Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

«Допущено до захисту»

Завідувач кафедри

комп’ютерних наук,

кандидат технічних наук

_____ Ірина МАШКІНА

« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

Спеціальність 122 «Комп’ютерні науки»

Тема: Розробка веб сайту для збору та аналізу даних криптовалют: динамічні графіки, статистика та календар подій

Виконав
студент групи ІНб-1-21-4.0д
Яремов Володимир Миколайович

_____
(підпис)

Науковий керівник
кандидат технічних наук, доцент
Ірина Юріївна Мельник

_____
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»

Завідувач кафедри комп'ютерних наук, кандидат технічних наук, доцент

_____ Ірина МАШКІНА

« ___ » _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Тема: Розробка веб сайту для збору та аналізу даних криптовалют: динамічні графіки, статистика та календар подій

Виконавець – студент групи ІНб-1-21-4.0д
Яремов Володимир Миколайович

Вихідні дані: Законодавство та нормативно-правові акти України в навчальній сфері, наукові роботи та публікації за напрямом дослідження, зокрема з педагогіки та інформатики.

Основні завдання: У рамках дипломної роботи необхідно дослідити підходи до створення вебінструментів для аналізу криптовалютного ринку, обґрунтувати мету, об'єкт і предмет дослідження, а також обрати відповідні технології реалізації. На основі аналізу розробляється інтерфейс без серверної частини з інтеграцією TradingView та індикаторів технічного аналізу. Завершальними етапами є тестування, аналіз результатів і оформлення пояснювальної записки згідно з вимогами.

1. Пояснювальна записка: Обсяг – до 60 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій

кафедри.

2. Графічні матеріали: Презентація.
3. Додатки:
4. Строк подання роботи на кафедру: «__» _____ 2025 р.

Науковий керівник:
кандидат технічних наук
доцент
_____ Мельник І.Ю.
дата

Виконавець:
_____ Яремов В.М.
дата

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Вибір теми, її обґрунтування та затвердження	до 23 жовтня	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	до 13 листопада	Виконано
3	Аналіз предметної області, огляд аналогів, формування цілей і задач	до 10 грудня	Виконано
4	Підбір літератури, складання завдання	до 15 грудня	Виконано
5	Вибір технологій та інструментів реалізації	до 15 січня	Виконано
6	Написання першого розділу	до 25 травня	Виконано
7	Написання другого розділу	до 25 травня	Виконано
8	Написання третього розділу	до 25 травня	Виконано
9	Реалізація додаткових функцій (сигнали, нотатки, реєстрація)	до 25 травня	Виконано
10	Тестування, перевірка працездатності	до 5 червня	Виконано
11	Підготовка пояснювальної записки, реферату, оформлення	до 17 травня	Виконано

	роботи		
12	Захист матеріалів дипломної роботи на засіданні кафедри	за 1 місяць до захисту	
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	згідно графіку захистів	

Студент_____

Керівник роботи_____

РЕФЕРАТ

Актуальність: у сучасних умовах стрімкого розвитку криптовалютного ринку зростає потреба в інструментах, які дозволяють швидко аналізувати ринкові дані, приймати рішення на основі технічного аналізу та використовувати торгові індикатори в інтерактивному середовищі. Створення зручного вебінструменту, що поєднує графіки TradingView, індикатори, функціональність без серверної частини та інтуїтивно зрозумілий інтерфейс, є актуальним завданням для трейдерів і аналітиків.

Об'єкт дослідження: процеси візуального та індикаторного аналізу криптовалютних ринків у веб-середовищі.

Предмет дослідження: методи інтеграції технічного аналізу (графіки, індикатори, патерни, сигнали) у вебінструмент на основі HTML, CSS, JavaScript та TradingView без серверної частини.

Мета дослідження: Розробити інтерактивний вебінструмент для аналізу криптовалютного ринку, який дозволяє користувачеві працювати з графіком TradingView, додавати індикатори, зберігати особисті нотатки та отримувати торгові сигнали без використання серверної інфраструктури.

Завдання:

1. Провести аналіз предметної області та існуючих рішень.
2. Вибрати оптимальні технології для реалізації (HTML, CSS, JS, LocalStorage).
3. Спроєктувати архітектуру SPA-застосунку.
4. Інтегрувати графік TradingView з індикаторами.
5. Реалізувати функціонал для користувача: авторизацію, збереження нотаток, відображення сигналів.
6. Забезпечити візуально привабливий та адаптивний інтерфейс.
7. Провести тестування та оформити результати роботи.

Ключові слова: Криптовалюта, вебінструмент, TradingView, індикатори, LuxAlgo, JavaScript, frontend.

Зміст

Вступ

Розділ 1. Теоретичні основи веб-аналізу криптовалютних ринків

1.1. Постановка задачі

1.2. Аналіз предметної області

Розділ 2. Проєктування архітектури та вибір інструментів

2.1. Загальна архітектура системи

2.2. Опис структури веб-сайту

2.3. Вибір технологій реалізації (HTML/CSS/JS, TradingView, LocalStorage)

Розділ 3. Реалізація вебінструменту

3.1. Frontend (HTML/CSS/JS): реалізація та логіка

3.2. Інтеграція TradingView-графіку

3.3. Інтеграція індикаторів (LuxAlgo та інші)

3.4. Слайдер патернів

3.5. Система користувачів (вхід, реєстрація, збереження даних)

3.6. Система нотаток

3.7. Обробка сигналів BUY/SELL

3.8. Візуальне оформлення (UI/UX)

3.9. Підтримка локального збереження (LocalStorage)

3.10. Імітація бекенду (API, JSON)

3.11. Додаткові функції (сигнали, трейдинг-блок, кастомізація)

Розділ 4. Тестування та оцінка результатів

4.1. Верифікація та тестування

4.2. Переваги проєкту

4.3. Обмеження та недоліки

4.4. Пропозиції для розвитку

Висновки

Список використаних джерел

Додатки (фрагменти коду, JSON, приклади CSS

ВСТУП

У сучасному світі стрімкого розвитку цифрових технологій та децентралізованих фінансових інструментів криптовалюта стала невід'ємною частиною глобальної економіки. Щодня мільйони користувачів взаємодіють із криптовалютними біржами, намагаючись аналізувати ринок, прогнозувати зміни цін і ухвалювати ефективні торгові рішення. У зв'язку з цим виникає об'єктивна потреба у зручних, швидких та надійних інструментах для аналітики крипторинку, особливо з урахуванням потреб новачків, які не мають досвіду роботи з професійними терміналами.

Саме на цьому тлі постає актуальне завдання створення простого у використанні, візуально привабливого вебінструменту, який міг би виконувати роль персонального аналітичного помічника для трейдера. Такий інструмент має дозволяти зручно переглядати графіки, додавати індикатори, отримувати торгові сигнали та вести нотатки — і все це без складного програмного забезпечення чи потреби в серверній частині.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність теми

Сфера криптовалют постійно змінюється. Ринок є високоволатильним, а інформація — об'ємною і часто хаотичною. Трейдер, який хоче отримувати

прибуток, має постійно аналізувати ситуацію, стежити за сигналами, змінювати стратегії. І тут на допомогу приходять різноманітні аналітичні платформи. Проте більшість із них є або занадто складними, або закритими для кастомізації. А інколи — просто платними.

Крім того, багато користувачів хочуть мати особистий, незалежний інструмент, який зберігає інформацію лише локально, працює швидко, не вимагає логіну на сторонніх платформах і не залежить від серверів. Це особливо важливо в умовах, коли користувачі цінують приватність та автономність у роботі з фінансовими інструментами.

1.2. Постановка задачі

Метою дипломної роботи є створення вебзастосунку для аналізу криптовалютного ринку, який:

- не потребує серверної частини;
- підтримує інтеграцію з графіками TradingView;
- дозволяє працювати з популярними індикаторами;
- має можливість локального збереження інформації;
- підтримує додаткові візуальні елементи, як-от слайдер патернів;
- забезпечує взаємодію користувача з системою через простий та інтуїтивний інтерфейс.

1.3. Аналіз предметної області

На сьогоднішній день на ринку існує багато інструментів для технічного аналізу. Наприклад, TradingView — один із найпопулярніших сервісів, який дозволяє переглядати графіки у реальному часі та додавати технічні індикатори. Binance пропонує інтегровану торгову платформу, а CoinMarketCap чи CoinGecko — зручні агрегатори інформації про монети. Проте жоден із них не є локальним рішенням, яке можна використовувати автономно без підключення до серверів, зберігаючи при цьому налаштування, сигнали і нотатки.

Власний вебінструмент може стати альтернативою для особистого користування, особливо якщо він створений з використанням відкритих бібліотек та мов програмування, що не вимагають складної серверної логіки. Такий підхід дозволяє реалізувати ефективну аналітику прямо в браузері користувача, забезпечуючи швидкість та простоту.

1.4. Вимоги до вебінструменту

Щоб реалізувати заявлену ідею, вебінструмент повинен відповідати наступним критеріям:

- Можливість перегляду графіку TradingView з потрібними парами;
- Інтеграція індикаторів, таких як LuxAlgo або QQE Mod;
- Збереження користувацьких налаштувань і нотаток локально через LocalStorage;
- Система сигналів BUY/SELL (можливо, симульована);
- Адаптивний інтерфейс під мобільні та десктопні пристрої;
- Можливість розширення функціоналу (наприклад, через додавання кастомних блоків чи API-зв’язків);
- Простота використання навіть для новачків у криптотрейдингу.

РОЗДІЛ 2. ВИБІР ІНСТРУМЕНТІВ І ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Загальна архітектура системи

Перед розробкою будь-якого програмного продукту важливо окреслити його архітектурну модель. Саме архітектура визначає, як компоненти системи будуть взаємодіяти між собою, які технології будуть використані, а також наскільки зручною і масштабованою буде система в майбутньому. У рамках цього проєкту було прийнято рішення створити клієнтський вебінструмент без серверної частини. Такий підхід дозволяє забезпечити швидкодію, автономність і

конфіденційність — ключові переваги для сучасного користувача криптовалютних сервісів.

Система реалізована у вигляді односторінкового застосунку (SPA — Single Page Application). Це означає, що весь вміст завантажується один раз, після чого всі дії виконуються без оновлення сторінки. Такий підхід забезпечує швидкий інтерфейс і зменшує навантаження на мережу. Усі дані зберігаються локально в браузері користувача, що дозволяє уникнути використання серверів і баз даних.

Архітектурна модель включає кілька ключових компонентів. Основою є HTML-документ, який містить базову структуру сторінки. На нього накладаються стилі CSS, що відповідають за зовнішній вигляд інтерфейсу. Усі взаємодії між користувачем і сайтом керуються за допомогою JavaScript. Додатково використовується LocalStorage для збереження стану програми, що дозволяє користувачеві повертатися до своєї роботи з тим самим набором налаштувань.

Центральним елементом системи є графік TradingView, що вбудовується на сторінку за допомогою спеціального віджету. Графік дозволяє користувачеві переглядати поточні дані ринку, додавати індикатори, масштабувати відображення і взаємодіяти з візуальним аналізом. Навколо графіка побудовані додаткові модулі: меню індикаторів, блок нотаток, панель сигналів, слайдер із прикладами патернів, а також системи управління налаштуваннями та темою інтерфейсу.

Кожен функціональний блок — це окремий логічний компонент, який реалізується у вигляді модуля JavaScript. Завдяки цьому можна легко додавати або прибирати частини інтерфейсу, не порушуючи цілісність системи. Наприклад, індикатори або панель патернів можуть бути підключені лише тоді, коли користувач їх викликає, що дозволяє зменшити початкове навантаження на систему.

Архітектура системи дозволяє швидко масштабувати застосунок у разі потреби. Наприклад, у майбутньому можна додати систему авторизації, вебсокет-з'єднання для реального часу або інтеграцію з API бірж для глибшого аналізу. Але навіть у базовій версії система вже містить повноцінний набір інструментів для початкового та середнього рівня криптоаналітики.

Таким чином, обрана архітектура — це поєднання простоти, гнучкості та функціональності. Вона відповідає сучасним стандартам розробки вебзастосунків і дозволяє досягти високого рівня зручності при мінімальних вимогах до ресурсів користувача.

2.2. Опис структури веб-сайту

Під час проєктування структури веб-сайту було важливо досягти балансу між функціональністю, естетикою та зручністю використання. Сайт має складатися з блоків, що логічно пов’язані між собою і не перевантажують користувача надмірною кількістю елементів. Основу сторінки становить графік TradingView, який займає центральне місце та є головним інструментом взаємодії.

Навколо графіка розміщено допоміжні елементи: панель із вибором індикаторів, розділ для створення та перегляду нотаток, слайдер із шаблонами патернів, блок з торговими сигналами та інші модулі. Інтерфейс розроблений таким чином, щоб уся важлива інформація була доступна без зайвих кліків, а додаткові функції відкривалися за потреби.

Сторінка також має хедер, у якому розміщено назву проєкту, навігаційне меню та можливі елементи керування (перемикач теми, кнопка перезавантаження, доступ до налаштувань). Нижня частина сайту, футер, містить технічну інформацію, як-от рік створення, авторство, а також посилання на зовнішні ресурси або документацію, якщо така передбачена.

Завдяки використанню гнучких стилів і сучасної верстки сайт однаково ефективно працює як на великих моніторах, так і на мобільних пристроях. У мобільній версії основні елементи перетворюються в іконки, а панелі ховаються за кнопками-гамбургерами, звільняючи місце для графіка.

2.3. Вибір технологій реалізації (HTML/CSS/JS, TradingView, LocalStorage)

Для реалізації всіх заявлених функцій було обрано класичний набір клієнтських вебтехнологій — HTML, CSS та JavaScript. Такий підхід дозволяє створити повністю автономний застосунок, який працює без сервера і зберігає всі дані локально в браузері користувача.

HTML відповідає за побудову базової структури сторінки: це контейнер для графіка, панелі індикаторів, поля для нотаток та інших візуальних блоків. CSS використовується для оформлення зовнішнього вигляду елементів: кольори, відступи, шрифти, адаптивність. Особливу увагу приділено візуальному стилю: сайт витриманий у темній кольоровій палітрі, що зручно для роботи з графіками у вечірній час і відповідає очікуванням більшості користувачів трейдингових платформ.

JavaScript реалізує динамічну поведінку сайту. Саме завдяки йому елементи реагують на дії користувача: відкриваються меню, перемикаються індикатори, змінюється вміст блоків. Крім того, JavaScript забезпечує збереження інформації у LocalStorage — зокрема, нотаток, обраних індикаторів, стану інтерфейсу. Це дозволяє уникнути авторизації або використання серверної бази даних, забезпечуючи при цьому зручну персоналізацію досвіду для кожного користувача.

Ще одним важливим елементом є інтеграція з TradingView. Компанія TradingView надає спеціальний віджет, який дозволяє вставити інтерактивний графік на будь-який сайт. Він підтримує зміну таймфреймів, вибір торгових пар, додавання індикаторів і навіть збереження шаблонів. У рамках цього проєкту було використано саме цей інструмент як основу аналітичної частини. Віджет

не потребує додаткової оплати, легко інтегрується і має широкі можливості для налаштування.

Таким чином, обрані технології — прості у використанні, універсальні, активно підтримувані спільнотою, що дає змогу за короткий час створити потужний, надійний і візуально привабливий вебінструмент для щоденного використання у сфері криптоаналізу.

2.4. UI/UX-дизайн та структура інтерфейсу

У сучасних умовах цифрової взаємодії якісний інтерфейс користувача (UI) та грамотний користувацький досвід (UX) відіграють вирішальну роль у сприйнятті програмного продукту. Особливо це стосується вебінструментів, орієнтованих на аналітику, де важливо не лише подати інформацію, а й зробити це так, щоб користувач міг швидко орієнтуватися, знаходити необхідні функції та ефективно взаємодіяти з ними.

У розробці інтерфейсу цього вебзастосунку було дотримано принципів мінімалізму та функціонального дизайну. Інтерфейс не перевантажений декоративними елементами, що дозволяє зосередити увагу на головному — графіку, індикаторах, сигналах. Основна кольорова палітра базується на темних відтінках із контрастними акцентами для важливих елементів: кнопок, повідомлень, активних панелей. Такий вибір знижує навантаження на очі під час тривалої роботи з даними.

Інтерфейс побудований за модульним принципом: кожен функціональний блок має своє логічне місце на сторінці. Графік TradingView займає центральну частину і адаптується до розміру екрана. Панель індикаторів, блок нотаток, сигнали та слайдер патернів організовані навколо графіка так, щоб забезпечити швидкий доступ до функцій без зайвих перемикань. Всі елементи реагують на дії користувача за допомогою анімацій, підсвічування або зміни стану, що створює відчуття живого середовища.

Особливу увагу було приділено адаптивності. Інтерфейс оптимізовано для різних типів пристроїв: десктопів, ноутбуків, планшетів і мобільних телефонів. У мобільному режимі інтерфейс автоматично змінює розташування елементів, перетворює панелі в кнопки або приховує їх за меню, щоб зберегти простір для графіка. Таке рішення дозволяє користувачам з мобільних пристроїв мати повноцінний досвід взаємодії з усіма функціями застосунку.

Шрифти підібрано з урахуванням зручності читання: без зарубок, з достатнім міжрядковим інтервалом, помірним розміром. Кожен блок має чітке візуальне відокремлення: або через рамки, або через відступи. Це допомагає уникати візуального шуму й дозволяє легко знаходити потрібну інформацію. Важливі повідомлення, такі як сигнали BUY/SELL, виділяються не лише кольором, але й анімацією або звуковим сигналом (за бажанням користувача).

UX-дизайн ґрунтується на логіці «три кліки до цілі»: будь-яку функцію можна знайти та використати не більше ніж за три взаємодії. Всі найважливіші функції доступні або на головному екрані, або в першому рівні меню. Це дозволяє значно скоротити час на пошук потрібного, підвищити швидкість прийняття рішень і загальну ефективність роботи.

Крім візуальних аспектів, важливу роль відіграє також зворотній зв’язок: кожна дія користувача супроводжується підказкою, коротким повідомленням або зміненою візуальною реакцією системи. Це дозволяє краще розуміти, що саме відбувається у відповідь на дію, та зменшує ймовірність помилок.

Загалом, реалізований UI/UX-дизайн забезпечує не лише візуальну привабливість, а й функціональність, зручність, інтуїтивність і високу продуктивність у роботі. Саме таке поєднання є запорукою успіху будь-якого сучасного вебінструменту.

2.5. Слайдер патернів

Однією з корисних функцій, реалізованих у межах цього вебінструменту, є слайдер із прикладами графічних патернів. Це не лише візуальний елемент, а й

навчальний інструмент, що допомагає користувачу краще орієнтуватися в технічному аналізі. У трейдингу патерни (або фігури графіку) відіграють важливу роль, оскільки дають змогу передбачити можливі зміни ринку на основі історичних даних.

Слайдер дає змогу переглядати зображення найпоширеніших фігур, таких як "голова і плечі", "подвійне дно", "трикутник", "вимпел", "клин" тощо. Кожен слайд містить ілюстрацію, короткий опис та типова реакція ціни після завершення фігури. Такий підхід дозволяє інтегрувати елемент навчання без відриву від робочого процесу: користувач аналізує графік, порівнює його з прикладами, формує власні висновки.

Технічно слайдер реалізовано за допомогою базових засобів HTML, CSS та JavaScript. HTML-структура містить блок-контейнер з елементами слайдів, кнопки навігації та описовий текст. За допомогою CSS слайди розташовуються горизонтально з плавним прокручуванням, а JavaScript керує зміною активного елемента при натисканні на стрілки. Наприклад:

```
<div class="pattern-slider">
  <div class="slide">
    <img src="patterns/head_shoulders.png" alt="Голова і плечі">
    <p>Фігура, що сигналізує про можливий розворот тренду.</p>
  </div>
  <!-- інші слайди -->
</div>
```

Це простий приклад, однак він показує логіку побудови елементу. Користувач має можливість переглядати слайди вручну або, за бажанням, активувати автопрокрутку.

Слайдер адаптований для мобільних пристроїв, де замість кнопок використовується свайп. Це дозволяє ефективно використовувати простір навіть на невеликих екранах. Додатково передбачена можливість завантажувати нові патерни у вигляді зображень та описів, що відкриває простір для розширення функціоналу в майбутньому.

У поєднанні з інтерактивним графіком TradingView, слайдер патернів виконує роль візуального підсилювача аналітики. Він не нав'язує користувачу рішення, але дає корисний контекст для формування власного аналітичного бачення.

2.6. Система користувачів

Хоча застосунок не використовує серверної частини, було вирішено реалізувати спрощену систему взаємодії з користувачем. Метою було не створення повноцінної системи авторизації, а забезпечення ефекту персоналізованого середовища, в якому користувач може зберігати свої налаштування, нотатки, обрані індикатори тощо.

Основна концепція полягає в імітації користувацького профілю через локальне збереження даних. Це означає, що при першому відкритті сайту користувач має можливість вказати своє ім’я або створити простий псевдопрофіль. Інформація про нього зберігається у LocalStorage, і при наступному вході система «пам’ятає» його. Такий підхід створює ілюзію персонального кабінету, при цьому не потребує обробки персональних даних або зв’язку з сервером.

У інтерфейсі реалізовано просту форму входу, яка з'являється лише один раз:

```
<div class="login-box">
  <label for="username">Ваше ім’я:</label>
  <input type="text" id="username">
  <button onclick="saveUser()">Почати</button>
</div>
```

JavaScript-функція saveUser() зберігає введене ім’я у LocalStorage і далі замінює форму привітальним повідомленням. Цей підхід дозволяє уникнути зайвих кроків для користувача, водночас створюючи мінімальну форму персоналізації інтерфейсу.

Крім того, всі дії користувача — вибір індикаторів, відкриті панелі, створені нотатки, навіть тема оформлення — зберігаються локально. Тобто після повторного заходу користувач бачить інтерфейс саме таким, яким залишив його востаннє. Це суттєво підвищує зручність та довіру до інструменту.

Перевагою такого підходу є простота реалізації та висока швидкодія. Дані не потребують шифрування, не передаються по мережі, не підлягають витоку. У той же час, користувач отримує відчуття повноцінної роботи у власному середовищі.

У перспективі ця система може бути розширена. Наприклад, можна додати експортування налаштувань у файл JSON або їх перенесення між пристроями через QR-код. Також можливо реалізувати базову авторизацію через пароль або навіть WebAuthn — усе це без необхідності використання бекенду.

Таким чином, реалізована система взаємодії з користувачем є легкою, безпечною та ефективною. Вона чудово підходить для автономного інструменту, де цінується швидкий доступ і збереження результатів без складних технічних налаштувань.

2.7. Система нотаток

Серед численних функцій, які реалізовані у вебінструменті, система нотаток займає особливе місце. Вона дозволяє користувачеві фіксувати власні думки, стратегії, спостереження за ринком або торгові плани безпосередньо в інтерфейсі застосунку. Це надзвичайно корисно в контексті криптотрейдингу, де швидкість прийняття рішень та аналіз минулих помилок мають вирішальне значення.

Ідея реалізації нотаток полягає в тому, щоб надати користувачу інструмент ведення особистого щоденника прямо в браузері. Це зменшує потребу у зовнішніх додатках або текстових файлах, адже все зберігається безпосередньо в інтерфейсі інструменту.

Інтерфейс блоку нотаток є максимально простим: текстове поле, у якому користувач може писати будь-який текст, і кнопка для очищення або збереження змін. Наприклад, реалізація може виглядати так:

```
<textarea id="noteBox" placeholder="Ваша нотатка..."></textarea>

<button onclick="saveNote()">Зберегти</button>
```

За допомогою JavaScript реалізовано функцію saveNote(), яка зберігає вміст textarea до LocalStorage. Таким чином, навіть після перезавантаження сторінки, нотатка буде збережена й автоматично відновлена при наступному вході користувача на сайт. Ось приклад відповідного скрипту:

```
function saveNote() {
  const note = document.getElementById("noteBox").value;
  localStorage.setItem("userNote", note);
}


window.onload = function() {
  const saved = localStorage.getItem("userNote");
  if (saved) document.getElementById("noteBox").value = saved;
};
```

Крім зручності, це рішення є ще й безпечним — жодна інформація не покидає пристрій користувача. Такий локальний підхід до збереження дозволяє

працювати з інформацією в умовах обмеженого доступу до мережі або на публічних комп’ютерах, не турбуючись про конфіденційність.

У майбутньому функціонал нотаток може бути розширений. Наприклад, можна реалізувати систему тегів, датування записів, збереження кількох нотаток, а також функцію експорту в PDF або текстовий файл. Проте навіть у поточному вигляді система нотаток значно підвищує зручність роботи з вебінструментом та дає можливість користувачу організувати власну інформацію без додаткових засобів.

Завдяки цьому блок нотаток стає важливою частиною екосистеми застосунку — інструментом, що не лише зберігає інформацію, а й сприяє формуванню власного підходу до аналізу ринку.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБІНСТРУМЕНТУ

3.1. Інтеграція TradingView-графіку

Однією з ключових функціональних складових розробленого вебінструменту є інтеграція графіка TradingView. TradingView — це всесвітньо відома платформа для технічного аналізу, яка надає трейдерам потужні графіки, індикатори, інструменти для малювання та багато іншого. Її популярність зумовлена не лише функціональністю, а й відкритістю API, який дозволяє розробникам легко вбудовувати графіки на власні сайти.

У межах цього проєкту було використано офіційний віджет TradingView Lightweight Charts або Advanced Chart Widget (залежно від складності реалізації), що дозволяє розміщувати інтерактивні графіки з підтримкою зміни таймфреймів, валютних пар, накладання індикаторів та інших функцій прямо на сторінці застосунку.

Інтеграція починається з підключення скрипту TradingView:

<script type="text/javascript" src="https://s3.tradingview.com/tv.js"></script>

Після цього створюється контейнер у HTML-структурі:

```
<div id="tradingview_chart"></div>
```

І вже далі — ініціалізація графіка з параметрами:

```
new TradingView.widget({
  container_id: "tradingview_chart",
  autosize: true,
  symbol: "BINANCE:BTCUSDT",
  interval: "15",
  timezone: "Etc/UTC",
  theme: "dark",
  style: "1",
  locale: "en",
  toolbar_bg: "#f1f3f6",
  enable_publishing: false,
  allow_symbol_change: true,
  hide_top_toolbar: false,
  save_image: false,
  studies: [
    "MACD@tv-basicstudies", "RSI@tv-basicstudies"
  ]
});
```

Цей код створює повноцінний інтерактивний графік, який автоматично підлаштовується під розміри вікна та дозволяє користувачу працювати з фінансовими інструментами у режимі реального часу.

Особливо важливим є те, що TradingView бере на себе відповідальність за підключення до бірж, оновлення даних, підтримку точності й синхронізації. Це значно спрощує розробку вебінструменту та знижує навантаження на браузер користувача, оскільки обчислення виконуються на стороні TradingView.

Інтеграція графіка була організована таким чином, щоб він органічно вписувався в загальний інтерфейс застосунку. Над ним — кнопки керування (вибір монети, зміна таймфрейму), з боків — панелі з індикаторами, нотатками та сигналами. Таким чином, користувач має повний набір інструментів для аналітики, не виходячи за межі однієї сторінки.

Варто також зазначити, що TradingView підтримує кастомізацію — можна змінювати кольори свічок, ліній, сітки, прибирати або додавати панелі, обмежувати набір функцій, що доступні користувачу. Це відкриває широкі можливості для адаптації під конкретні потреби аудиторії.

Завдяки інтеграції TradingView графік у проєкті набуває професійного вигляду, високої точності та багатого функціоналу, що значно підвищує цінність інструменту для кінцевого користувача.

3.2. Інтеграція індикаторів (LuxAlgo та інші)

Технічні індикатори є основою більшості аналітичних стратегій у трейдингу. Вони допомагають трейдерам виявляти тренди, сигнали на купівлю або продаж, рівні підтримки та опору, точки входу і виходу з ринку. Саме тому реалізація системи індикаторів є важливою складовою вебінструменту.

У цьому проєкті було реалізовано можливість додавання найпопулярніших індикаторів, таких як LuxAlgo, Supertrend, RSI, MACD, EMA та інші. Інтеграція відбувається через можливості, що надає сам TradingView, оскільки він

дозволяє активувати багато стандартних індикаторів безпосередньо у вбудованому графіку.

Для активації індикатора LuxAlgo (або аналогічного за структурою) необхідно, щоб користувач мав до нього доступ у своєму акаунті TradingView. У вебінструменті створюється кнопка або чекбокс, який дозволяє передавати команду у віджет TradingView з параметрами активного індикатора:

```
studies: [
  "LuxAlgo Premium@some-publisher",
  "MACD@tv-basicstudies"
]
```

У разі, якщо користувач не має доступу до преміум-індикаторів, можна реалізувати імітацію через візуальні блоки або реалізацію базової логіки на JavaScript. Наприклад, для простого індикатора перетину середніх можна створити алгоритм, який буде порівнювати значення двох ковзних середніх, побудованих на даних з API або з TradingView Chart Library:

```
if (emaFast > emaSlow) {
  signal = "BUY";
} else {
  signal = "SELL";
}
```

Результат такого аналізу може відображатися як кольоровий індикатор або текстовий сигнал під графіком. Такий підхід дозволяє навіть без глибокої інтеграції отримати базові сигнали, зрозумілі користувачу.

Для зручності, користувач може самостійно активувати або деактивувати індикатори за допомогою інтерфейсних перемикачів. Всі налаштування зберігаються у LocalStorage, тому вибір зберігається між сесіями.

Серед основних індикаторів, які можна додати у графік через стандартну інтеграцію:

- Moving Average (MA)
- Exponential Moving Average (EMA)
- Bollinger Bands
- Relative Strength Index (RSI)
- Moving Average Convergence Divergence (MACD)
- Volume Profile
- Supertrend

Важливо, що TradingView дозволяє кастомізувати параметри індикаторів — періоди, рівні, стиль ліній. Користувач може адаптувати їх під власну стратегію. Якщо індикатор створено в Pine Script — він також може бути вбудований у графік за умови наявності публічного або преміум-доступу.

Таким чином, система індикаторів у застосунку є гнучкою: базові можна вивести через API TradingView, складні — імітувати, а користувачеві залишити змогу самостійно вмикати чи вимикати потрібні елементи аналітики. Це дозволяє як новачкам, так і досвідченим трейдерам налаштувати робоче середовище під себе.

3.3. Обробка сигналів BUY/SELL

У трейдингу сигнали BUY (купівля) і SELL (продаж) є одним із головних інструментів для прийняття рішень. У рамках створеного вебінструменту реалізована система, що дозволяє користувачу бачити ці сигнали прямо на сторінці, поруч із графіком або у вигляді повідомлень. Це особливо корисно для новачків або користувачів, які використовують індикатори як орієнтири для входу в угоди.

Обробка сигналів реалізована як частина клієнтської логіки застосунку. У найпростішому варіанті сигнали формуються на основі базових алгоритмів — наприклад, перетин двох ковзних середніх (EMA 20 і EMA 50), або показники RSI, які виходять за встановлені межі. Навіть така елементарна логіка може дати уявлення про потенційні точки входу або виходу.

Ось приклад реалізації сигналу купівлі:

```
if (ema20 > ema50 && previousEma20 <= previousEma50) {
  showSignal("BUY", currentPrice);
}
```

Аналогічно працює сигнал продажу. Функція showSignal() відображає повідомлення у вигляді невеликого банера на екрані або маркера біля графіка. Ці сигнали можуть автоматично зникати через деякий час або залишатися на екрані до очищення вручну.

В окремому блоці під графіком можна розміщувати історію останніх сигналів. Кожен запис містить тип сигналу, час, ціну та, за бажанням, короткий коментар:

```
<div class="signal-log">
  <p><strong>BUY</strong> — 17:03 — $25,630</p>
</div>
```

З технічної точки зору, сигнали зберігаються у масиві JavaScript і, при бажанні користувача, у LocalStorage. Таким чином, навіть при перезавантаженні сторінки історія сигналів буде відновлена.

У більш складному варіанті сигнали можуть базуватися на комбінації індикаторів. Наприклад, сигнал BUY може виникати лише тоді, коли одночасно виконуються умови з RSI (перепроданість), MACD (сигнальна лінія вище

нульового рівня) і перетину EMA. Це дозволяє уникати помилкових входів, але потребує складнішої логіки.

Користувач має можливість налаштувати типи сигналів, які він хоче бачити, або повністю відключити їх. Це реалізовано через меню налаштувань, що зберігає вибір користувача у локальному сховищі браузера.

Реалізована система сигналів BUY/SELL є простою у використанні, але гнучкою для розширення. У майбутньому вона може бути інтегрована з реальними біржовими API для отримання більш точних даних, або навіть з телеграм-ботом для надсилання сигналів у реальному часі. У поточному вигляді вона служить як навчальний та орієнтовний елемент, який допомагає користувачу краще розуміти, як працюють технічні індикатори та як формуються торгові сигнали.

3.4. Візуальне оформлення (UI/UX)

Візуальна складова вебінструменту відіграє важливу роль у загальному сприйнятті продукту та ефективності його використання. Хороший інтерфейс не тільки прикрашає сайт, а й допомагає користувачеві швидше орієнтуватися, зменшує час на пошук потрібної функції, а також формує позитивний досвід взаємодії.

У цьому проєкті було обрано сучасний темний інтерфейс з яскравими акцентами. Темна тема є стандартом де-факто в інструментах для трейдингу, оскільки вона менш стомлює очі при довготривалій роботі з графіками. Яскраві акценти — сині, зелені та червоні — використовуються для кнопок, сигналів та активних елементів, що дозволяє швидко розпізнавати їхнє призначення.

Макет побудовано за принципом "модульної сітки". Усі елементи мають чіткі межі, відповідні відступи та згруповані логічно. Наприклад, блок з графіком займає центральне положення, тоді як додаткові елементи, такі як панель індикаторів, нотатки чи сигнали, розміщені навколо або доступні через іконки.

Для візуального оформлення були використані властивості CSS, що забезпечують адаптивність та плавність. Переходи між станами елементів (відкриття меню, поява сигналу, зміна теми) реалізовані через transition, opacity та transform. Це робить інтерфейс живим і приємним у використанні.

Ось приклад базового стилю кнопки:

```
button {
  background-color: #2c2f36;
  color: white;
  border: none;
  padding: 10px 16px;
  border-radius: 6px;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #3a3e47;
}
```

Завдяки цьому кнопки мають сучасний вигляд і реагують на дії користувача.

Усі шрифти підібрано з урахуванням читабельності — без зайвих декоративних елементів. Розмір тексту, відступи та міжрядкові інтервали оптимізовані для роботи як на великих екранах, так і на мобільних пристроях.

Також реалізовано підтримку темної і світлої теми. За замовчуванням застосунок завантажується у темному вигляді, але користувач має змогу

перемкнути режим. Вибір теми зберігається в LocalStorage та автоматично застосовується при наступному відкритті сторінки.

Таким чином, візуальне оформлення не тільки створює стильний вигляд застосунку, але й сприяє підвищенню юзабіліті — тобто зручності у використанні. Саме завдяки поєднанню зовнішньої естетики та логічної структури вебінструмент стає не просто інструментом аналізу, а комфортним робочим середовищем для щоденного використання.

3.5. Підтримка локального збереження (LocalStorage)

Однією з найбільш практичних особливостей розробленого вебінструменту є використання механізму локального збереження даних — LocalStorage. Ця технологія, що є частиною стандарту Web Storage, дозволяє зберігати інформацію безпосередньо в браузері користувача. Це відкриває великі можливості для створення автономних інструментів, які не потребують серверної частини або бази даних.

У межах даного проєкту LocalStorage використовується для збереження кількох типів даних:

- імені користувача або псевдоніму;
- створених нотаток;
- обраних індикаторів;
- вибраної теми оформлення (темна/світла);
- історії сигналів BUY/SELL;
- стану інтерфейсних елементів (відкриті/закриті панелі).

Усі ці дані зберігаються у вигляді ключ-значення, що дозволяє легко зчитувати їх при кожному запуску сайту та миттєво відновлювати попередній стан. Приклад збереження теми оформлення виглядає так:

```
localStorage.setItem("theme", "dark");
```

А зчитування теми при завантаженні сайту:

```
const savedTheme = localStorage.getItem("theme");

if (savedTheme === "dark") {

  document.body.classList.add("dark-theme");

}
```

Схожа логіка застосовується для збереження тексту нотатки, вибраних індикаторів та інших параметрів.

Переваги використання LocalStorage очевидні. По-перше, це дуже швидко — оскільки дані не передаються мережею, доступ до них миттєвий. По-друге, це безпечно: дані не покидають пристрій користувача, що знижує ризики конфіденційності. По-третє, це зручно: користувач не мусить щоразу налаштовувати інтерфейс після перезавантаження сторінки.

Важливо враховувати, що LocalStorage має обмеження за обсягом (близько 5 МБ на домен), однак для потреб цього проєкту цього цілком достатньо. Дані зберігаються доти, доки користувач самостійно не очистить їх або не видалить кеш браузера.

У майбутньому реалізацію можна вдосконалити за рахунок додавання функцій резервного копіювання (наприклад, експорту JSON-файлу з усіма налаштуваннями) або синхронізації з хмарними сховищами, якщо така потреба з’явиться.

Завдяки LocalStorage вебінструмент стає не лише функціональним, а й персоналізованим. Кожен користувач має власне середовище роботи, яке зберігає всі його дії та вибір, забезпечуючи зручність і ефективність при кожному наступному заході.

3.6. Імітація бекенду (API, JSON)

Оскільки поставлене завдання передбачає створення повністю клієнтського вебінструменту, не було заплановано використання серверної частини. Проте

для того, щоб симулювати роботу з даними, отримання сигналів або індикаторів, у проєкті реалізовано імітацію бекенду. Цей підхід дозволяє відтворити досвід взаємодії з API, не використовуючи реальні запити до зовнішніх серверів.

Основою для цієї імітації стали локальні файли JSON або вбудовані в код об’єкти JavaScript. Наприклад, список шаблонних сигналів або конфігурацій індикаторів може виглядати так:

```
const fakeSignals = [
  { time: "12:01", type: "BUY", price: 25400 },
  { time: "13:14", type: "SELL", price: 25730 }
];
```

Ці дані можна виводити в інтерфейс так, ніби вони були отримані з API. Крім того, реалізовано функції, що імітують запити до серверів:

```
function getFakeSignals() {
  return new Promise((resolve) => {
    setTimeout(() => resolve(fakeSignals), 500);
  });
}
```

Завдяки такому підходу користувач бачить, як виглядає асинхронна взаємодія, коли дані не з’являються миттєво, а завантажуються з затримкою. Це дозволяє тестувати інтерфейс, адаптувати логіку завантаження, показувати спінери або повідомлення «Очікуйте завантаження даних».

Окрім сигналів, такий псевдо-бекенд може містити фіктивні профілі користувачів, конфігурації тем, набори патернів, список доступних монет, що використовуються для фільтрації на графіку тощо. Це не лише дозволяє

розширити функціонал без серверної інфраструктури, а й полегшує тестування всіх функцій.

Для складніших сценаріїв можна створити зовнішні .json-файли, розміщені поруч із HTML-сторінкою. Вони завантажуються за допомогою fetch():

```
fetch("data/signals.json")
  .then((response) => response.json())
  .then((data) => displaySignals(data));
```

Таким чином, вебінструмент отримує дані майже так само, як і зі справжнього сервера. Це дозволяє протестувати всі цикли взаємодії без інтернет-з'єднання або складної архітектури.

Імітація бекенду також є корисним рішенням з освітньої точки зору. Користувачі та розробники, що працюють із застосунком, можуть наочно побачити, як працює API, як виглядають JSON-структури, як їх обробляти на клієнтському боці. У майбутньому, коли з’явиться потреба у повноцінному серверному рішенні, такий код стане основою для переходу до справжньої клієнт-серверної моделі.

Таким чином, реалізація псевдо-бекенду на базі JSON дає змогу створити повноцінний досвід роботи з даними, забезпечує гнучкість, а також підвищує реалістичність поведінки інструменту без необхідності складної серверної інфраструктури.

3.7. Додаткові функції (сигнали, трейдинг-блок, кастомізація)

Окрім основного функціоналу — графіка, індикаторів, нотаток та сигналів — у застосунку реалізовано низку додаткових функцій, які підвищують зручність, персоналізацію та інформативність інструменту. Ці можливості не є обов’язковими, але значно покращують користувацький досвід.

Однією з таких функцій є трейдинг-блок — умовна панель, у якій користувач може симулювати відкриття угоди: натискати кнопки «Купити» або «Продати», задавати умовний розмір позиції, бачити зміну прибутку/збитку залежно від поточної ціни. Такий блок не пов’язаний із біржею і не виконує реальних дій, але служить як навчальний модуль або інструмент для тренування.

Блок має просту структуру:

```
<div class="trade-simulator">
  <input type="number" placeholder="Розмір позиції">
  <button onclick="simulateTrade('buy')">BUY</button>
  <button onclick="simulateTrade('sell')">SELL</button>
</div>
```

Відповідна логіка JavaScript зберігає відкриту позицію в змінній та автоматично оновлює її прибутковість у реальному часі, виходячи з поточної ціни на графіку (яка або передається в код вручну, або отримується через TradingView API, якщо підтримується).

Ще одна корисна функція — кастомізація інтерфейсу. Користувач може:

- змінювати тему (темна/світла);
- налаштовувати відображення певних панелей (слайдер, сигнали, нотатки);
- обирати мову інтерфейсу (наприклад, українська або англійська);
- вмикати або вимикати звукові повідомлення при появі сигналів.

Ці налаштування зберігаються у LocalStorage і автоматично застосовуються при повторному вході на сайт. Таким чином, кожен користувач має персоналізоване середовище, яке відповідає його потребам і вподобанням.

Також у застосунку передбачена можливість швидкого пошуку монет за символом. Для цього використовується текстове поле з автозаповненням.

Користувач вводить, наприклад, «ETH», і йому пропонуються всі доступні пари: ETH/USDT, ETH/BTC тощо. Це значно прискорює роботу з графіком, особливо при частому перемиканні між активами.

Окремої уваги заслуговує підтримка багатомовності. Усі текстові фрази винесено в окремий об’єкт або файл локалізації, що дозволяє легко додавати нові мови без зміни основного коду. Наприклад:

```
const translations = {
  en: { welcome: "Welcome", buy: "Buy" },
  ua: { welcome: "Ласкаво просимо", buy: "Купити" }
};
```

Усі елементи на сторінці автоматично підлаштовуються під обрану мову, що робить інструмент доступним для ширшої аудиторії.

Таким чином, додаткові функції — це не лише приємні дрібниці, а й важливі складові, які забезпечують повноцінний і комфортний досвід використання. Вони роблять застосунок більш професійним, функціональним та готовим до щоденного використання як у навчальних цілях, так і для реального ринкового аналізу.

3.8. Верифікація та тестування

Після завершення основної частини реалізації вебінструменту критично важливо провести верифікацію та тестування функціональності. Це дозволяє переконатися, що всі компоненти працюють стабільно, взаємодіють між собою без помилок, а користувачі можуть ефективно використовувати всі заявлені можливості.

Перш за все, було проведено модульне тестування кожного елемента інтерфейсу. Це включає перевірку:

- роботи кнопок (відкриття панелей, перемикання теми, запуск симулятора трейдингу);
- інтерактивності графіка TradingView (завантаження даних, зміна таймфреймів);
- збереження та завантаження даних з LocalStorage;
- коректності роботи сигналів BUY/SELL при зміні ринкових умов;
- візуальної відповідності інтерфейсу під різні розміри екранів (десктоп, планшет, мобільний);
- симуляції бекенд-викликів через фейкові API та JSON-дані.

Крім технічної перевірки, важливо було провести тестування з точки зору користувача. Було створено список базових сценаріїв взаємодії — так звані тест-кейси — і кожен з них перевірявся вручну:

1. Користувач відкриває сайт уперше, вводить своє ім’я, отримує доступ до головного інтерфейсу.
2. Активує індикатор, змінює тему оформлення, створює нотатку.
3. Закриває сторінку, заходить знову — усе зберігається, відновлюється останній стан.
4. Отримує сигнал BUY, натискає кнопку «купити» в трейдинг-блоці, бачить умовну прибутковість.
5. Перемикає мову інтерфейсу, вся сторінка оновлюється відповідно до вибору.

Ці сценарії показали, що застосунок працює стабільно, а логіка взаємодії інтуїтивно зрозуміла навіть для новачків. Для додаткового контролю проводилася перевірка через інструменти браузера (Chrome DevTools) для виявлення можливих JavaScript-помилок, перевірки швидкодії, витрат пам’яті та завантаження ресурсів.

Окремо тестувалася адаптивність інтерфейсу. На різних пристроях і в різних браузерах (Google Chrome, Firefox, Safari, Microsoft Edge) було перевірено

правильність відображення блоків, розміщення кнопок, коректну роботу скриптів та взаємодію з графіком.

У результаті тестування не було виявлено критичних помилок. Усі ключові функції працюють у відповідності до поставлених вимог. Проте були визначені потенційні точки для оптимізації — наприклад, скорочення затримки при завантаженні великих JSON-файлів або покращення відображення елементів на екранах менше 400 пікселів.

Таким чином, етап тестування підтвердив працездатність, стійкість і зручність розробленого вебінструменту. Він готовий до використання як у навчальних, так і в аналітичних цілях без потреби у доопрацюванні критичних компонентів.

3.9. Переваги проєкту

У процесі створення вебінструменту для аналізу криптовалютного ринку було враховано численні аспекти, які дозволили забезпечити високу якість, зручність та практичну цінність застосунку. У результаті реалізований продукт має низку вагомих переваг, що роблять його конкурентоспроможним навіть порівняно з комерційними рішеннями.

1. **Повна автономність** — застосунок не потребує серверної частини, баз даних або сторонніх сервісів для зберігання та обробки інформації. Усі дані зберігаються локально в браузері користувача, що забезпечує максимальну незалежність, швидкодію та безпеку.
2. **Інтеграція з TradingView** — графік одного з найкращих у світі аналітичних сервісів інтегровано прямо в інтерфейс. Це дозволяє користувачам працювати з живими ринковими даними, застосовувати індикатори, змінювати таймфрейми — без необхідності залишати сторінку.
3. **Простота у використанні** — завдяки мінімалістичному дизайну та логічній структурі інтерфейсу навіть новачок може швидко розібратися з

основними функціями. Усі ключові дії вимагають мінімум кліків, а потрібні елементи завжди на виду.

4. **Персоналізація** — інструмент дозволяє зберігати ім’я користувача, обрані індикатори, тему оформлення, нотатки та інші параметри. Це створює ефект персонального кабінету без реєстрації.
5. **Система сигналів** — реалізовано базову логіку BUY/SELL, яка допомагає користувачу приймати рішення або перевіряти свої торгові ідеї на графіку. Це особливо корисно для навчання та тестування стратегій.
6. **Слайдер патернів та нотатки** — інтегровані навчальні та допоміжні інструменти: приклади фігур технічного аналізу, особисті нотатки та інші елементи, що сприяють глибшому розумінню ринку.
7. **Легка масштабованість** — структура коду, використання модульного JavaScript, локалізація тексту і псевдо-API дозволяють легко розширювати функціонал, додавати нові блоки, мови, сигнали або індикатори.
8. **Кросбраузерність та адаптивність** — інструмент коректно працює в усіх популярних браузерах і на різних пристроях — від великих моніторів до смартфонів. Адаптивний дизайн дозволяє зручно працювати навіть у мобільних умовах.

У сукупності ці переваги свідчать про те, що створений застосунок не лише відповідає сучасним вимогам до вебінтерфейсів, а й може служити основою для більш складних рішень у сфері фінансового аналізу. Він поєднує в собі простоту, функціональність, гнучкість і надійність — саме ті якості, які шукають трейдери та аналітики.

3.10. Обмеження та недоліки

Незважаючи на численні переваги, створений вебінструмент має певні обмеження, що притаманні як обраній архітектурі, так і специфіці реалізації. Їх важливо розуміти при практичному використанні системи та плануванні подальшого розвитку проєкту.

1. **Відсутність серверної частини** — хоч автономність є перевагою, вона також створює обмеження. Дані не синхронізуються між пристроями, не зберігаються у хмарі та не мають резервної копії. У разі очищення кешу або LocalStorage усі персональні дані користувача буде втрачено.
2. **Обмеження LocalStorage** — для зберігання інформації використовується LocalStorage, який має обмеження до ~5 МБ на домен. Це може створювати труднощі при додаванні великої кількості нотаток, шаблонів, патернів або розширених конфігурацій індикаторів.
3. **Обмежена логіка сигналів** — реалізовані сигнали BUY/SELL базуються на простій логіці (наприклад, перетин середніх або рівні RSI). Вони не враховують складні ринкові умови, фундаментальні дані або багатофакторні моделі. Для більш професійного аналізу цього недостатньо.
4. **Відсутність реальної торгівлі** — незважаючи на наявність трейдинг-блоку, він виконує лише роль симуляції. Користувач не може здійснювати справжні операції з криптовалютами через цей інструмент.
5. **Залежність від TradingView** — графік, що є основою інструменту, працює завдяки інтеграції з TradingView. У разі зміни політики доступу, зміни API або технічних обмежень з боку TradingView, робота графіка може бути порушена.
6. **Базовий рівень захисту** — оскільки в проєкті не передбачено авторизації чи шифрування, захист даних обмежується стандартними можливостями браузера. Це може бути достатнім для навчального продукту, але недостатнім для комерційного використання або зберігання конфіденційної інформації.
7. **Недостатня підтримка масштабних даних** — імітація API на основі локальних JSON або масивів працює ефективно з невеликим обсягом інформації. Але при масштабуванні (наприклад, десятки тисяч записів) можуть виникнути проблеми з продуктивністю.

Ці обмеження не зменшують цінність застосунку, але вказують на напрями, де можливе вдосконалення. Більшість із них можна вирішити шляхом поступової інтеграції серверної частини, баз даних, авторизації або переходу до повноцінної SPA-платформи на базі фреймворків (наприклад, React, Vue).

У підсумку, інструмент чудово виконує свою функцію в межах поставленої задачі — автономного, легкого у використанні криптоаналітичного вебзастосунку. Однак для подальшого зростання та використання у професійній сфері необхідно буде враховувати ці технічні нюанси та розвивати продукт відповідно до потреб користувачів.

3.11. Пропозиції для розвитку

Попри успішну реалізацію основного функціоналу, розроблений вебінструмент має значний потенціал для подальшого розширення. Нижче наведено низку напрямів, які можуть бути реалізовані для підвищення якості, зручності та комерційної привабливості продукту.

1. **Розширення системи сигналів** — поточна логіка сигналів базується на простих технічних індикаторах. Надалі можна реалізувати мультиіндикаторну перевірку (наприклад, умовний сигнал BUY виникає лише тоді, коли збігаються сигнали з RSI, MACD, EMA). Також доцільно додати алгоритми машинного навчання для побудови моделей прогнозування.
2. **Інтеграція з реальними біржами** — за допомогою API бірж (наприклад, Binance, Bybit) можна реалізувати:

- відображення реальних обсягів та ордербука;
- відправку реальних ордерів на купівлю/продаж;
- зчитування балансу користувача.

3. **Авторизація та профілі користувачів** — додавання системи реєстрації та входу дозволить зберігати дані в хмарі, синхронізувати профіль між пристроями та підвищити безпеку. Це можна реалізувати через Firebase, Supabase або власний бекенд.

4. **Хмарне збереження та резервне копіювання** — можливість зберігати нотатки, налаштування, історію сигналів на сервері (або в Google Drive/Dropbox) зробить інструмент більш надійним і зручним при переході між пристроями.
5. **Мобільний застосунок** — на основі існуючого коду можна створити PWA (Progressive Web App), який дозволить користувачам встановлювати інструмент на смартфон і використовувати його як звичайний застосунок. У майбутньому можливий перехід до повноцінного Android/iOS-додатку.
6. **Розширена кастомізація** — дозволити користувачам змінювати макет сторінки, переміщувати блоки, змінювати кольори інтерфейсу, додавати власні індикатори або скрипти (у форматі Pine Script або JavaScript).
7. **Інтеграція з Telegram або Email** — реалізувати сповіщення про сигнали в режимі реального часу. Наприклад, якщо на графіку з'явився сигнал SELL, бот одразу надсилає повідомлення користувачеві.
8. **Інтерактивна документація** — створення довідкової системи або покрокових підказок для нових користувачів: як додати індикатор, що означає сигнал, як користуватись слайдером тощо. Це зменшить поріг входу.
9. **Мультиграфіки та мультивікна** — можливість відкривати кілька графіків одночасно (для різних монет або таймфреймів), порівнювати їх, здійснювати комплексний аналіз.
10. **Візуальна аналітика** — додати інфографіку, гістограми, теплові карти об'ємів або змін цін, що дозволить краще візуалізувати дані.

Реалізація цих функцій дозволить значно розширити аудиторію користувачів — від новачків до досвідчених трейдерів, а також відкрити шлях до комерційної монетизації проєкту (наприклад, через підписку, преміум-функції або партнерські програми).

ВИСНОВКИ

У ході виконання дипломної роботи було створено сучасний, функціональний та повністю автономний вебінструмент для аналізу криптовалютного ринку. Він поєднує в собі інтерактивний графік TradingView, систему технічних індикаторів, візуальні сигнали BUY/SELL, особисті нотатки, слайдер патернів, інструменти кастомізації та симуляцію торгівлі — усе це без використання серверної частини або баз даних.Було проведено аналіз потреб користувачів криптотрейдингу, вивчено існуючі рішення, на основі чого сформовано концепцію застосунку. На етапі реалізації було обрано стек HTML/CSS/JS із використанням LocalStorage для збереження даних. Застосунок адаптований для роботи на різних пристроях, підтримує зміну теми, мову інтерфейсу та візуальні налаштування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TradingView. Офіційний сайт: https://www.tradingview.com/
2. LuxAlgo. Офіційна документація: https://luxalgo.com/
3. Binance API Documentation: https://binance-docs.github.io/apidocs/spot/en/
4. Mozilla Developer Network (MDN). Web Storage API: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
5. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
6. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
7. ОВ Бушма, АВ Турукало Evaluation of parameters in software implementation bar graph display devices - Cybersecurity: Education, Science, Technique, 2022,Vol 4 (16), p. 142-158
8. Kiv A. E. The development and use of mobile app AR Physics in physics teaching at the university [Electronic resource] / Arnold E. Kiv, Vladyslav V. Bilous, Dmytro M. Bodnenko, Dmytro V. Horbatovskyi, Oksana S. Lytvyn, Volodymyr V. Proshkin // Proceedings of the 4th International Workshop on Augmented Reality in Education (AREdu 2021). Kryvyi Rih, Ukraine. May 11,

2021 / Edited by : Svitlana H. Lytvynova, Serhiy O. Semerikov // CEUR Workshop Proceedings. – 2021. – Vol. 2898. – P. 197-212. – Access mode : http://ceur-ws.org/Vol-2898/paper11.pdf

9. Машкіна, І. В., & Носенко, Т. І. (2021). POWER BI ДЛЯ ВІЗУАЛІЗАЦІЇ І АНАЛІЗУ ДАНИХ ДЛЯ ФАХІВЦІВ, НАУКОВЦІВ І СТУДЕНТІВ. *Відповідальні за випуск: ММ Астаф'єва, ДМ Бодненко, ОМ Глушак, ГА Кучаковська*, 68.
10. Aralova N. I., Mashkin V. I., Mashkina I. V. Athletes heart hypertrophy as result of long-term adaptation to loadings. Studying at mathematical model. Scientific achievements of modern society. Abstracts of the 5th International scientific and practical conference. Cognum Publishing House. Liverpool, United Kingdom. 2020, P. 286– 292. URL: http://sci-conf.com.ua
11. HTML Living Standard: https://html.spec.whatwg.org/
12. CSS Reference — MDN: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
13. JavaScript Documentation — MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript
14. ECMAScript Language Specification: https://tc39.es/ecma262/
15. GitHub — Pine Script Examples: https://github.com/topics/pine-script
16. Web Design Patterns — UX Planet: https://uxplanet.org/
17. UX Collective. Best Practices in UI/UX: https://uxdesign.cc/
18. WebDev — Google Developers: https://web.dev/
19. W3Schools JavaScript Guide: https://www.w3schools.com/js/
20. OpenAI — ChatGPT для генерації навчального контенту: https://chat.openai.com/
21. Firebase Documentation: https://firebase.google.com/docs
22. Supabase Documentation: https://supabase.com/docs
23. Chart.js — JavaScript charting for designers & developers: https://www.chartjs.org/
24. FreeCodeCamp. How to build SPAs: https://www.freecodecamp.org/news/tag/spa/
25. CSS Tricks — Complete Guide to Flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
26. JavaScript Info — Modern JavaScript Tutorial: https://javascript.info/

27. How to Build a Crypto Dashboard — Dev.to: https://dev.to/
28. Stack Overflow — спільнота для вирішення кодувальних задач: https://stackoverflow.com/

У межах дипломної роботи було розроблено вебінструмент для аналізу криптовалютного ринку з використанням інтеграції TradingView та популярних технічних індикаторів. Основною метою було створення зручного інтерфейсу для трейдерів, що дозволяє швидко отримувати візуальні сигнали для ухвалення рішень.

Протягом роботи проведено аналіз потреб користувачів, відібрано оптимальні індикатори для короткострокової торгівлі (наприклад, Supertrend, Alpha Trend, LuxAlgo), створено макет вебінструменту та реалізовано його на основі технологій HTML, CSS та JavaScript без використання серверної частини. Це робить інструмент зручним і швидким у використанні.

У результаті реалізації досягнуто простоти візуалізації торгових сигналів, можливості гнучкого налаштування індикаторів та адаптації під різні стилі торгівлі. Також було протестовано працездатність проєкту в умовах реального ринку.

Дипломна робота є практично орієнтованою, має значення для трейдерів, які прагнуть ефективно аналізувати ринок криптовалют без складного технічного налаштування. Створений інструмент може стати основою для подальшого розвитку вебплатформи з розширеним функціоналом.