуванні фінансових інструментів та дозволяє гнучко масштабувати систему під різні варіанти використання.

3.2 Стратегія тестування інформаційної системи

Система аналізу часових рядів пройшла ретельне тестування, що включало як мануальну перевірку основних функцій, так і автоматизоване тестування ключових компонентів.

Стратегія тестування. Тестування проводилось за такими напрямками:

– модульне тестування: перевірка окремих компонентів системи;
– інтеграційне тестування: перевірка взаємодії між модулями;
– функціональне тестування: перевірка виконання основних сценаріїв роботи.

Автоматизоване тестування. Для автоматизації тестування використовувався фреймворк pytest [10]. Було створено тестовий набір який включає в себе: тести роботи з базою даних, тести модуля DataUtils (див. рис. 3.5), тести моделей машинного навчання.

```python
import pandas as pd
import numpy as np
import pytest

def test_compute_indicators_with_enough_data(data_utils, sample_indicators_data):
    result = data_utils.compute_indicators(sample_indicators_data)
    assert not result.empty
    assert 'SMA_50' in result.columns
    assert 'RSI' in result.columns

def test_indicators_calculation(data_utils):
    """Перевірка коректності обчислення технічних індикаторів"""
    test_data = pd.DataFrame({
        'close': np.sin(np.linspace(0, 10, 250)) * 100 + 200,
        'high': np.sin(np.linspace(0, 10, 250)) * 100 + 220,
        'low': np.sin(np.linspace(0, 10, 250)) * 100 + 180,
        'volume': np.random.randint(1000, 10000, 250)
    })
    result = data_utils.compute_indicators(test_data)
    assert 'RSI' in result.columns
    assert 'MACD' in result.columns

def test_empty_data_handling(data_utils):
    """Перевірка обробки відсутніх даних"""
    result = data_utils.compute_indicators(pd.DataFrame())
    assert result is None

def test_invalid_symbol_handling(data_utils):
    """Перевірка реакції на невалідні тікери"""
    with pytest.raises(ValueError):
        data_utils.get_data("INVALID_SYMBOL", "2023-01-01", "2023-01-03")
```

Рисунок 3.5 – Приклад одного з файлів тестового набору

Тести роботи з базою даних:

– перевірка коректності збереження та отримання даних;

– тестування фільтрації за часовими проміжками;

– перевірка цілісності даних.

Тести модуля DataUtils:

– перевірка розрахунку технічних індикаторів;

– тестування обробки недостатньої кількості даних;

– перевірка механізму кешування даних.

Тести моделей машинного навчання:

– перевірка підготовки даних для LSTM-мереж;

– тестування процесу навчання моделі.

По результатам тестування система успішно пройшла всі основні тести (див. рис. 3.6).

```
collected 10 items

tests/test_data_sources.py::test_base_source_abstract_methods PASSED                [ 10%]
tests/test_data_sources.py::test_yfinance_source_get_historical_data PASSED         [ 20%]
tests/test_data_utils.py::test_compute_indicators_with_enough_data PASSED           [ 30%]
tests/test_data_utils.py::test_indicators_calculation PASSED                        [ 40%]
tests/test_data_utils.py::test_empty_data_handling PASSED                           [ 50%]
tests/test_data_utils.py::test_invalid_symbol_handling PASSED                       [ 60%]
tests/test_database.py::test_db_initialization PASSED                               [ 70%]
tests/test_database.py::test_save_and_retrieve_data PASSED                          [ 80%]
tests/test_models.py::test_predict_trend PASSED                                     [ 90%]
tests/test_viz_utils.py::test_plot_data_basic PASSED                                [100%]
```

Рисунок 3.6 – Результати тестування pytest

Мануальне тестування. Додатково було проведено мануальну перевірку:

– завантаження даних для різних тікерів (AAPL, TSLA, AMZ, тощо);

– перевірка візуалізації технічних індикаторів;

– тестування граничних випадків: неправильні тікери; недоступність API; некоректні вхідні параметри.

Система демонструє стабільну роботу в різних умовах. Автоматизовані тести покривають всі критичні сценарії роботи. Для подальшого розвитку системи передбачається:

– додати тести для нових джерел даних;
– розширити тестування продуктивності;
– додати інтеграційні тести для повного циклу обробки.

3.3 Реалізація функціональних компонентів

Програмна система структурована у вигляді набору окремих функціональних модулів, кожен з яких відповідає за певний етап обробки часових рядів – від завантаження даних до побудови прогнозу та візуалізації результатів. Такий підхід забезпечує високу розширюваність, тестованість і повторне використання коду.

За завантаження даних та обчислення індикаторів відповідає модуль data_utils.py також він відповідає за асинхронне отримання історичних фінансових даних через обране джерело, кешування запитів та обчислення технічних індикаторів. Одна з функцій модуля відображена на рисунку 3.7.

```
def get_data(self, symbol: str, start_date: str, end_date: str, source: str = 'yfinance') -> pd.DataFrame:
    """Отримання даних з обраного джерела"""
    if source not in self.sources:
        raise ValueError(f"Невідоме джерело даних: {source}")

    print(f"Отримання даних з {self.sources[source].source_name}...")
    data = self.sources[source].get_historical_data(symbol, start_date, end_date)
    if data is None or data.empty:
        raise ValueError(f"Не вдалося отримати дані для символу '{symbol}'")

    if not data.empty:
        self.db.save_data(symbol, self.sources[source].source_name, data)
    return data
```

Рисунок 3.7 – Функція get_data

Підготовка та навчання LSTM-моделі знаходиться у модулі model_utils.py який включає функції для підготовки даних, формування навчальних вибірок та побудови нейронної мережі типу LSTM. Модель побудована з регуляризацією Dropout і шаром нормалізації, також передбачено функції збереження та повторного завантаження моделі реалізацію (див. рис. 3.8).

```python
def save_model(model, symbol):
    os.makedirs('models', exist_ok=True)
    model.save(f'models/{symbol}_model.h5')
    print(f"Модель збережена як models/{symbol}_model.h5")

def load_saved_model(symbol):
    model_path = f'models/{symbol}_model.h5'
    if os.path.exists(model_path):
        return load_model(model_path)
    return None
```

Рисунок 3.8 – Функція збереження та повторного завантаження моделі

Модуль візуалізації viz_utils.py в якому для інтерактивної візуалізації використано matplotlib, з можливістю одночасного відображення графіку цін, індикаторів та прогнозів. Невеликий фрагмент коду, який відповідає за це, наведено на рисунку 3.9.

```python
def plot_data(data, forecast=None, forecast_7_days=None):
    plt.style.use('seaborn-v0_8-white')
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 8), gridspec_kw={'width_ratios': [3, 1]})

    # Основний графік (ліва частина)
    ax1.plot(data['Close'], label='Ціна (Close)', color='blue', linewidth=2)
    ax1.plot(data['SMA_50'], label='SMA 50', color='red', linestyle='--', alpha=0.8)
    ax1.plot(data['SMA_200'], label='SMA 200', color='green', linestyle='--', alpha=0.8)
```

Рисунок 3.9 – Фрагмент функції відповідаючий за візуалізацію

Модуль main.py координує роботу всіх інших, виконуючи повний цикл:

– отримання даних;
– розрахунок індикаторів;
– побудова послідовностей;
– навчання моделі;
– побудова прогнозу;
– візуалізація результату.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи на тему «Розробка інформаційної системи аналізу часових рядів» було досягнуто поставлених цілей та вирішено ключові задачі. Узагальнити основні результати роботи можна в чотирьох аспектах: теоретичні основи, практична реалізація, ефективність системи, інтерфейс користувача та напрямки подальших вдосконалень.

Теоретичні основи: проведено аналіз існуючих методів аналізу, існуючих відкритих API для отримання фінансових даних та прогнозування часових рядів, зокрема статистичних підходів та методів машинного навчання. Визначено їх переваги та недоліки, що дозволило обґрунтувати вибір LSTM як основного інструменту для прогнозування.

Практична реалізація: розроблено інформаційну систему, яка інтегрує модулі завантаження даних та має таку структуру в яку легко можна додавати нові джерела отримання фінансових даних, обчислення технічних індикаторів, підготовки даних для моделі нормалізація, формування вікон та навчання LSTM-мережі. Система також включає модуль візуалізації результатів за допомогою matplotlib.

Ефективність системи: проведено тестування системи на реальних фінансових даних, що показало високу точність прогнозування порівняно з традиційними методами. Використання LSTM дозволило враховувати довгострокові залежності та адаптуватися до динаміки ринку.

Інтерфейс користувача: реалізовано консольний інтерфейс, який забезпечує зручність взаємодії з системою. Користувач може обирати джерело отримання даних, вводити тикер активу, отримувати прогнози та аналізувати результати через графіки та текстові індикатори.

Визначено напрямки подальшого вдосконалення системи, такі як:

– додавання гібридних моделей (наприклад, поєднання ARIMA та LSTM);

- додавання нових джерел фінансових даних за допомогою API;
- розширення функціоналу для роботи з різними типами даних (наприклад, соціальні медіа, економічні показники);
- впровадження вебінтерфейсу для більш зручного використання.

Робота демонструє, що розроблена інформаційна система є ефективним інструментом для аналізу та прогнозування часових рядів, особливо у фінансовій сфері. Вона може бути корисна для трейдерів, аналітиків та дослідників, які працюють з даними, що змінюються в часі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Deep Learning for Time Series Forecasting: Tutorial and Literature Survey / Benidis K. and etc. ACM Computing Surveys. 2022. Vol. 55, Iss. 6, no. 121. P. 1-36. URL: https://doi.org/10.1145/3533382 (дата звернення: 21.02.2025).

2. Dama F., Sinoquet C. Time Series Analysis and Modeling to Forecast: a Survay. 2021. URL: https://doi.org/10.48550/arXiv.2104.00164 (дата звернення: 21.02.2025).

3. Adhikari R., Agrawal R. K. An Introductory Study on Time Series Modeling and Forecasting. *LAP Lambert Academic Publishing*. 2013. URL: https://doi.org/10.48550/arXiv.1302.6613 (дата звернення: 21.02.2025).

4. Chollet F. Deep Learning with Python. Manning Publications, 2021. URL: https://github.com/fchollet/deep-learning-with-python-notebooks (дата звернення: 16.03.2025).

5. Masini R. P., Medeiros M. C., Mendes E. F. Machine Learning Advances for Time Series Forecasting. 2021. URL: https://doi.org/10.48550/arXiv.2012.12802 (дата звернення: 21.03.2025).

6. Lim B., Zohren S. Time Series Forecasting With Deep Learning: A Survey. *Royal Society*. 2021. Vol. 379, Iss. 2194. P. 1-14. URL: https://doi.org/10.1098/rsta.2020.0209 (дата звернення: 21.03.2025).

7. Yahoo Finance API Documentation. *Yahoo!Finance*. URL: https://finance.yahoo.com/ (дата звернення: 27.03.2025).

8. TA-Lib: Technical Analysis Library. URL: https://ta-lib.org/ (дата звернення: 27.03.2025).

9. An end-to-end platform for machine learning. *TensorFlow*. URL: https://www.tensorflow.org/ (дата звернення: 27.03.2025).

10. Python 3.13.4 documentation. *Python*. URL: https://docs.python.org/3/ (дата звернення: 30.03.2025).

11. Alpha Vantage Documentation. URL: https://www.alphavantage.co/ (дата звернення: 30.03.2025).

12. Financial Data that drives your success. *Twelvedata*. URL: https://twelvedata.com (дата звернення: 31.03.2025).

13. Finnhub Stock ADI. *Finnhub*. URL: https://finnhub.io (дата звернення: 31.03.2025).

14. Binance : Developer Center. *Binance*. URL: https://developers.binance.com/en (дата звернення: 31.03.2025).

15. Matplotlib 3.10.3 Documentation. *Matplotlib*. URL: https://matplotlib.org/stable/ (дата звернення: 04.04.2025).

16. Vibe Code Your Way to Data Apps. *Plotly Documentation*. URL: https://plotly.com (дата звернення: 04.04.2025).

17. Mplfinance: matplotlib finance API. *PyPI*. URL: https://pypi.org/project/mplfinance/ (дата звернення: 04.04.2025).

18. LightGBM Documentation: Parameters. *LightGBM*. URL: https://lightgbm.readthedocs.io/en/latest/Parameters.html (дата звернення: 19.04.2025).

19. Bokeh Documentation. *Bokeh*. URL: https://docs.bokeh.org/en/latest/ (дата звернення: 21.04.2025).

20. From ad-hoc plots to publication-ready figures. *Plotnine 0.15.0a2*. URL: https://plotnine.org (дата звернення: 21.04.2025).

21. XGBoost Documentation. *DMLC XGBoost*. URL: https://xgboost.readthedocs.io/en/stable/ (дата звернення: 21.04.2025).

22. Forecasting at scale. *Prophet*. URL: https://facebook.github.io/prophet/ (дата звернення: 21.04.2025).

23. PyTorch Documentation. *PyTorch*. URL: https://docs.pytorch.org/docs/stable/index.html (дата звернення: 21.04.2025).

24. Scikit-learn: Machine Learning in Python. *Scikit-learn*. URL: https://scikit-learn.org/stable/ (дата звернення: 21.04.2025).