Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Назаркіна Степана Анатолійовича*
(ПІБ)

академічної групи *121-19-2*
(шифр)

напряму підготовки *121 Інженерія програмного забезпечення*
(код і назва напряму підготовки)

на тему: *Розробка веб-трекера ринку криптовалют*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Приходченко С.Д*			
розділів:				
спеціальний	*доц. Приходченко С.Д*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*ст. викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-2* (група) *Назаркін С.А.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-трекера ринку криптовалют*

затверджена наказом ректора НТУ «ДП» від *16 . 05 .2023 р.* № *350 -с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц.Приходченко С.Д.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Назаркін С.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: __

РЕФЕРАТ

Пояснювальна записка: 87 с., 21 рис., 3 дод., 24 джерела.

Об'єкт розробки: Веб-додаток для відстеження ринку криптовалют.

Метою даної дипломної роботи є розробка веб-трекера криптовалютного ринку з використанням React.js, Firebase та Visual Studio Code. Додаток має на меті надати користувачам інтуїтивно зрозумілий інтерфейс та широкі можливості для задоволення їхніх потреб у відстеженні криптовалют.

У вступі даної дипломної роботи проаналізовано та розглянуто поточний стан ринку криптовалют, визначено конкретну мету дослідження, окреслено сферу застосування проекту та надано обґрунтування актуальності обраної теми. Також представлено постановку задачі, в якій висвітлено цілі та завдання процесу розробки, а також вказано вимоги до програмного забезпечення, технології та інструменти, які будуть використовуватися.

У першому розділі дипломної роботи проведено детальний аналіз ринку криптовалют, підкреслено важливість та актуальність розробки веб-трекера. У розділі визначено специфічні вимоги до функціональності трекера.

У другому розділі обрано відповідні платформи для розробки та описано процес проектування та розробки. Також описано алгоритм і структуру функціоналу застосунку. Крім того, в розділі висвітлено процес виклику та завантаження додатку, вказано вхідні та вихідні дані, а також надано інформацію про апаратні вимоги для запуску трекера.

В економічному розділі дипломної роботи визначено трудомісткість розробки веб-трекера. Вона включає в себе розрахунок вартості робіт з розробки та оцінку часу, необхідного для їх виконання.

Практичне значення проекту полягає в розробці веб-трекера криптовалютного ринку зі зручним інтерфейсом. Додаток має на меті надати користувачам безпечну платформу для відстеження та моніторингу цін на криптовалюти, ринкових тенденцій та ефективності портфелів. Використовуючи такі технології, як React.js та Firebase, трекер пропонує оновлення даних у режимі реального часу, персоналізовані списки спостереження та автентифікацію користувачів для безперебійної та надійної роботи.

Актуальність цього програмного продукту зумовлена зростаючим попитом на інструменти для відстеження криптовалют у цифрову епоху. Зі зростанням популярності криптовалют і потребою інвесторів бути в курсі подій на ринку, веб-трекер пропонує зручне та ефективне рішення.

Ключові слова: КРИПТОВАЛЮТА, ТРЕКЕР РИНКУ, ВЕБ-ДОДАТОК, REACT.JS, FIREBASE, VISUAL STUDIO CODE.

ABSTRACT

Explanatory note: 87 p., 21 figs., 3 appx., 24 sources.

Object of development: A web application for tracking the cryptocurrency market.

The aim of this thesis is to develop a web-based cryptocurrency market tracker using React.js, Firebase and Visual Studio Code. The application aims to provide users with an intuitive interface and extensive features to meet their cryptocurrency tracking needs.

In the introduction of this thesis, the current state of the cryptocurrency market is analysed and reviewed, the specific purpose of the research is defined, the scope of the project is outlined, and the relevance of the chosen topic is justified. It also presents the task statement, which highlights the goals and objectives of the development process, as well as the software requirements, technologies and tools to be used.

In the first chapter of the thesis, a detailed analysis of the cryptocurrency market is carried out, emphasising the importance and relevance of developing a web tracker. The chapter identifies specific requirements for the tracker's functionality.

The second chapter selects appropriate development platforms and describes the design and development process. The algorithm and structure of the application functionality are also described. In addition, the chapter highlights the process of calling and downloading the application, specifies the input and output data, and provides information on the hardware requirements for running the tracker.

The economic section of the thesis defines the labour intensity of web tracker development. It includes the calculation of the cost of development work and an estimate of the time required to complete it.

The practical significance of the project is to develop a web-based cryptocurrency market tracker with a user-friendly interface. The application aims to provide users with a secure platform for tracking and monitoring cryptocurrency prices, market trends, and portfolio performance. Using technologies such as React.js and Firebase, the tracker offers real-time data updates, personalised watchlists, and user authentication for smooth and reliable operation.

The relevance of this software product is driven by the growing demand for cryptocurrency tracking tools in the digital age. With the growing popularity of cryptocurrencies and the need for investors to keep abreast of market developments, the web tracker offers a convenient and effective solution.

Keywords: CRYPTOCURRENCY, MARKET TRACKER, WEB APPLICATION, REACT.JS, FIREBASE, VISUAL STUDIO CODE.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

Криптовалюта – різновид цифрової валюти, емісія та облік якої виконується децентралізованою платіжною системою повністю в автоматичному режимі (без можливості внутрішнього або зовнішнього адміністрування).

Firebase – платформа для розробки застосунків, яка допомагає створювати та розвивати програми

React – відкрита бібліотека JavaScript для розробки інтерфейсів користувача.

JavaScript – інтерпретована або JIT-компільована, об'єктно-орієнтована мова з функціями першого класу.

VS Code – редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS.

Git – розподілена система управління версіями.

ПЗ – програмне забезпечення.

JSON – (JavaScript Object Notation) текстовий формат обміну даними, заснований на JavaScript.

ВСТУП

Метою цієї кваліфікаційної роботи є розробка веб-трекера криптовалютного ринку з використанням технологій React та Firebase, спрямованого на полегшення повсякденної криптоаналітики, моніторингу та управління портфелем криптовалют.

Криптовалюти - це цифрові або віртуальні валюти, які покладаються на криптографію для забезпечення безпеки та функціонують незалежно від центральних банків. Оскільки популярність криптовалют продовжує зростати, стає все більш важливим мати надійний інструмент для легкого відстеження динамічних ринкових цін і тенденцій. Розроблене програмне забезпечення надасть користувачам доступ до перегляду та аналізу поточних цін на криптовалюти в режимі реального часу.

React - це широко використовуваний і дуже універсальний фреймворк для розробки веб-додатків, який пропонує надійне поєднання функцій і продуктивності. Firebase, з іншого боку, є комплексною платформою, яка надає основні бекенд-сервіси, включаючи базу даних в режимі реального часу, автентифікацію користувачів та хостинг. Спільне використання React та Firebase дозволяє створити масштабований, адаптивний та надійний додаток для безперешкодного відстеження криптовалют.

В цілому, розроблене програмне забезпечення стане цінним інструментом для криптовалютних ентузіастів та інвесторів, надаючи зручний інтерфейс та розширений функціонал для відстеження ринку в реальному часі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

За останні роки світ криптовалют зазнав величезного зростання та інновацій, трансформуючи фінансовий ландшафт і привертаючи увагу приватних осіб, інвесторів та бізнесу в усьому світі. Криптовалюти, такі як Bitcoin, Ethereum та Ripple, з'явилися як цифрові активи, що функціонують у децентралізованих мережах, використовуючи криптографічні методи для захисту транзакцій та регулювання створення нових одиниць.

Зі зростанням популярності криптовалют потреба в комплексних інструментах для відстеження, моніторингу та аналізу криптовалютного ринку стала першочерговою. Веб-трекер криптовалютного ринку слугує важливою платформою для приватних осіб та організацій для доступу до інформації в режимі реального часу, історичних даних та глибокого аналізу щодо різних криптовалют та їхніх ринків.

Ринок криптовалют вирізняється своєю динамічністю та швидкими темпами розвитку. На відміну від традиційних фінансових ринків, ринок криптовалют працює 24/7, без будь-якого централізованого органу, що регулює його діяльність. Таке безперервне торгове середовище в поєднанні з високою волатильністю цін на криптовалюту створює складні умови для інвесторів і трейдерів.

Ціни на криптовалюти можуть значно коливатися протягом короткого періоду часу під впливом таких факторів, як ринковий попит, регуляторні зміни, технологічний прогрес і настрої інвесторів. Відстеження цих швидких змін і розуміння ринкових тенденцій має вирішальне значення для прийняття обґрунтованих інвестиційних рішень і формулювання ефективних торгових стратегій.

На ринку криптовалют своєчасний доступ до точних і надійних даних має важливе значення для інвесторів, трейдерів і ентузіастів. Дані в режимі реального часу дозволяють учасникам ринку бути в курсі останніх цін, обсягів торгів, ринкової капіталізації та інших важливих показників. Ця інформація дає їм можливість швидко реагувати на ринкові тенденції, визначати потенційні торгові можливості та ефективно управляти ризиками.

Крім того, комплексний аналіз ринку криптовалют може надати цінну інформацію про історичні цінові моделі, кореляції між різними криптовалютами та нові ринкові тенденції. Використовуючи методи аналізу даних, такі як статистичні моделі, технічний аналіз та аналіз настроїв, окремі особи та організації можуть отримати глибше розуміння динаміки ринку та приймати рішення на основі даних.

Веб-трекери криптовалютних ринків відіграють ключову роль у полегшенні доступу учасників ринку до даних у режимі реального часу, історичної інформації та аналітичних інструментів. Ці трекери агрегують дані з різних джерел, включаючи криптовалютні біржі, блокчейн-мережі та ринкові API, щоб надати користувачам централізовану та зручну платформу.

Веб-трекер криптовалютного ринку пропонує такі функції, як відстеження цін у реальному часі, настроювані списки спостереження, історичні графіки цін та оновлення ринку. Це дозволяє користувачам відстежувати кілька криптовалют одночасно, аналізувати ринкові тенденції та приймати обґрунтовані рішення на основі достовірної інформації.

Крім того, веб-трекери часто мають інтерактивний та інтуїтивно зрозумілий користувацький інтерфейс, що забезпечує простоту використання та доступність на різних пристроях і платформах. Вони є незамінними інструментами як для початківців, так і для досвідчених учасників криптовалютного ринку, дозволяючи їм залишатися в курсі подій, ефективно управляти своїми інвестиціями та впевнено орієнтуватися на динамічному криптовалютному ринку.

1.2 Призначення розробки та галузь застосування

Метою розробки веб-трекера криптовалютного ринку є надання користувачам надійної та ефективної платформи для моніторингу та аналізу криптовалютних даних у режимі реального часу. Веб-трекер покликаний надавати точну та актуальну інформацію про ціни на криптовалюти, ринкові тенденції, обсяги торгів та інші релевантні дані. Пропонуючи комплексний огляд криптовалютного ринку, веб-трекер допомагає користувачам приймати обґрунтовані рішення та бути в курсі динамічного розвитку криптовалютного ландшафту.

Сфера застосування веб-трекера криптовалютного ринку широка і різноманітна. Він орієнтований на широке коло користувачів, включаючи криптовалютних трейдерів, інвесторів, фінансових аналітиків, дослідників та ентузіастів. Веб-трекер слугує цінним інструментом для різних осіб та організацій, залучених до криптовалютної екосистеми, таких як:

Криптовалютні трейдери та інвестори: Трейдери та інвестори покладаються на точну і своєчасну ринкову інформацію для прийняття обґрунтованих торгових рішень. Веб-трекер надає їм дані в режимі реального часу про ціни на криптовалюту та ринкові тенденції, що дозволяє їм відстежувати ринкові умови, виявляти вигідні можливості та ефективно управляти своїми інвестиційними портфелями.

Фінансові аналітики: Фінансові аналітики використовують дані криптовалютного ринку для проведення поглиблених досліджень та аналізу ринкових тенденцій. Веб-трекер надає їм доступ до історичних даних і даних в режимі реального часу, які вони можуть використовувати для проведення технічного аналізу, розробки торгових стратегій і оцінки загальних ринкових настроїв.

Загальні ентузіасти криптовалют: Ентузіасти криптовалют, як окремі особи, так і спільноти, зацікавлені в тому, щоб бути в курсі останніх подій на ринку. Веб-трекер пропонує їм зручний інтерфейс для вивчення та відстеження

криптовалют, перегляду ринкової статистики, а також доступу до новин та інформації, пов'язаної з криптовалютною екосистемою.

Розробляючи веб-трекер для криптовалютного ринку, я маю на меті задовольнити інформаційні потреби різних користувачів у криптовалютній індустрії. Веб-трекер слугуватиме надійною та доступною платформою для відстеження та моніторингу даних криптовалютного ринку, надаючи користувачам цінну інформацію та розширюючи їхні можливості для прийняття рішень на цьому ринку, що стрімко розвивається.

1.3 Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

– освітня програма спеціальності 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;

– завдання на кваліфікаційну роботу на тему «Розробка веб-трекеру ринку криптовалют».

1.4 Постановка завдання

Метою цього проекту є розробка веб-додатку, який слугуватиме трекером криптовалютного ринку, надаючи користувачам інформацію та аналіз різних криптовалют у режимі реального часу. Додаток дозволить

користувачам відстежувати ціни на криптовалюти, історичні дані, ринкові тенденції та іншу важливу інформацію для прийняття обґрунтованих інвестиційних рішень.

Для цього веб-додаток буде створено з використанням сучасних технологій веб-розробки. Основною мовою програмування для додатку буде JavaScript, яка широко використовується для веб-розробки завдяки своїй універсальності та розгалуженій екосистемі бібліотек і фреймворків. Зокрема, бібліотека React.js буде використана для створення динамічного та інтерактивного інтерфейсу користувача.

React.js був обраний за його здатність ефективно обробляти складні користувацькі інтерфейси та керувати станом додатку. Завдяки використанню React.js веб-трекер забезпечить безперебійний і чуйний користувальницький інтерфейс, що дозволить користувачам легко орієнтуватися і взаємодіяти з даними криптовалютного ринку.

Веб-трекер буде покладатися на RESTful API для отримання криптовалютних даних в режимі реального часу з різних бірж і джерел даних. API надаватиме стандартизований інтерфейс для доступу та оновлення ринкових даних, забезпечуючи точність та своєчасність інформації, що надається користувачам.

Розробляючи цей веб-трекер криптовалютного ринку, користувачі отримають комплексний інструмент для моніторингу та аналізу тенденцій криптовалютного ринку, що дозволить їм приймати обґрунтовані інвестиційні рішення та залишатися в курсі динамічного розвитку криптовалютної індустрії.

1.5 Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Веб-трекер криптовалютного ринку буде розроблений таким чином, щоб надати користувачам комплексний набір функціональних можливостей для ефективного відстеження та аналізу криптовалютного ринку. Очікується, що кінцевий продукт матиме наступні функціональні характеристики:

– Ринкові дані в реальному часі: Веб-трекер буде отримувати дані в режимі реального часу з різних криптовалютних бірж і представляти їх користувачам. Сюди входять ціни на криптовалюту, обсяги торгів, ринкова капіталізація та інші відповідні показники.

– Розширені графіки та аналіз: Веб-трекер пропонує інтерактивні та настроювані графіки, що дозволяють користувачам аналізувати історичні цінові тенденції, проводити технічний аналіз і виявляти закономірності та індикатори для прийняття обґрунтованих рішень.

– Зручний інтерфейс: Веб-трекер повинен мати інтуїтивно зрозумілий і зручний інтерфейс, що дозволяє користувачам легко переміщатися між різними розділами і отримувати доступ до потрібної інформації без особливих зусиль. Чіткі та візуально привабливі візуалізації покращать користувацький досвід.

– Пошук та фільтрація: Користувачі зможуть шукати конкретні криптовалюти. Що дозволить користувачам швидко знаходити потрібну інформацію і зосередитися на своїх улюблених криптовалютах.

– Сумісність з різними платформами: Веб-трекер має бути сумісним з різними платформами та пристроями, включаючи персональні комп'ютери, ноутбуки, планшети та мобільні телефони. Він має забезпечувати послідовний та оптимізований користувацький досвід на різних розмірах екрану та роздільній здатності.

Відповідаючи цим функціональним вимогам, веб-трекер криптовалютного ринку надасть користувачам необхідні інструменти та інформацію для ефективного моніторингу та аналізу криптовалютного ринку, прийняття обґрунтованих інвестиційних рішень, а також для того, щоб бути в курсі динамічного розвитку криптовалютної індустрії.

1.5.2 Вимоги до інформаційної безпеки

Забезпечення надійної інформаційної безпеки є надзвичайно важливим для веб-трекера ринку криптовалют. Для захисту даних користувачів і забезпечення безпечного середовища для відстеження криптовалют необхідно дотримуватися наступних основних вимог:

Конфіденційність інформації: Важливо підтримувати конфіденційність даних користувачів, включаючи персональні дані, інформацію про обліковий запис та історію транзакцій. Веб-додаток повинен реалізовувати надійні методи шифрування і контролю доступу, щоб запобігти несанкціонованому доступу або розголошенню конфіденційної інформації.

Цілісність даних: Для забезпечення цілісності даних дуже важливо захистити їх від несанкціонованої модифікації або фальсифікації.

Доступність інформації: Веб-трекер повинен забезпечувати постійну доступність інформації для користувачів. Такі заходи, як надійна серверна інфраструктура, балансування навантаження і планування аварійного відновлення, повинні бути реалізовані для мінімізації часу простою і забезпечення безперебійного доступу до додатку і його даних.

Автентичність даних: Для встановлення автентичності даних криптовалютного ринку повинні існувати механізми перевірки джерела та цілісності інформації. Впровадження криптографічних методів, цифрових сертифікатів та надійних джерел даних допомагає забезпечити автентичність даних, представлених на веб-трекері.

Таким чином, веб-трекер ринку криптовалют повинен дотримуватися цих вимог, щоб створити безпечне середовище, яке гарантує конфіденційність, цілісність, доступність та автентичність даних користувачів. Впроваджуючи відповідні заходи безпеки, користувачі можуть довіряти додатку для захисту їхньої конфіденційної інформації та надання надійних і точних послуг з відстеження ринку криптовалют.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для ефективного використання веб-трекера криптовалютного ринку необхідно дотримуватися певних технічних специфікацій та вимог. Ці специфікації охоплюють як апаратні, так і програмні компоненти, необхідні для безперебійної роботи. Нижче наведені рекомендовані вимоги до компонентів системи:

Веб-браузер: Для роботи веб-додатку потрібен сумісний веб-браузер. Рекомендується використовувати популярні браузери, сумісні з операційною системою Windows. Мінімальні специфікації для більшості браузерів включають

- Процесор: Pentium 4 або еквівалент
- Графічний адаптер: nVidia, Intel, AMD/ATI 3D адаптер
- Відеопам'ять: 128 МБ
- Сховище для зберігання даних: Не менше 150 ГБ
- Оперативна пам’ять: Мінімум 2 ГБ

Мобільні платформи: Для користувачів, які отримують доступ до веб-трекера через мобільні пристрої, для оптимальної роботи потрібні певні версії операційних систем. Мінімально підтримувані версії

- iOS: Версія 13.0 і вище
- Android: Версія 6 і вище

Дотримання цих специфікацій забезпечує безперебійну роботу з сайтом, дозволяючи користувачам отримувати доступ до інформації про ринок

криптовалют в режимі реального часу і ефективно відстежувати свої інвестиції на різних платформах і пристроях.

1.5.4 Вимоги до інформаційної та програмної сумісності

Важливо забезпечити сумісність між веб-трекером криптовалютного ринку та інформацією і програмним забезпеченням, яке використовують користувачі. У той час як старіші версії браузерів можуть не підтримувати останні інновації, бібліотеки або фреймворки, сучасні браузери зазвичай мають функцію автоматичного оновлення, що гарантує, що користувачі можуть отримати доступ до веб-додатка без проблем із сумісністю.

Для забезпечення безперебійної роботи користувачів рекомендується використовувати найновіші версії найпоширеніших браузерів. Ці браузери часто включають в себе новітні технології та вдосконалення, що дозволяє користувачам повною мірою використовувати можливості і функції, пропоновані веб-трекером.

Важливо переконатися, що веб-трекер розроблений з використанням широко підтримуваних і сумісних бібліотек і фреймворків. Це гарантує, що користувачі зможуть отримати доступ до трекера, не стикаючись з проблемами.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Функціональне призначення веб-трекера криптовалютного ринку полягає в тому, щоб надати користувачам потужний інструмент для моніторингу, аналізу та відстеження ефективності різних криптовалют в режимі реального часу. Додаток буде розроблено з використанням сучасних веб-технологій, включаючи JavaScript та ReactJS[4], щоб забезпечити ефективний та зручний користувацький досвід. Основна функціональність додатку полягає в наступному:

Агрегація криптовалютних даних: додаток збиратиме та агрегуватиме дані з різних криптовалютних бірж та джерел. Він буде отримувати таку інформацію, як ціни на криптовалюту, обсяги торгів, ринкова капіталізація та інші відповідні показники, щоб надати користувачам вичерпні ринкові дані.

Оновлення ринку в режимі реального часу: веб-трекер буде постійно оновлювати дані криптовалютного ринку, щоб забезпечити користувачам доступ до найсвіжішої інформації. Він буде відображати зміни цін в реальному часі, ринкові тенденції та інші актуальні оновлення, щоб допомогти користувачам приймати обґрунтовані рішення.

Управління списком спостереження: користувачі зможуть створювати та управляти списком улюблених монет. За допомогою цих списків можна відстежувати актуальні ціни на різні криптовалюти.

Розширені графіки та аналіз: додаток буде пропонувати розширені інструменти побудови графіків і можливості аналізу, які допоможуть користувачам аналізувати історичні цінові тенденції, виконувати технічний аналіз і визначати патерни або індикатори для торгових стратегій.

Зручний інтерфейс: Додаток матиме інтуїтивно зрозумілий та зручний інтерфейс, що дозволить користувачам легко переміщатися між різними

розділами, отримувати доступ до необхідної інформації та ефективно виконувати бажані дії.

Виконуючи ці функціональні вимоги, веб-трекер криптовалютного ринку має на меті надати користувачам комплексний набір інструментів та інформації для ефективного відстеження та аналізу криптовалютного ринку. Додаток забезпечить безперебійну та ефективну роботу користувачів, допомагаючи як початківцям, так і досвідченим криптовалютним трейдерам у прийнятті обґрунтованих інвестиційних рішень.

2.2. Опис застосованих математичних методів

Проектування та розробка веб-трекера криптовалютного ринку не вимагали використання складних математичних методів або алгоритмів. Основна увага при розробці цього додатку була зосереджена на отриманні та обробці ринкових даних в режимі реального часу, а не на складних математичних розрахунках. Математичні операції, що використовувалися в додатку, були переважно базовими арифметичними обчисленнями.

Хоча веб-трекер включає різні статистичні та аналітичні функції, які допомагають користувачам зрозуміти ринкові тенденції, ці функції не покладаються на складні математичні моделі.

2.3. Опис використаної архітектури та шаблонів проектування

При розробці веб-трекера криптовалютного ринку було використано поєднання архітектури та патернів проектування для забезпечення ефективної та масштабованої роботи. Ключовими компонентами архітектури є використання принципів REST (Representational State Transfer), HTTP-запитів та API.

REST або RESTful архітектура - це широко прийнятий архітектурний стиль для проектування мережевих додатків. Він надає набір принципів для побудови масштабованих і слабко пов'язаних систем.[14] У трекері

криптовалютного ринку принципи RESTful-архітектури були використані для створення запитів до API, орієнтованого на ресурси. Такий підхід дозволяє легко інтегруватися з іншими системами і забезпечувати стандартизований спосіб взаємодії з додатком. Ключові принципи та характеристики REST включають:

Клієнт-серверну архітектуру: REST розділяє клієнтські та серверні компоненти, дозволяючи їм розвиватися незалежно. Клієнт ініціює запити до сервера, який обробляє ці запити та надсилає відповіді.

Уніфікований інтерфейс: RESTful API дотримуються єдиного і послідовного інтерфейсу для взаємодії. Це включає використання стандартних методів HTTP (GET, POST, PUT, DELETE) для різних типів операцій над ресурсами, використання ідентифікаторів ресурсів (URL) для ідентифікації ресурсів і використання стандартних медіа-типів (таких як JSON або XML) для представлення даних.[16]

Ресурсно-орієнтований: Ресурси є ключовим поняттям у REST. Ресурс - це будь-яка сутність, яку можна ідентифікувати і якою можна маніпулювати, наприклад, користувач, продукт або фрагмент даних. Кожен ресурс повинен мати унікальний ідентифікатор (URI/URL), а API надає операції для виконання дій над цими ресурсами.

Зв'язок без статусу: Кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для того, щоб сервер міг його зрозуміти і обробити. Сервер не зберігає ніякої клієнтської інформації між запитами. Такий підхід покращує масштабованість і спрощує реалізацію сервера.

RESTful API зазвичай використовуються для створення веб-сервісів, які можуть використовуватися різними клієнтами, включаючи веб-браузери, мобільні додатки та інші сервери. Вони забезпечують гнучкий і масштабований спосіб надання даних і функціональних можливостей через Інтернет у стандартизований та інтероперабельний спосіб.[15]

У контексті веб-трекера криптовалютного ринку, RESTful API можна використовувати для доступу та отримання даних, пов'язаних з цінами на

криптовалюту, ринковими тенденціями, обсягами торгів та іншою важливою інформацією. Кінцеві точки API можуть бути призначені для представлення різних ресурсів, таких як криптовалюти або ринкові зведення. Клієнти можуть взаємодіяти з цими кінцевими точками, надсилаючи HTTP-запити (наприклад, GET, POST, PUT, DELETE) для виконання операцій над ресурсами і отримання потрібних даних.

HTTP (Hypertext Transfer Protocol - протокол передачі гіпертексту) визначає набір методів запитів або HTTP-дієслів, які вказують на бажану дію, що має бути виконана на даному ресурсі. Ці методи HTTP забезпечують стандартизований спосіб взаємодії клієнтів із серверами та виконання різних операцій. Ось найпоширеніші методи HTTP-запитів:[17]

- GET: метод GET використовується для отримання представлення ресурсу з сервера. Це безпечний та ідемпотентний метод, що означає, що декілька ідентичних GET-запитів повинні мати такий самий ефект, як і один запит. Відповідь зазвичай містить запитуваний ресурс у тілі повідомлення.
- POST: метод POST використовується для передачі даних, які мають бути оброблені певним ресурсом. Він часто використовується для створення нових ресурсів на сервері. На відміну від GET, POST не є ідемпотентним, що означає, що декілька однакових POST-запитів можуть мати різні наслідки. Дані, що передаються, включаються в тіло повідомлення.
- PUT: метод PUT використовується для оновлення існуючого ресурсу на сервері. Він замінює все представлення ресурсу корисним навантаженням запиту. PUT є безсилим, оскільки декілька однакових запитів PUT повинні мати той самий ефект, що і один запит.
- DELETE: метод DELETE використовується для видалення вказаного ресурсу на сервері. Він видаляє ресурс, ідентифікований URL-адресою запиту. DELETE є ідемпотентним, тобто декілька

ідентичних запитів DELETE повинні мати той самий ефект, що і один запит.

- PATCH: метод PATCH використовується для часткового оновлення існуючого ресурсу. Він схожий на PUT, але замість того, щоб замінити все представлення, PATCH оновлює лише певні поля або властивості ресурсу.
- HEAD: метод HEAD схожий на GET, але він отримує тільки заголовки відповіді, без тіла повідомлення. Він часто використовується для отримання метаданих або перевірки стану ресурсу.
- OPTIONS: метод OPTIONS використовується для отримання підтримуваних параметрів зв'язку для даного ресурсу або сервера. Він може бути використаний для визначення доступних методів запиту, заголовків або інших можливостей.

API (Application Programming Interface - інтерфейс прикладного програмування) - це набір правил і протоколів, який дозволяє різним програмним додаткам спілкуватися і взаємодіяти один з одним. Він визначає методи та формати даних, що використовуються для обміну інформацією та виконання операцій між різними системами або компонентами.[13]

API надають розробникам доступ до функціональних можливостей існуючих програмних сервісів або платформ без необхідності розбиратися в деталях їхньої реалізації. Вони абстрагуються від складності базової системи і надають спрощений інтерфейс, з яким можуть працювати розробники.

API можуть слугувати різним цілям, зокрема:

Інтеграція: API дозволяють різним програмним системам з'єднуватися та обмінюватися даними. Вони дозволяють програмам обмінюватися інформацією, отримувати доступ до ресурсів і виконувати дії на різних платформах. Наприклад, платформи соціальних мереж часто надають API, які дозволяють розробникам інтегрувати свої додатки з такими функціями, як автентифікація, публікація контенту або отримання інформації про користувача.

Розширюваність: API надають засоби для розширення функціональності програмних систем. Вони дозволяють розробникам спиратися на існуючі платформи чи сервіси та додавати власні функції чи можливості. API уможливлюють створення плагінів, розширень або сторонніх інтеграцій, які розширюють функціональність програми.

Розробка: API зазвичай використовуються розробниками як будівельні блоки для створення нових додатків. Надаючи набір чітко визначених методів і структур даних, API надають розробникам необхідні інструменти для взаємодії з певною програмною системою або сервісом. Це прискорює процес розробки і дозволяє розробникам використовувати існуючу функціональність, а не починати з нуля.

API можна класифікувати на різні типи залежно від стилю реалізації та протоколів. Деякі з найпоширеніших типів включають

RESTful API: REST (Representational State Transfer) API дотримуються набору принципів і обмежень для побудови веб-сервісів. Вони використовують стандартні методи HTTP, такі як GET, POST, PUT і DELETE, для виконання операцій над ресурсами, ідентифікованими унікальними URI. RESTful API широко використовуються завдяки своїй простоті, масштабованості та відсутності стану.[16]

SOAP API: SOAP (Simple Object Access Protocol - простий протокол доступу до об'єктів) API використовують для зв'язку протокол SOAP на основі XML. Вони зазвичай покладаються на XML-схеми для визначення структур

даних і WSDL (мова опису веб-сервісів) для опису доступних операцій. SOAP API забезпечують більш жорсткий і формальний підхід до інтеграції в порівнянні з RESTful API.

GraphQL API: GraphQL - це мова запитів і середовище виконання для API. Вона дозволяє клієнтам запитувати певні дані та формувати відповідь відповідно до своїх потреб. API-інтерфейси GraphQL пропонують гнучкість, дозволяючи клієнтам вказувати точні дані, які їм потрібні, зменшуючи надлишкову та недостатню вибірку даних.

Інші спеціалізовані API: Існує безліч інших типів API, призначених для конкретних цілей, таких як API баз даних (наприклад, SQL API), API обміну повідомленнями (наприклад, MQTT), платіжні API (наприклад, PayPal API) та багато інших. Ці API обслуговують певні домени або галузі та надають спеціалізовану функціональність.

Для зв'язку із зовнішніми API та отримання актуальних даних використовувалися HTTP-запити (Hypertext Transfer Protocol). Веб-трекер криптовалютного ринку надсилав HTTP-запити до кінцевих точок API різних криптовалютних бірж або постачальників ринкових даних. Ці запити, як правило, робилися з використанням методу GET для отримання такої інформації, як ціни на криптовалюту, обсяги торгів і ринкові тенденції.[17]

Поєднання RESTful архітектури, HTTP-запитів та зовнішнього API дозволило розробити адаптивний та надійний веб-трекер криптовалютного ринку. Принципи RESTful-дизайну забезпечили чітку структуру запитів до API, а HTTP-запити дозволили безперешкодно інтегруватися із зовнішніми джерелами даних.

2.4. Опис використаних технологій та мов програмування

В розробці цього проекту були використані такі технології та мови:

- Java-Script
- React
- Axios
- Material-UI
- React-Router-DOM
- React Alice Carousel
- CoinGecko Crypto API
- Firebase

JavaScript - це високорівнева інтерпретована мова програмування, яка використовується переважно для розробки інтерфейсів. Це універсальна мова, яка також може бути використана для бекенд-розробки, розробки мобільних додатків, ігор тощо. Ось кілька ключових моментів про JavaScript:[18-20]

Веб-розробка на стороні клієнта: JavaScript насамперед відома завдяки своєму використанню у клієнтській веб-розробці. Вона дозволяє розробникам створювати інтерактивні та динамічні веб-сторінки, маніпулюючи об'єктною моделлю документа (DOM), обробляючи події користувача та роблячи асинхронні запити до серверів.

Крос-браузерна сумісність: JavaScript підтримується всіма основними веб-браузерами, включаючи Chrome, Firefox, Safari та Edge. Це робить її універсальною мовою для створення веб-додатків, які можуть працювати на різних платформах і пристроях.

Об'єктно-орієнтоване програмування (ООП): JavaScript підтримує принципи об'єктно-орієнтованого програмування, такі як інкапсуляція, успадкування та поліморфізм. Це дозволяє розробникам створювати багаторазовий і модульний код, визначаючи класи, об'єкти та методи.

Асинхронне програмування: JavaScript підтримує асинхронне програмування за допомогою таких функцій, як callbacks, promises, та

async/await. Це дозволяє обробляти трудомісткі операції, такі як мережеві запити або доступ до баз даних, не блокуючи виконання іншого коду.

Бібліотеки та фреймворки: JavaScript має величезну екосистему бібліотек і фреймворків, які спрощують і покращують веб-розробку. Популярні фреймворки, такі як React.js, Angular та Vue.js, надають потужні інструменти для створення складних веб-додатків з багаторазовими компонентами та ефективним рендерингом.

Спільнота та підтримка: JavaScript має велику та активну спільноту розробників, які сприяють його розвитку та еволюції. Онлайн-ресурси, форуми та спільноти надають широку документацію, навчальні посібники та підтримку для розробників усіх рівнів кваліфікації.

JavaScript - це потужна і широко використовувана мова програмування, яка зробила революцію у веб-розробці. Її універсальність, сумісність з багатьма браузерами та розгалужена екосистема роблять її важливим інструментом для створення інтерактивних та динамічних веб-додатків. Незалежно від того, чи це маніпулювання елементами веб-сторінки, обробка взаємодії з користувачем або створення складних серверних систем, JavaScript продовжує залишатися на передньому краї сучасної розробки програмного забезпечення.

React - це популярна бібліотека JavaScript, що використовується для створення користувацьких інтерфейсів (UI) у веб-додатках. Розроблена та підтримувана компанією Facebook, React набула широкого розповсюдження завдяки своїй компонентній архітектурі, маніпуляціям з віртуальною DOM (Document Object Model) та ефективним можливостям рендерингу. Ось деякі ключові аспекти React:[1-5]

Компонентна архітектура: React дотримується компонентного підходу, де інтерфейс користувача розбивається на незалежні компоненти, які можна використовувати повторно. Кожен компонент інкапсулює власну логіку, стан та рендеринг, що полегшує створення складних користувацьких інтерфейсів та сприяє повторному використанню коду.

Віртуальний DOM: React використовує віртуальний DOM, який є абстракцією реального DOM-дерева. Віртуальний DOM дозволяє React ефективно оновлювати та рендерити компоненти, обчислюючи мінімальний набір необхідних змін. Ця оптимізація мінімізує перезавантаження браузера та підвищує продуктивність.

Синтаксис JSX: React використовує JSX (JavaScript XML), розширення до JavaScript, яке дозволяє змішувати HTML-подібний синтаксис у JavaScript-коді. JSX надає стислий та декларативний спосіб опису структури та зовнішнього вигляду компонентів інтерфейсу користувача.

Методи життєвого циклу компонентів: React надає набір методів життєвого циклу, які дозволяють розробникам підключатися до різних етапів життя компонента, таких як ініціалізація, монтування, оновлення та демонтаж. Ці методи життєвого циклу дозволяють виконувати певні дії у відповідний час протягом життєвого циклу компонента.

React Native: React виходить за рамки веб-розробки і також використовується для розробки мобільних додатків завдяки React Native. React Native дозволяє створювати крос-платформні мобільні додатки з використанням React та JavaScript, орієнтовані на платформи iOS та Android з єдиною кодовою базою.[7]

Велика та активна спільнота: React має велику та енергійну спільноту розробників, які активно сприяють його розвитку. Спільнота надає обширну документацію, навчальні посібники та бібліотеки з відкритим кодом, які підвищують продуктивність розробки та відповідають різним варіантам використання.

Сила React полягає в його здатності створювати інтерактивні та адаптивні інтерфейси, ефективно оновлювати компоненти та полегшувати повторне використання коду. Його популярність та підтримка спільноти призвели до створення численних сторонніх бібліотек, інструментів та інтеграцій, що робить React універсальним вибором для створення сучасних веб-додатків.

Axios - це популярна бібліотека JavaScript, яка використовується для виконання HTTP-запитів з веб-браузерів або Node.js. Вона надає простий та інтуїтивно зрозумілий API, який підтримує різні функції, що робить її зручним вибором для обробки AJAX-запитів та взаємодії з API. Ось деякі ключові деталі про Axios:

HTTP-запити: Axios підтримує всі основні методи HTTP, включаючи GET, POST, PUT, DELETE та інші. Це дозволяє робити запити до серверів і отримувати дані з API.

На основі обіцянок: Axios побудований на JavaScript Promises, які забезпечують чистий і ефективний спосіб обробки асинхронних операцій. Axios повертає обіцянки при виконанні запитів, дозволяючи вам використовувати методи then() і catch() для обробки відповідей на успіх і помилки.

Конфігурація запитів: Axios дозволяє конфігурувати кожен запит за допомогою таких параметрів, як заголовки, параметри запиту, тіло запиту, токени автентифікації тощо. Ці параметри можна передати як об'єкт під час створення запиту.

Перехоплювачі: Axios надає перехоплювачі, які дозволяють перехоплювати і змінювати запити або відповіді до того, як вони будуть оброблені. Перехоплювачі корисні для додавання спільних заголовків, обробки автентифікації, ведення журналів та обробки помилок у декількох запитах.

Обробка відповідей: Axios автоматично аналізує дані відповіді на основі типу контенту і надає їх в об'єкті відповіді. Він підтримує JSON, XML, HTML та інші формати даних. Ви можете отримати доступ до статусу відповіді, заголовків і даних, використовуючи властивості об'єкта відповіді.

Скасування запиту: Axios підтримує скасування запиту, що дозволяє вам скасувати поточний запит, якщо він більше не потрібен. Це допомагає у випадках, коли користувач переходить зі сторінки або перериває запит вручну.

Cross-Origin Resource Sharing (CORS): Axios обробляє CORS за замовчуванням і дозволяє вам робити запити до різних доменів або субдоменів,

не стикаючись з проблемами перехресного походження. Він містить необхідні заголовки для CORS і автоматично обробляє попередні запити.

Інтеграція з відомими фреймворками: Axios часто використовується з популярними JavaScript-фреймворками, такими як React, Vue.js та Angular. Його можна легко інтегрувати в ці фреймворки для обробки даних та взаємодії з API.

Загалом, Axios спрощує процес створення HTTP-запитів, надаючи чистий API, надійну обробку помилок, синтаксис на основі обіцянок і підтримку таких функцій, як перехоплювачі та скасування запитів. Його популярність зумовлена простотою використання, гнучкістю та сумісністю з різними JavaScript.

Material-UI - це широко використовувана бібліотека з відкритим вихідним кодом, яка надає набір попередньо розроблених компонентів і стилів для побудови користувацьких інтерфейсів у React-додатках. Вона заснована на принципах Material Design від Google, які сприяють створенню чистого, сучасного та візуально привабливого дизайну. [20]

React Router DOM - це популярна бібліотека, яка надає можливості маршрутизації для React-додатків. Вона дозволяє створювати динамічні та інтерактивні односторінкові додатки (SPA), забезпечуючи навігацію між різними поданнями або компонентами на основі URL-адреси.

React Alice Carousel - це популярний кастомний компонент каруселі для React-додатків. Він дозволяє створювати динамічні та інтерактивні каруселі або слайдери для демонстрації контенту, наприклад, зображень, відео або будь-яких інших React-компонентів.

CoinGecko Crypto API - це комплексний і широко використовуваний API, який надає розробникам доступ до великої кількості даних та інформації, пов'язаних з криптовалютами. Він слугує цінним ресурсом для отримання ринкових даних в реальному часі та історичних даних, інформації про монети, обмінні курси тощо. Ось детальний огляд криптовалютного API CoinGecko:

Покриття даних: CoinGecko Crypto API пропонує широке охоплення даних, пов'язаних з криптовалютою. Він включає інформацію про тисячі криптовалют, таких як Bitcoin (BTC), Ethereum (ETH), Ripple (XRP) та багато інших. API надає дані про ціни на монети, ринкову капіталізацію, обсяги торгів, графіки цін, історичні дані, ринкові тенденції та різні інші показники.

Дані в режимі реального часу та історичні дані: API дозволяє розробникам отримувати як поточні, так і історичні дані про криптовалюти. Дані в режимі реального часу надають актуальну інформацію про зміни цін, обсяги торгів та інші ринкові показники. Історичні дані дозволяють розробникам отримати доступ до минулих ринкових показників і проаналізувати тенденції з плином часу, що може бути корисним для побудови графіків, проведення досліджень або аналізу даних.

Огляд ринку: CoinGecko Crypto API надає кінцеву точку огляду ринку, яка пропонує моментальний знімок всього ринку криптовалют. Це включає загальну ринкову капіталізацію, загальний обсяг торгів та інші агреговані показники. Це дозволяє розробникам отримати швидкий огляд поточного стану ринку та відстежувати загальні ринкові тенденції.

Інформація про монету: За допомогою API розробники можуть отримувати детальну інформацію про окремі криптовалюти. Сюди входять такі дані, як назва монети, символ, логотип, опис, веб-сайт, посилання на соціальні мережі та інші релевантні деталі. Це дозволяє розробникам збирати вичерпну інформацію про конкретні монети та відображати її у своїх додатках.

Обмінні курси: API пропонує дані про обмінні курси, які дозволяють розробникам конвертувати ціни криптовалют в різні фіатні валюти або інші криптовалюти. Ця функція особливо корисна для додатків, які вимагають конвертації валют або відображення цін в різних валютах.

Зручна для розробників документація: CoinGecko Crypto API надає добре задокументовані кінцеві точки і параметри, що полегшує розробникам розуміння і ефективне використання API. Документація містить докладні

пояснення, приклади коду і керівництва, які допоможуть розробникам безперешкодно інтегрувати API в свої додатки.

Доступ до API та аутентифікація: CoinGecko Crypto API пропонує безкоштовний доступ до своїх послуг, що дозволяє розробникам експериментувати і створювати додатки без фінансових бар'єрів. Хоча аутентифікація не потрібна для більшості кінцевих точок API, API надає можливість аутентифікувати запити за допомогою ключа API для підвищення безпеки і гнучкості обмеження курсу.

Підтримка спільноти: CoinGecko Crypto API має велику і активну спільноту розробників, яка надає підтримку, ділиться прикладами коду і обговорює кращі практики. Розробники можуть звертатися за допомогою або брати участь в обговореннях на форумах, платформах соціальних мереж і в спільнотах розробників, що може бути корисно для усунення несправностей і вивчення чужого досвіду.

На завершення, CoinGecko Crypto API - це всеосяжний і надійний ресурс для доступу до даних, пов'язаних з криптовалютою. Широке охоплення даних, можливості роботи з даними в режимі реального часу та історичними даними, огляд ринку, інформація про монети, курси обміну та зручна для розробників документація роблять його цінним інструментом для розробників, які бажають створювати додатки, пов'язані з криптовалютами, виконувати аналіз даних або відстежувати ринкові тенденції.

Firebase - це комплексна платформа для мобільної та веб-розробки від Google. Вона пропонує набір хмарних сервісів та інструментів, які дозволяють розробникам швидко та ефективно створювати та розгортати високоякісні додатки. Firebase надає різні сервіси, але в кваліфікаційній роботі я зосередився на базі даних Firebase та аутентифікації.[8-12]

База даних Firebase:

Firebase Database - це гнучка і масштабована хмарна база даних NoSQL, яка дозволяє розробникам зберігати і синхронізувати дані для своїх додатків в

режимі реального часу. Ось деякі ключові особливості та функціональні можливості Firebase Database:

Синхронізація даних в режимі реального часу: Firebase Database використовує механізм синхронізації даних у режимі реального часу, що означає, що будь-які зміни, внесені до даних у базі даних, миттєво поширюються на всі підключені пристрої. Ця синхронізація в режимі реального часу дозволяє розробникам створювати спільні та адаптивні додатки, де кілька користувачів можуть одночасно взаємодіяти з одними і тими ж даними.

Структура даних NoSQL: База даних Firebase використовує структуру даних NoSQL, що означає, що вона використовує гнучкий, безсхемний підхід до зберігання даних. Це дозволяє розробникам зберігати та отримувати дані у вигляді JSON-документів, що полегшує адаптацію даних без необхідності масштабної міграції схеми бази даних.

Офлайн підтримка: Firebase Database забезпечує автономну підтримку, дозволяючи додаткам продовжувати працювати навіть за відсутності мережевого з'єднання. Коли пристрій знаходиться в автономному режимі, Firebase Database зберігає внесені зміни локально і синхронізує їх з сервером, як тільки з'єднання відновлюється. Ця автономна підтримка забезпечує безперебійну роботу користувачів і узгодженість даних на різних пристроях.

Безпека та правила: Firebase Database пропонує потужну та гнучку модель безпеки. Розробники можуть визначати правила безпеки, які визначають, хто може мати доступ до даних і змінювати їх. Ці правила написані з використанням декларативної мови і можуть бути налаштовані відповідно до конкретних вимог програми. Firebase Database інтегрується з Firebase Authentication для автентифікації користувачів і контролю доступу до даних.

Firebase Authentication:

Firebase Authentication забезпечує безпечну і просту у використанні систему автентифікації для управління користувачами в додатках на основі

Firebase. Нижче наведені ключові особливості та функціональні можливості Firebase Authentication:

Автентифікація користувачів: Firebase Authentication підтримує різні методи автентифікації, включаючи електронну пошту/пароль, номер телефону, Google, Facebook, Twitter тощо. Це спрощує процес автентифікації користувачів, дозволяючи розробникам швидко і безпечно інтегрувати автентифікацію в свої додатки.

Управління користувачами: Firebase Authentication надає набір API та інструментів для управління обліковими записами користувачів. Це дозволяє розробникам легко обробляти реєстрацію користувачів, автентифікацію, скидання паролів, перевірку електронної пошти та інші операції, пов'язані з користувачами.

Постачальники ідентифікаційних даних: Firebase Authentication інтегрується з популярними постачальниками ідентифікаційних даних, такими як Google, Facebook і Twitter, дозволяючи користувачам входити в додаток, використовуючи свої існуючі облікові записи на цих платформах. Це спрощує процес реєстрації та покращує користувацький досвід.

Безпека та контроль доступу: Firebase Authentication забезпечує безпечну автентифікацію користувачів шляхом обробки облікових даних, управління сеансами та безпечної генерації токенів. Вона надає методи для реалізації контролю доступу на основі ролей, дозволяючи розробникам контролювати і обмежувати доступ до певних ресурсів на основі ролей і дозволів користувачів.

Інтеграція зі службами Firebase: Firebase Authentication легко інтегрується з іншими службами Firebase, такими як База даних Firebase, Хмарні функції та Хмарне сховище. Ця інтеграція дозволяє розробникам створювати безпечні та персоналізовані додатки, використовуючи дані автентифікації користувачів у всій екосистемі Firebase.

Таким чином, Firebase надає потужну та багатофункціональну платформу для створення додатків, а база даних Firebase та автентифікація є ключовими компонентами пакету Firebase. Firebase Database пропонує синхронізацію

даних в режимі реального часу, підтримку офлайн і гнучку структуру даних NoSQL, в той час як Firebase Authentication спрощує автентифікацію користувачів і управління ними. Разом ці сервіси дозволяють розробникам створювати надійні, масштабовані та безпечні додатки з можливістю безперебійної синхронізації даних та автентифікації користувачів.

2.5. Опис структури програми та алгоритмів її функціонування

Для наочного зображення алгоритмів, що використовуються у веб-трекері, було використано UML-діаграму. Ця діаграма відображає взаємодію користувача з інтерфейсом застосунку. Як видно з рисунку 2.1, взаємодія користувача включає в себе різні дії, такі як пошук криптовалют, доступ до ринкових даних та аналіз тенденцій. Ця взаємодія дозволяє користувачам ефективно відстежувати та контролювати різні монети, що дає їм змогу приймати обґрунтовані інвестиційні рішення.

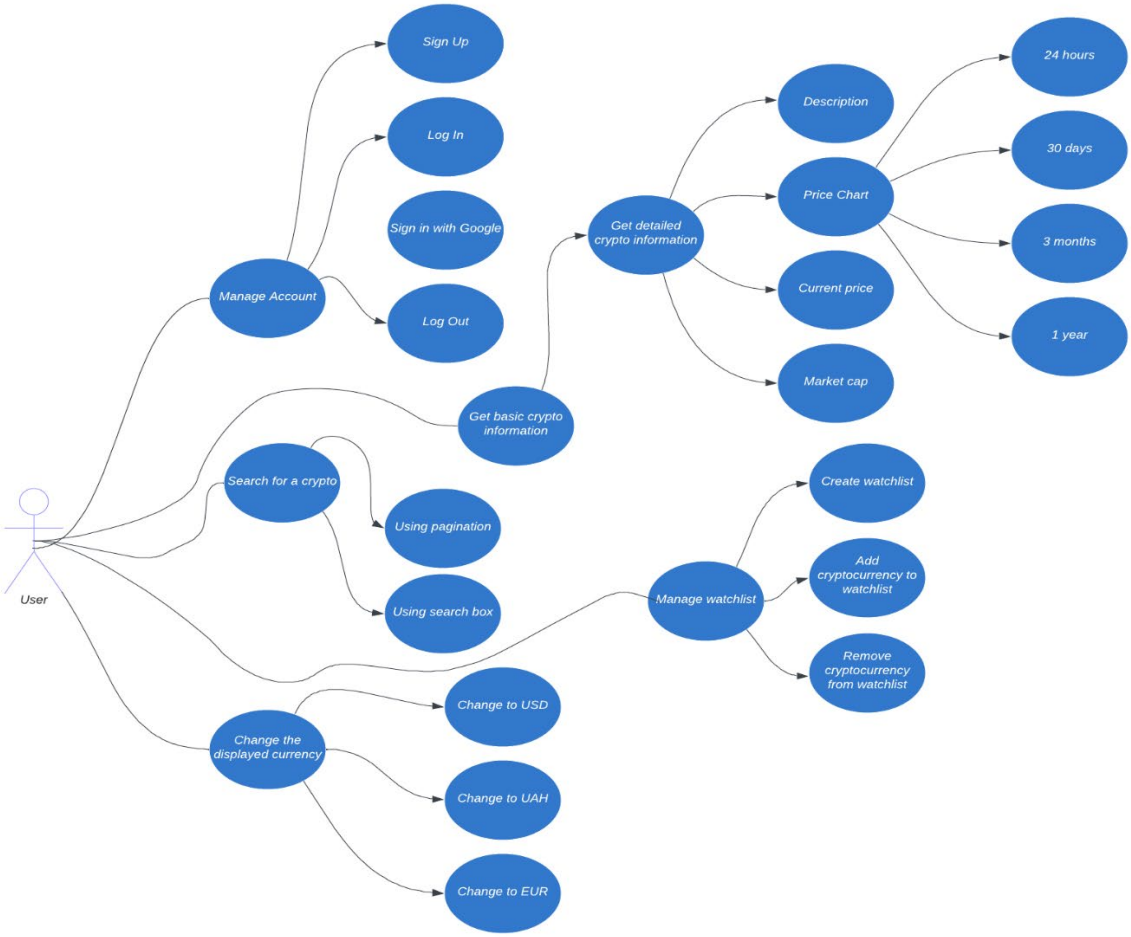

Рис. 2.1. UML-діаграма варіантів використання

При розробці цього веб-застосунку було використано компонентний підхід, що є фундаментальним аспектом розробки на React. У цьому підході кожен елемент, присутній на веб-сторінці, розглядається як окремий компонент, що дозволяє повторно використовувати його в різних частинах проекту. Важливо, що ці компоненти є незалежними сутностями, що полегшує управління та обслуговування.

Прийняття компонентного підходу в React не тільки рекомендується, але й підкреслюється в офіційній документації. Цей підхід значно полегшує процес розробки, пропонуючи масштабованість. Компоненти можуть варіюватися від невеликих і тривіальних до великих розділів, блоків або навіть цілих сторінок, що робить його дуже гнучким і адаптованим до вимог проекту.

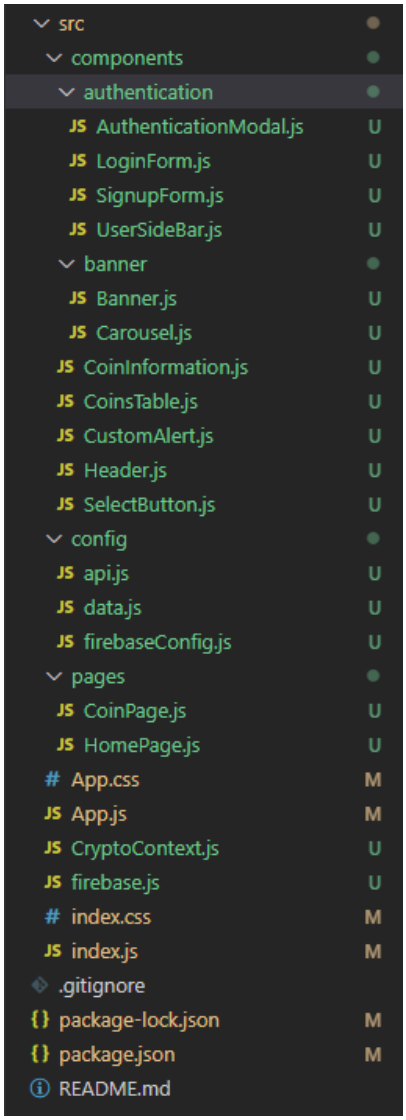

Рис. 2.2. Перелік файлів проекту

2.6. Обґрунтування та організація вхідних та вихідних даних програми

У веб-трекері криптовалют обмін даними відбувається у форматі JSON, який виявляється придатним як для обробки на стороні сервера, так і для обробки на стороні клієнта.

Вхідні дані для цього проекту охоплюють різні типи інформації. Користувачі мають можливість зареєструватися, використовуючи свої адреси електронної пошти, створюючи персоналізовані облікові записи, які дозволяють їм отримати доступ до додаткових функцій. Крім того, користувачі можуть створити список спостереження за конкретними криптовалютами, в яких вони зацікавлені, що дозволяє їм відстежувати продуктивність і коливання цих конкретних монет.

На виході веб-трекер надає користувачам цінну інформацію, пов'язану з ринком криптовалют. Сюди входять дані в режимі реального часу про ціни на криптовалюту, обсяги торгів, ринкову капіталізацію та інші відповідні показники. Користувачі можуть переглядати графіки, що відображають історичні показники різних криптовалют, полегшуючи аналіз ринкових тенденцій.

Виправданням для цих входів і виходів є задоволення потреб криптовалютних ентузіастів та інвесторів, які шукають точну і своєчасну інформацію про ринок. Надаючи безперешкодний доступ до даних, персоналізованих списків спостереження та інформативних візуалізацій, веб-трекер дозволяє користувачам приймати обґрунтовані рішення щодо своїх криптовалютних інвестицій. Організація цих вхідних і вихідних даних забезпечує інтуїтивно зрозумілий інтерфейс, що підвищує загальний користувацький досвід.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для комфортної роботи з сучасними веб-браузерами розробники рекомендують такі мінімальні вимоги до комп'ютера:

1. Процесор (CPU): двоядерний процесор (наприклад, Intel Core i3) або еквівалент.
2. Графічний адаптер: інтегрована графіка або спеціальна відеокарта з оновленими драйверами.
3. Оперативна пам'ять: щонайменше 4 ГБ оперативної пам'яті для безперебійної роботи в багатозадачному режимі та керування вкладками.
4. Сховище: достатньо вільного місця для зберігання інсталяцій браузера та тимчасових файлів.
5. Операційна система: остання версія підтримуваної операційної системи (наприклад, Windows 10, macOS, Linux).
6. Дисплей: монітор з роздільною здатністю 1280x800 пікселів або вище для кращого перегляду.
7. Підключення до Інтернету: стабільне та надійне з'єднання з інтернетом для перегляду та доступу до веб-контенту.

Ось специфікації обладнання, яке використовувалось під час створення кваліфікаційної роботи:

1. Процесор (CPU): AMD Ryzen 7 1700x.
2. Графічний адаптер: GTX 1660 ti.
3. Відеопам'ять: 6 ГБ.
4. Сховище: накопичувач на 1 ТБ.
5. Оперативна пам’ять: 16 ГБ.
6. Периферійні пристрої: клавіатура, миша та монітор.

2.7.2. Використані програмні засоби

Під час розробки проекту для кваліфікаційної роботи були використані наступні програмні засоби:

- Visual Studio Code
- Node.js
- Git

Visual Studio Code (VS Code) - це легкий, універсальний і широко використовуваний редактор вихідного коду, розроблений компанією Microsoft. Він є безкоштовним і має відкритий вихідний код, надаючи розробникам потужний інструмент для написання, редагування та налагодження коду на різних мовах програмування та платформах. Ось деякі ключові особливості та функціональні можливості Visual Studio Code:[21]

Крос-платформна сумісність: VS Code доступний для операційних систем Windows, macOS та Linux, що дозволяє розробникам безперешкодно працювати на різних платформах.

Інтуїтивно зрозумілий користувальницький інтерфейс: Інтерфейс користувача Visual Studio Code чистий, сучасний та легко налаштовується. Він забезпечує середовище кодування, що не відволікає від роботи, з такими функціями, як розділене представлення, настроюваний макет і міні-карти для зручної навігації.

Широка мовна підтримка: VS Code підтримує широкий спектр мов програмування з коробки, включаючи такі популярні мови, як JavaScript, Python, C++, Java, HTML, CSS та багато інших. Він забезпечує підсвічування синтаксису, IntelliSense (завершення коду) та розширення для конкретних мов для підвищення продуктивності.

Розширення та маркетплейс: Visual Studio Code має широку екосистему розширень, створених спільнотою. Ці розширення додають додаткові функціональні можливості, такі як фрагменти коду, лінтери, засоби налагодження, інтеграція контролю версій та підтримка різних фреймворків і

бібліотек. VS Code Marketplace дозволяє користувачам легко знаходити та встановлювати розширення прямо з редактора.

Інтегрований термінал: VS Code надає інтегрований термінал у редакторі, що дозволяє розробникам виконувати команди та запускати скрипти без перемикання на зовнішній термінал. Ця функція сприяє безперебійному процесу розробки та швидкому налагодженню.

Інтеграція з Git'ом: Visual Studio Code має чудову інтеграцію з системою контролю версій Git, надаючи такі можливості, як керування вихідним кодом, візуалізація гілок, історія комітів та перегляд відмінностей поруч. Це дозволяє розробникам зручно співпрацювати та керувати своїми сховищами коду.

Можливості налагодження: VS Code пропонує надійні засоби налагодження з підтримкою декількох мов програмування. Розробники можуть встановлювати точки зупинки, переходити по коду, перевіряти змінні та налагоджувати свої програми за допомогою вбудованих інструментів налагодження або розширень для конкретної мови.

Розширюваність та кастомізація: Розширюваність VS Code - одна з найсильніших його особливостей. Розробники можуть налаштувати майже кожен аспект редактора, від тем та іконок до сполучень клавіш та налаштувань. Користувачі можуть налаштувати середовище кодування відповідно до своїх уподобань та робочих процесів.

Node.js - це потужне і широко використовуване середовище виконання з відкритим вихідним кодом, яке дозволяє розробникам запускати JavaScript-код на стороні сервера. Воно побудоване на JavaScript-движку V8, тому самому, на якому працює браузер Google Chrome, і забезпечує керовану подіями, неблокуючу модель вводу/виводу, що робить його високоефективним і масштабованим для створення мережевих додатків.

Ось деякі ключові аспекти та особливості Node.js:[22]

JavaScript на стороні сервера: Node.js дозволяє розробникам писати серверні додатки за допомогою JavaScript, який традиційно асоціюється з клієнтськими скриптами. Ця уніфікація мови програмування дозволяє

розробникам використовувати одну і ту ж мову і кодову базу як на клієнті, так і на сервері, що сприяє повторному використанню коду і підвищенню продуктивності.

Асинхронний та неблокуючий ввід/вивід: Однією з найважливіших переваг Node.js є його асинхронна, неблокуюча модель вводу/виводу. Замість того, щоб блокувати і чекати завершення операцій вводу/виводу, Node.js використовує архітектуру, керовану подіями, і зворотні виклики для асинхронної обробки операцій вводу/виводу. Такий підхід дозволяє Node.js ефективно обробляти велику кількість одночасних з'єднань, що робить його ідеальним для додатків реального часу та веб-сайтів з високим трафіком.

Масштабованість та продуктивність: Завдяки своїй неблокуючій моделі вводу/виводу, Node.js має високу масштабованість і чудово працює під високими навантаженнями. Він може обробляти тисячі одночасних з'єднань з мінімальними ресурсами, що робить його придатним для створення масштабованих веб-додатків, API та мікросервісів.

Широка екосистема пакетів: Node.js має багату екосистему пакетів з відкритим кодом, доступних через реєстр npm (Node Package Manager). npm надає величезну колекцію багаторазових модулів і бібліотек, які розробники можуть легко інтегрувати в свої додатки. Ця велика екосистема пакетів значно прискорює розробку і дозволяє розробникам використовувати існуючі рішення для різних функціональних можливостей.

Підтримка веб-протоколів: Node.js має вбудовану підтримку різних веб-протоколів, включаючи HTTP, HTTPS, WebSocket і TCP/IP. Це дозволяє розробникам легко створювати веб-сервери, RESTful API, додатки для чату в реальному часі, потокові сервери та інші мережеві додатки.

Розширюваність: Node.js надає C++ API, який дозволяє розробникам створювати власні доповнення та розширювати функціональність Node.js. Ця можливість забезпечує інтеграцію з існуючими бібліотеками та системами C/C++, що робить Node.js універсальною платформою для різних випадків використання.

Git - це широко використовувана розподілена система контролю версій, яка дозволяє розробникам відстежувати зміни, співпрацювати над проектами та ефективно керувати вихідним кодом. Розроблений Лінусом Торвальдсом, творцем Linux, Git відомий своєю швидкістю, гнучкістю та надійністю. Він став важливим інструментом для розробки програмного забезпечення і використовується окремими особами та командами всіх розмірів.

Ось деякі ключові аспекти та особливості Git'у:[23-24]

Розподілене керування версіями: Git - це розподілена система контролю версій, що означає, що кожен розробник має повну копію всього сховища, включаючи його історію. Така децентралізація дозволяє працювати в автономному режимі, пришвидшити роботу та безперешкодно співпрацювати з іншими. Розробники можуть фіксувати зміни у своїх локальних репозиторіях, а потім синхронізувати їх з віддаленими репозиторіями.

Розгалуження та злиття: Git надає потужні можливості розгалуження та злиття. Розробники можуть створювати легкі гілки, щоб працювати над новими функціями або виправленнями помилок незалежно. Розгалуження дозволяє здійснювати паралельну розробку, не впливаючи на основну кодову базу. Після завершення змін гілки можуть бути об'єднані назад в основну гілку, що дозволить легко об'єднати зміни.

Історія та відстеження комітів: Git зберігає детальну історію коммітів, яка відстежує кожну зміну, внесену до сховища. Кожна фіксація являє собою знімок проекту в певний момент часу. Розробники можуть легко переглядати, порівнювати і повертатися до попередніх версій коду. Git також відстежує метадані, такі як авторство, мітки часу та повідомлення про комміти, надаючи повну історію розвитку проекту.

Спільна робота та віддалені сховища: Git полегшує співпрацю між розробниками, дозволяючи їм одночасно працювати над однією і тією ж кодовою базою. Віддалені репозиторії, наприклад, розміщені на таких платформах, як GitHub, GitLab або Bitbucket, слугують центральними репозиторіями, куди члени команди можуть вносити свої зміни, ділитися кодом

та переглядати роботу один одного. Віддалені репозиторії забезпечують централізоване місце для співпраці, перегляду коду та управління проектами.

Етапування та фіксація змін: Git дозволяє розробникам вибірково вносити зміни, перш ніж фіксувати їх. Цей етап дозволяє точно контролювати, які зміни включати в кожну фіксацію. Розробники можуть розділяти логічні зміни на менші, сфокусовані комміти, що полегшує перегляд, повернення або відбір певних наборів змін.

2.7.3. Виклик та завантаження програми

Розроблений веб-трекер ринку криптовалют доступний в Інтернеті за стандартизованою адресою (URL: http://localhost:3000) за допомогою сучасних браузерів, що підтримуються різними операційними системами. Для настільних операційних систем, таких як Windows 10, Linux та macOS, підтримуються такі популярні браузери, як Google Chrome, Firefox та Brave. На мобільних платформах сумісні такі браузери, як Google Chrome, Dolphin, Opera Mobile, Mozilla Firefox і Safari.

Додаток призначений для роботи на останніх двох версіях всіх підтримуваних браузерів, як визначено офіційними розробниками. Хоча додаток може працювати і на старіших версіях браузерів, його успішна робота в них не гарантується.

Тому користувачі можуть отримати доступ до веб-трекеру через свій улюблений браузер, що забезпечує широку сумісність і доступність на різних платформах, операційних системах і пристроях.

2.7.4. Опис інтерфейсу користувача

Веб-трекер криптовалют має зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам зручно відстежувати та аналізувати криптовалютні дані. Після запуску веб-застосунку користувачу відкриється головна сторінка (Homepage), на якій зібрана базова інформація про криптовалюти.

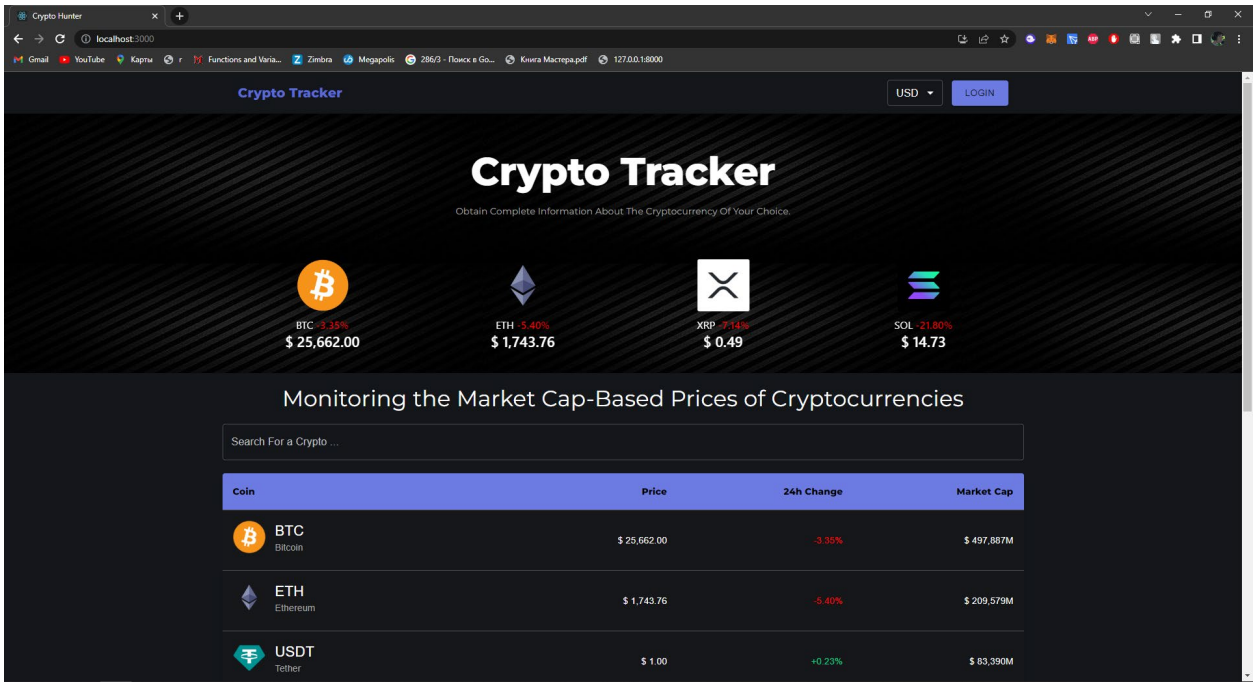

Рис. 2.3. Головна сторінка

На головній сторінці помітно виділяється заголовок, який містить кнопку "Вхід". При натисканні на цю кнопку відкривається модальне вікно, де користувачі можуть або увійти в систему, або зареєструвати новий обліковий запис. Крім того, для більшої зручності користувачі можуть увійти, використовуючи свої облікові дані Google. При успішному вході користувач побачить невелике повідомлення.

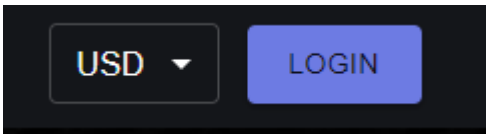

Рис. 2.4. Кнопка логіну

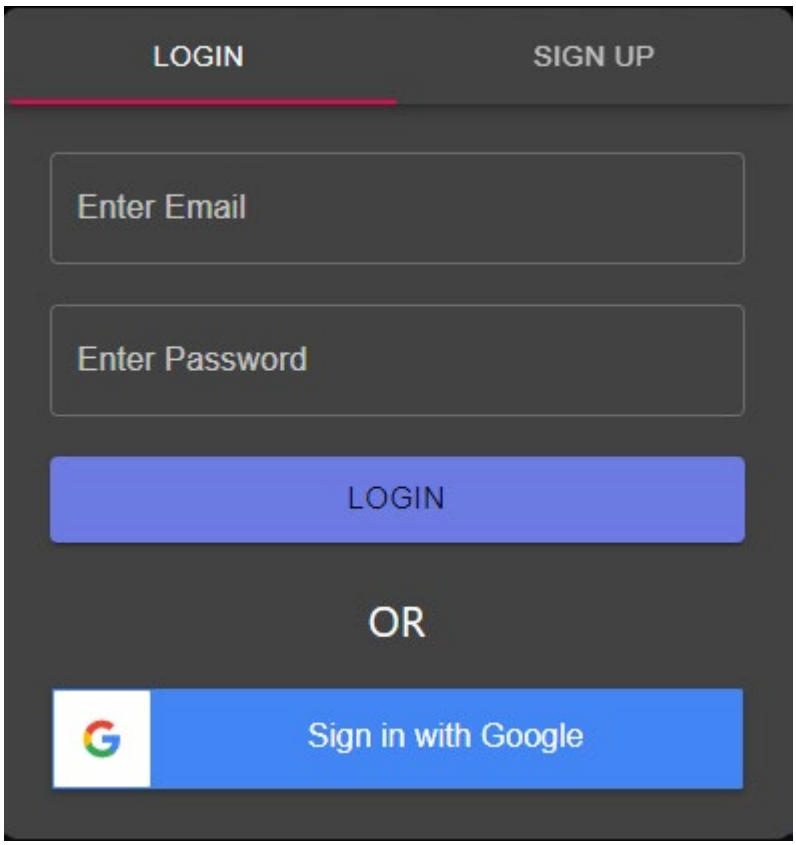

Рис. 2.5. Модальне вікно логіну

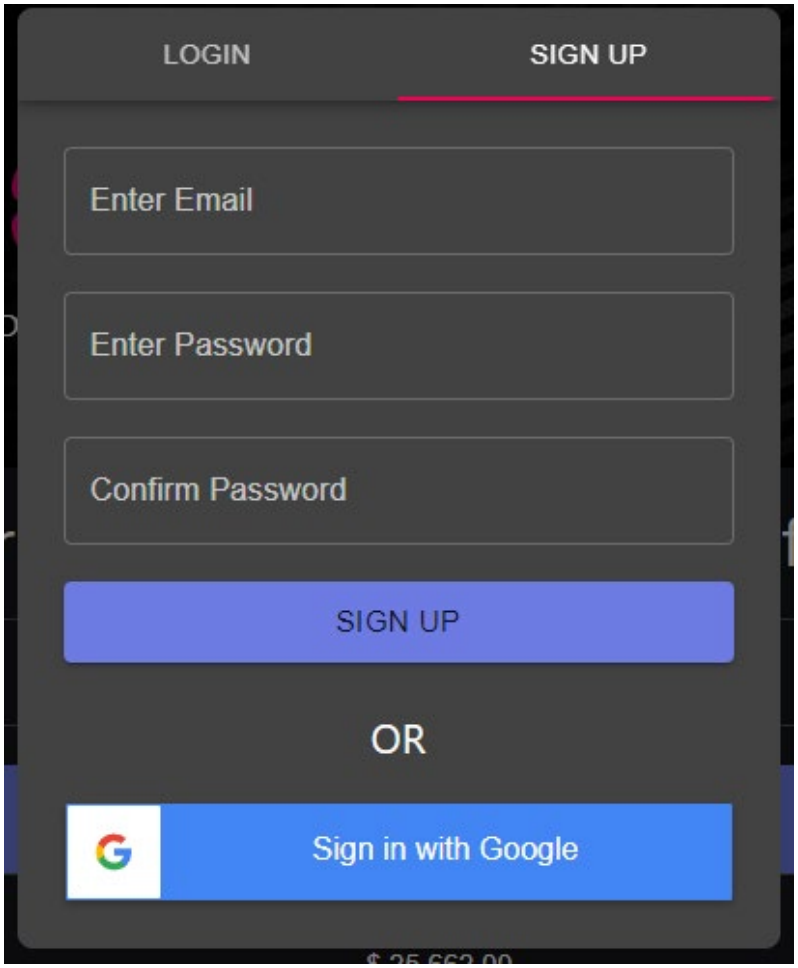

Рис. 2.6. Модальне вікно реєстрації

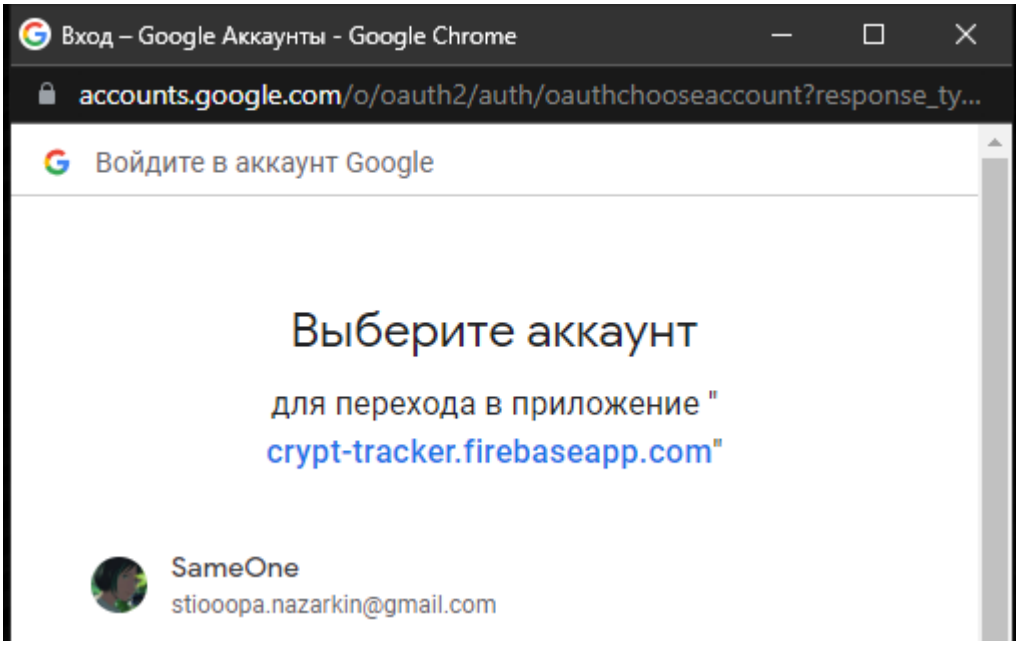

Рис. 2.7. Логін за допомогою Google

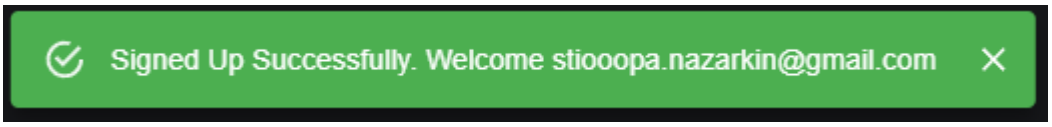

Рис. 2.8. Повідомлення про успішний вхід в акаунт

На головній сторінці трекера розміщено банерний розділ, який висвітлює трендові криптовалюти. У цьому розділі відображаються поточні ціни криптовалют і відсоткові зміни їх вартості.

Рис. 2.9. Банер з трендовими монетами

У заголовку також є випадаюче меню для зміни валюти, що відображається, з такими варіантами, як USD, UAH та EUR.

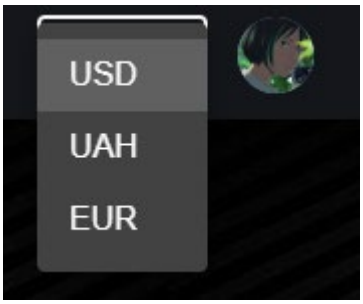

Рис. 2.10. Зміна валюти для цін монет

Рис. 2.11. Результат зміни валюти

На головній сторінці доступне вікно пошуку, що дозволяє користувачам шукати конкретні монети, які їх цікавлять. Функція посторінкової навігації забезпечує легку навігацію по декількох сторінках лістингів криптовалют.

Рис. 2.10. Пошук монети

Рис. 2.11. Пагінація

При натисканні на певну криптовалюту відкривається сторінка з детальним описом. На цій сторінці представлена вичерпна інформація про обрану монету, включаючи її ціну і ринкову капіталізацію.

На сторінці з детальним описом монети також представлений графік цін, що дозволяє користувачам проаналізувати коливання ціни монети протягом різних періодів часу. Доступні кнопки для перемикання між переглядом цінового графіка за 24 години, 30 днів, 3 місяці або 1 рік, що дозволяє користувачам отримати ширшу перспективу щодо продуктивності монети.

Рис. 2.12. Сторінка з детальною інформацією про монету

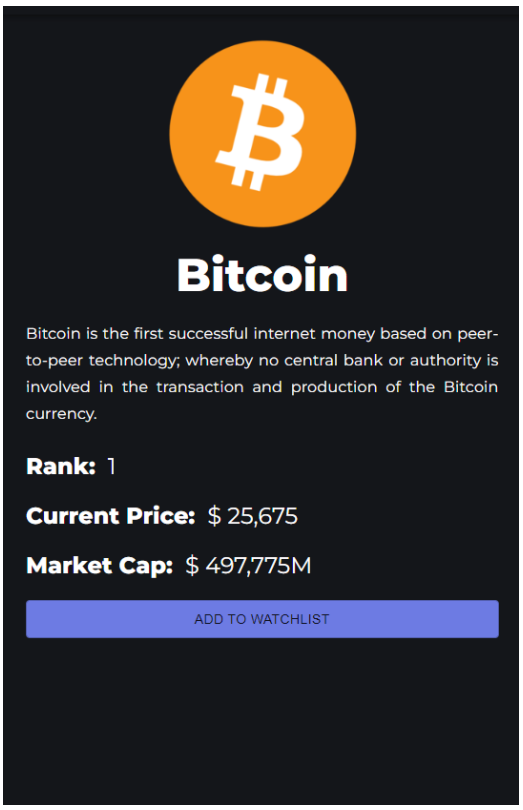

Рис. 2.13. Опис монети

Рис. 2.14. Графік цін

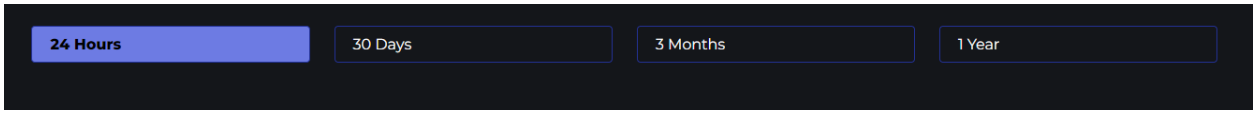

Рис. 2.15. Кнопки, за якими можна змінити діапазон графіку

Рис. 2.16. Графік цін за 30 днів

Рис. 2.17. Графік цін за 3 місяці

Рис. 2.18. Графік цін за 1 рік

Для зареєстрованих користувачів, які увійшли до свого облікового запису, веб-трекер надає можливість додавати або видаляти монети зі свого персонального списку спостереження. Ця функція дозволяє користувачам зручно і ефективно відстежувати свої улюблені криптовалюти.

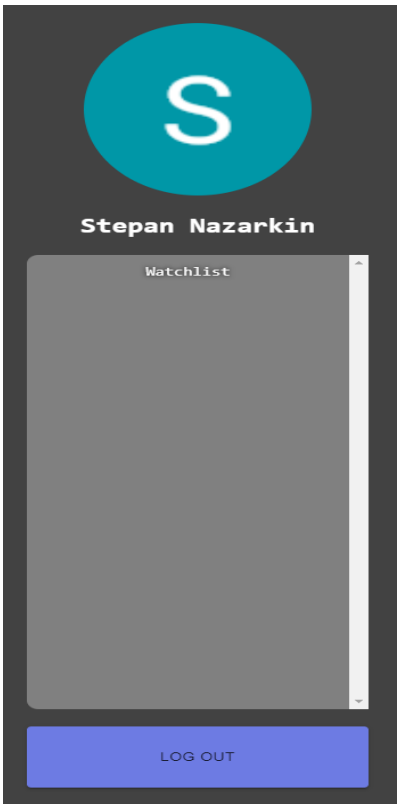

Рис. 2.19. Бокова панель користувача

Рис. 2.20. Кнопка для додавання монети до списку спостереження

Рис. 2.20. Додавання монети до списку спостереження

Рис. 2.20. Кнопка для видалення монети із списку спостереження

Рис 2.21. Видалення монети із списку спостереження

В цілому, користувальницький інтерфейс веб-трекера криптовалют розроблений таким чином, щоб забезпечити користувачам безперебійну та інформативну роботу. Поєднання функціональності входу в систему, варіантів валют, банерів з трендами, детальних описів монет, графіків цін, що налаштовуються, і персоналізованих списків спостереження гарантує, що користувачі мають доступ до необхідних інструментів для ефективного відстеження та аналізу криптовалют.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1573;
2. коефіцієнт складності програми – 1,2;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста – 113 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 17 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

ti – витрати праці на дослідження алгоритму рішення задачі;

ta – витрати праці на розробку блок-схеми алгоритму;

tп – витрати праці на програмування по готовій блок-схемі;

tотл – витрати праці на налагодження програми на ЕОМ;

tд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q – передбачуване число операторів;

С – коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1573 \cdot 1{,}2 \cdot (1 + 0{,}2) = 2265$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot K}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u = (2265 \cdot 1{,}3) / (85 \cdot 1{,}1) = 31{,}5 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}, \tag{3.4}$$

$$t_a = 2265 / (20 \cdot 1{,}1) = 103 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K}, \quad (3.5)$$

$t_n = 2265 / (23 \cdot 1,1) = 89,52$ людино-годин.

Витрати праці на налагодження програми на ЕОМ:

− за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot K}, \quad (3.6)$$

$t_{отл} = 2265 / (5 \cdot 1,1) = 411,81$ людино-годин.

− за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1,2 \cdot t_{отл}, \quad (3.7)$$

$t_{отл}^{k} = 1,2 \cdot 411,81 = 494,17$ людино-годин.

Витрати праці на підготовку документації визначається за формулою:

$$t_д = t_{др} + t_{до}\text{ , людино-годин,}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15...20)*k}\text{ , людино-годин.}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 * t_д \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримуємо:

$$t_{др} = \frac{2265}{20*1{,}1} = 102{,}95, \text{ людино-годин.}$$

$$t_{до} = 0{,}75 * 102{,}95 = 77{,}21, \text{ людино-годин.}$$

$$t_д = 102{,}95 + 77{,}21 = 180{,}16, \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 31{,}5 + 103 + 89{,}52 + 411{,}81 + 180{,}16 = 865{,}99 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ включають витрати на заробітну плату виконавця програми З/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{ грн.,} \qquad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР} \text{ грн.,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{зп} = 865{,}99 * 113 = 97856 \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М} \text{ грн.}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год.

$$З_{мв} = 411{,}81 * 17 = 7000 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{по} = 97856 + 7000 = 104856 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.}, \quad (3.14)$$

де $В_k$- число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$=176 годин).

$$T = \frac{865{,}99}{1 \cdot 176} \approx 4{,}9 \text{міс.}$$

Висновок: вартість розробки веб-трекеру цін на криптовалюту становить 104856 грн. Приблизний час, який буде витрачено на розробку

одним виконавцем – 4,9 місяців при 40-годинному робочому тижні. Такий термін включає в собі час необхідний для дослідження галузі, технологій, розробки клієнтської частини, налагодження серверної, розробки алгоритму розв’язання задачі, створення документації. Загальна кількість людино-годин, яку буде витрачено на розробку – 865,99.

ВИСНОВКИ

Метою даної дипломної роботи було створення зручного та ефективного веб-застосунку, який дозволяє користувачам відстежувати та аналізувати ціни на різні криптовалюти на основі ринкової капіталізації.

В процесі розробки було досягнуто декілька ключових цілей. По-перше, додаток був спроектований і реалізований з використанням сучасних технологій. Ці технології забезпечили міцний фундамент для створення адаптивного та інтерактивного веб-інтерфейсу.

Інтеграція CoinGecko Crypto API дозволила отримувати криптовалютні дані в режимі реального часу, гарантуючи, що користувачі отримують точну та актуальну інформацію. API надає доступ до широкого спектру криптовалютних метрик, включаючи ціни, ринкову капіталізацію, обсяги торгів тощо. Таке широке охоплення даних підвищило загальну функціональність і корисність веб-трекера.

Для управління базами даних та аутентифікації використовувавалась Firebase. Вона забезпечила надійне та безпечне рішення для зберігання даних користувачів, а також надала можливість реєстрації та автентифікації користувачів. Це дозволило користувачам створювати персоналізовані облікові записи та отримувати доступ до власних списків спостереження.

Впровадження різних патернів проектування, таких як компонентна розробка та використання бібліотек React, таких як React Router DOM та Material-UI, сприяло підвищенню загальної ефективності та ремонтопридатності кодової бази. Ці патерни дизайну допомогли створити модульні, багаторазові та масштабовані компоненти, що дозволило створити надійний та гнучкий веб-додаток.

Загалом, веб-трекер криптовалютного ринку успішно досягає своїх цілей, надаючи користувачам інтуїтивно зрозумілу та багатофункціональну платформу для відстеження та аналізу цін на криптовалюти на основі ринкової капіталізації. Він пропонує зручний інтерфейс, оновлення даних в режимі реального часу і персоналізовані списки спостереження.

У майбутньому веб-трекер може бути вдосконалений для розширення функціональності, наприклад, додавання додаткових інструментів аналізу даних, впровадження торгових функцій та інтеграції більшої кількості криптовалютних бірж. Постійний моніторинг та оновлення додатку забезпечить його актуальність та корисність на криптовалютному ринку, що постійно розвивається.

Таким чином, розробка веб-трекера криптовалютного ринку відкриває нові можливості для інвесторів, трейдерів та ентузіастів залишатися в курсі подій та приймати обґрунтовані рішення в динамічному світі криптовалют. Він служить інструментом для навігації в криптовалютному ринку і надає користувачам необхідну інформацію для прийняття розумних інвестиційних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zac Gordon (Author), Mikall Angela Hill (Editor), Robbie Adair (Editor). React Explained: Your Step-by-Step Guide to React. — Independently published, 2020. — 366 с.
2. Alex Banks and Eve Porcello. Learning React: Modern Patterns for Developing React Apps. — O'Reilly Media, 2020. — 307 с.
3. Greg Lim. Beginning React (incl. Redux and React Hooks). — Greg Lim, 2020. — 164 с.
4. React. URL: https://react.dev
5. Robin Wieruch. The Road to React: Your journey to master plain yet pragmatic React.js. — Independently published, 2018. — 292 с.
6. David Griffiths (Author), Dawn Griffiths (Author). React Cookbook: Recipes for Mastering the React Framework. — O'Reilly Media, 2021. — 510 с.
7. Alexander Benedikt Kuttig. Professional React Native: Expert techniques and solutions for building high-quality, cross-platform, production-ready apps. — Packt Publishing, 2022. — 268 с.
8. Gerardus Blokdyk. Firebase The Ultimate Step-By-Step Guide. — 5STARCooks, 2022. — 302 с.
9. Robin Wieruch. The Road to React with Firebase: Your journey to master advanced React for business web applications. — Independently published, 2019. — 199 с.
10. Scott Domes. Progressive Web Apps with React: Create lightning fast web apps with native power using React and Firebase. — Packt Publishing, 2017. — 302 с.
11. Firebase. URL: https://firebase.google.com
12. Houssem Yahiaoui. Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. — Packt Publishing, 2017. — 288 с.
13. JJ Geewax. API Design Patterns. — Manning, 2021. — 480 с.

14. Harihara Subramanian (Author), Pethuru Raj (Author). Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs. — Packt Publishing, 2019. — 378 c.
15. Fernando Doglio. REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development. — Apress, 2018. — 402 c.
16. Matthias Biehl. RESTful API Design: Best Practices in API Design with REST (API-University Series Book 3). — API-University Press, 2017. — 207c.
17. Stephen Ludin (Author), Javier Garza (Author). Learning HTTP/2: A Practical Guide for Beginners. — O'Reilly Media, 2017. — 156 c.
18. David Flanagan. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. — O'Reilly Media, 2020. — 704 c.
19. Philip Ackermann. JavaScript: The Comprehensive Guide. — Rheinwerk Computing, 2023. — 982 c.
20. Adam Boduch. JavaScript: React Material-UI Cookbook: Build captivating user experiences using React and Material-UI. — Packt Publishing, 2019. — 534 c.
21. Bruce Johnson. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. — Wiley, 2019. — 192 c.
22. Jonathan Wexler. Get Programming with Node.js. — Manning, 2019. — 480c.
23. Git. URL: https://git-scm.com
24. Ferdinando Santacroce. Git Essentials: Create, merge, and distribute code with Git, the most powerful and flexible versioning system available, 2nd Edition. — Packt Publishing, 2017. — 238.

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг authentication/AuthenticationModal.js:

```
// Import necessary dependencies and components
import { makeStyles } from "@material-ui/core/styles"
import {useState} from "react"
import Modal from "@material-ui/core/Modal"
import Backdrop from "@material-ui/core/Backdrop"
import Fade from "@material-ui/core/Fade"
import {AppBar, Button, Tab, Tabs, Box} from "@material-ui/core"
import SignupForm from "./SignupForm"
import LoginForm from "./LoginForm"
import {CryptoState} from "../../CryptoContext"
import {auth} from "../../firebase"
import GoogleButton from "react-google-button"
import {GoogleAuthProvider, signInWithPopup} from "firebase/auth"

// Custom styles for the component
const useStyles = makeStyles((theme) => ({
  modalContainer: {
    justifyContent: "center",
    display: "flex",
    alignItems: "center",
  },
  modalContent: {
    backgroundColor: theme.palette.background.paper,
    color: " #FFFFFF ",
    width: 395,
    borderRadius: 8,
  },
  googleContainer: {
    flexDirection: "column",
    textAlign: "center",
    padding: 25,
    paddingTop: 0,
    display: "flex",
    gap: 21,
    fontSize: 21,
  },
}));

// AuthenticationModal component
export default function AuthenticationModal() {
  const classes = useStyles();
  const [isModalOpen, setIsModalOpen] = useState(false);

  const { setAlertMessage } = CryptoState();

  const handleOpenModal = () => {
    setIsModalOpen(true);
  };

  const handleCloseModal  = () => {
    setIsModalOpen(false);
  };

  const [selectedTab, setSelectedTab] = useState(0);
```

```
const handleChange = (event, newValue) => {
  setSelectedTab(newValue);
};

const googleAuthProvider  = new GoogleAuthProvider();

const handleSignInWithGoogle  = () => {
  signInWithPopup(auth, googleAuthProvider )
    .then((res) => {
      setAlertMessage ({
        open: true,
        message: `Signed Up Successfully. Welcome ${res.user.email}`,
        type: "success",
      });

      handleCloseModal();
    })
    .catch((error) => {
      setAlertMessage ({
        open: true,
        message: error.message,
        type: "error",
      });
      return;
    });
};

return (
  <div>
    <Button variant="contained"
      style={{width: 87,height: 39,marginLeft: 14,backgroundColor: "#6d7be3",}}
      onClick={handleOpenModal}
    >
      Login
    </Button>
    {/* Modal component */}
    <Modal
      aria-labelledby="transition-modal-title"
      aria-describedby="transition-modal-description"
      className={classes.modalContainer}
      open={isModalOpen}
      onClose={handleCloseModal}
      closeAfterTransition
      BackdropComponent={Backdrop}
      BackdropProps={{timeout: 499}}
    >
      {/* Modal content */}
      <Fade in={isModalOpen}>
        <div className={classes.modalContent}>
          <AppBar
            position="static"
            style={{backgroundColor: "transparent",color: " #FFFFFF ",}}
          >
            <Tabs
              value={selectedTab}
              onChange={handleChange}
              variant="fullWidth"
              style={{ borderRadius: 8 }}
            >
              <Tab label="Login" />
              <Tab label="Sign Up" />
            </Tabs>
```

```
          </AppBar>
          {/* Render login form or signup form based on the selected tab */}
          {selectedTab === 0 && <LoginForm handleClose={handleCloseModal} />}
          {selectedTab === 1 && <SignupForm handleClose={handleCloseModal} />}
          {/* Google sign-in button */}
          <Box className={classes.googleContainer}>
            <span>OR</span>
            <GoogleButton
              style={{ width: "100%", outline: "none" }}
              onClick={handleSignInWithGoogle}
            />
          </Box>
        </div>
      </Fade>
    </Modal>
  </div>
 );
}
```

Лістинг authentication/LoginForm.js:

```
// Import necessary dependencies and components
import {useState} from "react"
import {TextField, Box, Button} from "@material-ui/core"
import {CryptoState} from "../../CryptoContext"
import {auth} from "../../firebase"
import {signInWithEmailAndPassword} from "firebase/auth"

// LoginForm component
const LoginForm = ({ handleClose }) => {
  // State variables for email and password
  const [loginEmail, setLoginEmail] = useState("");
  const [loginPassword, setLoginPassword] = useState("");

  // Access alert and setAlert functions from CryptoState
  const {setAlertMessage} = CryptoState();

  const handleLogin = async () => {
    // Validate if email and password fields are not empty
    if (!loginEmail || !loginPassword) {
      setAlertMessage({
        open: true,
        message: "Please fill in all the fields",
        type: "error",
      });
      return;
    }

    try {
      // Sign in with email and password using Firebase authentication
      const result = await signInWithEmailAndPassword(auth, loginEmail, loginPassword);
      setAlertMessage({
        open: true,
        message: `Logged in Successfully. Welcome ${result.user.email}`,
        type: "success",
      });

      // Close the login form
      handleClose();
    } catch (error) {
      // Display error message if login fails
      setAlertMessage({
        open: true,
```

```
        message: error.message,
        type: "error",
      });
      return;
    }
  };

  return (
    <Box p={3} style={{display: "flex",flexDirection: "column",gap: "20px",}}>
      <TextField
        variant="outlined"
        type="email"
        label="Enter Email"
        value={loginEmail}
        onChange={(e) => setLoginEmail(e.target.value)}
        fullWidth
      />
      <TextField
        variant="outlined"
        label="Enter Password"
        type="password"
        value={loginPassword}
        onChange={(e) => setLoginPassword(e.target.value)}
        fullWidth
      />
      <Button
        variant="contained"
        size="large"
        onClick={handleLogin}
        style={{ backgroundColor: "#6d7be3" }}
      >
        Login
      </Button>
    </Box>
  );
};

export default LoginForm;
```

Лістинг authentication/SignupForm.js:

```
// Import necessary dependencies and components
import {useState} from "react"
import {TextField, Box, Button,} from "@material-ui/core"
import {CryptoState} from "../../CryptoContext"
import {createUserWithEmailAndPassword} from "firebase/auth"
import {auth} from "../../firebase"

const SignupForm = ({ handleClose }) => {
  // State variables for form inputs
  const [signUpEmail, setSignUpEmail] = useState("");
  const [signUpPassword, setSignUpPassword] = useState("");
  const [confirmSignUpPassword, setConfirmSignUpPassword] = useState("");

  // Accessing global state using context
  const {setAlertMessage} = CryptoState();

  // Handle form submission
  const handleSignup = async () => {
    // Check if passwords match
    if (signUpPassword !== confirmSignUpPassword) {
      setAlertMessage({
        open: true,
        message: "Passwords do not match",
        type: "error",
      });
      return;
    }

    try {
      // Create a new user with email and password
      const result = await createUserWithEmailAndPassword(auth, signUpEmail, signUpPassword);

      // Display success message
      setAlertMessage({
        open: true,
        message: `Sign Up Successful. Welcome ${result.user.email}`,
        type: "success",
      });

      // Close the signup form
      handleClose();
    } catch (error) {
      // Display error message
      setAlertMessage({
        open: true,
        message: error.message,
        type: "error",
      });
      return;
    }
  };

  return (
    <Box p={3} style={{display: "flex",flexDirection: "column",gap: "19px",}}>
      {/* Email input */}
      <TextField
        variant="outlined"
        type="email"
        label="Enter Email"
        value={signUpEmail}
        onChange={(e) => setSignUpEmail(e.target.value)}
```

```
        fullWidth
      />
      {/* Password input */}
      <TextField
        variant="outlined"
        label="Enter Password"
        type="password"
        value={signUpPassword}
        onChange={(e) => setSignUpPassword(e.target.value)}
        fullWidth
      />
      {/* Confirm password input */}
      <TextField
        variant="outlined"
        label="Confirm Password"
        type="password"
        value={confirmSignUpPassword}
        onChange={(e) => setConfirmSignUpPassword(e.target.value)}
        fullWidth
      />
      {/* Submit button */}
      <Button
        variant="contained"
        size="large"
        style={{ backgroundColor: "#6d7be3" }}
        onClick={handleSignup}
      >
        Sign Up
      </Button>
    </Box>
  );
};

export default SignupForm;
```

Лістинг authentication/UserSideBar.js:

```
// Import necessary dependencies and components
import React from "react"
import {makeStyles} from "@material-ui/core/styles"
import Drawer from "@material-ui/core/Drawer"
import {Avatar, Button} from "@material-ui/core"
import {CryptoState} from "../../CryptoContext"
import {signOut} from "firebase/auth"
import {auth, db} from "../../firebase"
import {numberWithCommas} from "../CoinsTable"
import {AiFillDelete} from "react-icons/ai"
import {doc, setDoc} from "firebase/firestore"

// Define styles using Material-UI makeStyles hook
const useStyles = makeStyles({
  // Styling for the container of the sidebar
  sidebarContainer: {
    width: 349,
    height: "99%",
    padding: 24,
    display: "flex",
    flexDirection: "column",
    fontFamily: "monospace",
  },
  // Styling for the user profile section
  profileSection: {
    flex: 1,
```

```
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
    gap: "19px",
    height: "93%",
  },
  // Styling for the logout button
  logoutButton: {
    height: "7%",
    width: "99%",
    backgroundColor: "#6d7be3",
    marginTop: 19,
  },
  // Styling for the user's profile picture
  profilePicture: {
    width: 199,
    height: 199,
    cursor: "pointer",
    backgroundColor: "#6d7be3",
    objectFit: "contain",
  },
  // Styling for the watchlist section
  watchlistSection: {
    flex: 1,
    width: "99%",
    backgroundColor: "grey",
    borderRadius: 8,
    padding: 14,
    paddingTop: 8,
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
    gap: 13,
    overflowY: "scroll",
  },
  cryptoCoin: {
    padding: 8,
    borderRadius: 4,
    color: "black",
    width: "100%",
    display: "flex",
    justifyContent: "space-between",
    alignItems: "center",
    backgroundColor: "#6d7be3",
    boxShadow: "0 0 4px black",
  },
});

export default function UserSidebar() {
  const classes = useStyles();
  const [drawerState, setDrawerState] = React.useState({
    right: false,
  });
  const { user, setAlertMessage, watchlist, cryptoCoins, cryptoSymbol } = CryptoState();

  // Toggle the sidebar drawer open/closed
  const toggleDrawer = (anchor, open) => (event) => {
    if (
      event.type === "keydown" &&
      (event.key === "Tab" || event.key === "Shift")
    ) {
      return;
```

```
  }

  setDrawerState({ ...drawerState, [anchor]: open });
};

// Handle the logout functionality
const handleLogout = () => {
  signOut(auth);
  setAlertMessage({
    open: true,
    type: "success",
    message: "Logout Successfull !",
  });

  toggleDrawer();
};

// Remove a coin from the user's watchlist
const removeCoinFromWatchlist = async (coin) => {
  const coinRef = doc(db, "watchlist", user.uid);
  try {
    await setDoc(
      coinRef,
      { coins: watchlist.filter((wish) => wish !== coin?.id) },
      { merge: true }
    );

    setAlertMessage({
      open: true,
      message: `${coin.name} Removed from the Watchlist !`,
      type: "success",
    });
  } catch (error) {
    setAlertMessage({
      open: true,
      message: error.message,
      type: "error",
    });
  }
};

return (
  <div>
    {["right"].map((anchor) => (
      <React.Fragment key={anchor}>
        <Avatar
          onClick={toggleDrawer(anchor, true)}
          style={{height: 38,width: 38,marginLeft: 15,cursor: "pointer",backgroundColor: "#6d7be3",}}
          src={user.photoURL}
          alt={user.displayName || user.email}
        />
        <Drawer
          anchor={anchor}
          open={drawerState[anchor]}
          onClose={toggleDrawer(anchor, false)}
        >
          <div className={classes.sidebarContainer}>
            <div className={classes.profileSection}>
              <Avatar
                className={classes.profilePicture}
                src={user.photoURL}
                alt={user.displayName || user.email}
```

```
            />
            <span
              style={{
                width: "100%",
                fontSize: 25,
                textAlign: "center",
                fontWeight: "bolder",
                wordWrap: "break-word",
              }}
            >
              {user.displayName || user.email}
            </span>
            <div classNameName={classes.watchlistSection}>
              <span style={{ fontSize: 15, textShadow: "0 0 5px black" }}>
                Watchlist
              </span>
              {/* Display coins in the user's watchlist */}
              {cryptoCoins.map((coin) => {
                if (watchlist.includes(coin.id))
                  return (
                    <div classNameName={classes.cryptoCoin}>
                      <span>{coin.name}</span>
                      <span style={{ display: "flex", gap: 8 }}>
                        {cryptoSymbol}{" "}
                        {numberWithCommas(coin.current_price.toFixed(2))}
                        <AiFillDelete
                          style={{ cursor: "pointer" }}
                          fontSize="16"
                          onClick={() => removeCoinFromWatchlist(coin)}
                        />
                      </span>
                    </div>
                  );
                else return <></>;
              })}
            </div>
          </div>
          {/* Logout button */}
          <Button
            variant="contained"
            classNameName={classes.logoutButton}
            onClick={handleLogout}
          >
            Log Out
          </Button>
        </div>
      </Drawer>
    </React.Fragment>
  ))}
 </div>
 );
}
```

Лістинг banner /Banner.js:

```
// Import necessary dependencies and components
import { Container, makeStyles, Typography } from "@material-ui/core";
import Carousel from "./Carousel";

// Define styles using Material-UI makeStyles hook
const useStyles = makeStyles((theme) => ({
  banner: {
    backgroundImage: "url(./banner.jpg)",
  },
  bannerContent: {
    height: 399,
    display: "flex",
    flexDirection: "column",
    paddingTop: 24,
    justifyContent: "space-around",
  },
  slogan: {
    display: "flex",
    height: "39%",
    flexDirection: "column",
    justifyContent: "center",
    textAlign: "center",
  },
  carousel: {
    height: "49%",
    display: "flex",
    alignItems: "center",
  },
}));

function Banner() {
  const classes = useStyles();

  return (
    <div className={classes.banner}>
      {/* Container to hold the banner content */}
      <Container className={classes.bannerContent}>
        {/* Tagline section */}
        <div className={classes.slogan}>
          <Typography
            variant="h2"
            style={{
              fontWeight: "bold",
              marginBottom: 15,
              fontFamily: "Montserrat",
            }}
          >
            Crypto Tracker
          </Typography>
          <Typography
            variant="subtitle2"
            style={{
              color: "darkgrey",
              textTransform: "capitalize",
              fontFamily: "Montserrat",
            }}
          >
            Obtain complete information about the cryptocurrency of your choice.
          </Typography>
        </div>
        {/* Carousel section */}
```

```
        <Carousel />
      </Container>
    </div>
  );
}

export default Banner;
```

Лістинг banner/Carousel.js:

```
// Import necessary dependencies and components
import { makeStyles } from "@material-ui/core";
import axios from "axios";
import { useEffect, useState } from "react";
import AliceCarousel from "react-alice-carousel";
import { Link } from "react-router-dom";
import { TrendingCoins } from "../../config/api";
import { CryptoState } from "../../CryptoContext";
import { numberWithCommas } from "../CoinsTable";

const Carousel = () => {
  const [trendingCoins, setTrendingCoins] = useState([]);
  const { currency, symbol } = CryptoState();

  // Fetch trending coins data from the API
  const fetchTrendingCoinsData = async () => {
    const { data } = await axios.get(TrendingCoins(currency));

    console.log(data);
    setTrendingCoins(data);
  };

  useEffect(() => {
    // Fetch trending coins when the currency value changes
    fetchTrendingCoinsData();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [currency]);

  const useStyles = makeStyles((theme) => ({
    carouselContainer: {
      height: "50%",
      display: "flex",
      alignItems: "center",
    },
    carouselItem: {
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      cursor: "pointer",
      textTransform: "uppercase",
      color: "white",
    },
  }));

  const classes = useStyles();

  // Generate carousel items based on trending coins data
  const carouselItems = trendingCoins.map((coin) => {
    let isProfitable = coin?.price_change_percentage_24h >= 0;

    return (
      <Link className={classes.carouselItem} to={`/coins/${coin.id}`}>
        <img
```

```
          src={coin?.image}
          alt={coin.name}
          height="80"
          style={{ marginBottom: 10 }}
        />
        <span>
          {coin?.symbol}
           
          <span
            style={{
              color: isProfitable > 0 ? "rgb(14, 203, 129)" : "red",
              fontWeight: 500,
            }}
          >
            {isProfitable && "+"}
            {coin?.price_change_percentage_24h?.toFixed(2)}%
          </span>
        </span>
        <span style={{ fontSize: 22, fontWeight: 500 }}>
          {symbol} {numberWithCommas(coin?.current_price.toFixed(2))}
        </span>
      </Link>
    );
  });

  const responsiveConfig = {
    0: {
      items: 2,
    },
    512: {
      items: 4,
    },
  };

  return (
    <div className={classes.carouselContainer}>
      <AliceCarousel
        mouseTracking
        infinite
        autoPlayInterval={1000}
        animationDuration={1500}
        disableDotsControls
        disableButtonsControls
        responsive={responsiveConfig}
        items={carouselItems}
        autoPlay
      />
    </div>
  );
};

export default Carousel;
```

Лістинг CoinInformation.js:

```
// Import necessary dependencies and components
import axios from "axios";
import { useEffect, useState } from "react";
import { HistoricalChart } from "../config/api";
import { Line } from "react-chartjs-2";
import {CircularProgress, createTheme, makeStyles, ThemeProvider,} from "@material-ui/core";
import SelectButton from "./SelectButton";
import { chartDays } from "../config/data";
import { CryptoState } from "../CryptoContext";
import {Chart, LineController, LineElement, PointElement, LinearScale, Title, CategoryScale} from 'chart.js';
// Register necessary Chart.js components
Chart.register (LineController, LineElement, PointElement, LinearScale, Title, CategoryScale);

const CoinInformation = ({ coin }) => {
  const [historicData, setHistoricData] = useState();
  const [selectedDays, setSelectedDays] = useState(1);
  const {cryptoCurrency} = CryptoState();
  const [isLoading, setIsLoading] = useState(false);

  const useStyles = makeStyles((theme) => ({
    coinInfoContainer: {
      width: "74%",
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      justifyContent: "center",
      marginTop: 24,
      padding: 39,
      [theme.breakpoints.down("md")]: {
        width: "99%",
        marginTop: 0,
        padding: 19,
        paddingTop: 0,
      },
    },
  }));

  const classes = useStyles();

  const fetchHistoricData = async () => {
    const { data } = await axios.get(HistoricalChart(coin.id, selectedDays, cryptoCurrency));
    setIsLoading(true);
    setHistoricData(data.prices);
  };

  useEffect(() => {
    fetchHistoricData();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [selectedDays]);

  const darkTheme = createTheme({
    palette: {
      primary: {
        main: "#fff",
      },
      type: "dark",
    },
  });

  return (
    <ThemeProvider theme={darkTheme}>
```

```
    <div className={classes.coinInfoContainer}>
      {!historicData | isLoading===false ? (
        // Display a circular progress indicator while data is being fetched
        <CircularProgress
          style={{ color: "#3535d4" }}
          size={250}
          thickness={1}
        />
      ) : (
        <>
          {/* Render the line chart */}
          <Line
            data={{
              labels: historicData.map((coin) => {
                let date = new Date(coin[0]);
                let time =
                  date.getHours() > 12
                    ? `${date.getHours() - 12}:${date.getMinutes()} PM`
                    : `${date.getHours()}:${date.getMinutes()} AM`;
                return selectedDays === 1 ? time : date.toLocaleDateString();
              }),

              datasets: [
                {
                  data: historicData.map((coin) => coin[1]),
                  label: `Price ( Past ${selectedDays} Days ) in ${cryptoCurrency}`,
                  borderColor: "#3535d4",
                },
              ],
            }}
          />
          {/* Render the select buttons for choosing the time range */}
          <div
            style={{
              display: "flex",
              marginTop: 20,
              justifyContent: "space-around",
              width: "100%",
            }}
          >
            {chartDays.map((day) => (
              <SelectButton
                key={day.value}
                onClick={() => {setSelectedDays(day.value);
                  setIsLoading(false);
                }}
                selected={day.value === day}
              >
                {day.label}
              </SelectButton>
            ))}
          </div>
        </>
      )}
    </div>
  </ThemeProvider>
 );
};

export default CoinInformation;
```

Лістинг CoinsTable.js:

```
// Import necessary dependencies and components
import React, { useEffect, useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import Pagination from "@material-ui/lab/Pagination";
import
{Container,createTheme,TableCell,LinearProgress,ThemeProvider,Typography,TextField,TableBody,TableRow,
  TableHead,TableContainer,Table,Paper,} from "@material-ui/core";
import { useNavigate } from "react-router-dom";
import { CryptoState } from "../CryptoContext";

// Helper function to format numbers with commas
export function numberWithCommas(x) {
  return x.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",");
}

export default function CoinsTable() {
  const [searchInput, setSearchInput] = useState("");
  const [coinsPage, setCoinsPage] = useState(1);

  const { currency, symbol, coins, loading, fetchCoins } = CryptoState();

  const useStyles = makeStyles({
    row: {
      backgroundColor: "#16171a",
      cursor: "pointer",
      "&:hover": {
        backgroundColor: "#131111",
      },
      fontFamily: "Montserrat",
    },
    pagination: {
      "& .MuiPaginationItem-root": {
        color: "#5c6ad1",
      },
    },
  });

  const classes = useStyles();
  const navigate = useNavigate();

  const darkTheme = createTheme({
    palette: {
      primary: {
        main: "#fff",
      },
      type: "dark",
    },
  });

  useEffect(() => {
    fetchCoins();
    // eslint-disable-next-line react-hooks/exhaustive-deps
  }, [currency]);

  // Filter the coins based on the search input
  const filterCoins = () => {
    return coins.filter(
      (coin) =>
        coin.name.toLowerCase().includes(searchInput) ||
        coin.symbol.toLowerCase().includes(searchInput)
    );
```

```
};

return (
  <ThemeProvider theme={darkTheme}>
    <Container style={{ textAlign: "center" }}>
      <Typography
        variant="h4"
        style={{ margin: 18, fontFamily: "Montserrat" }}
      >
        Cryptocurrency Prices by Market Cap
      </Typography>
      <TextField
        label="Search For a Crypto Currency.."
        variant="outlined"
        style={{ marginBottom: 19, width: "99%" }}
        onChange={(e) => setSearchInput(e.target.value)}
      />
      <TableContainer component={Paper}>
        {loading ? (
          <LinearProgress style={{ backgroundColor: "#5c6ad1" }} />
        ) : (
          <Table aria-label="simple table">
            <TableHead style={{ backgroundColor: "#5c6ad1" }}>
              <TableRow>
                {/* Render table headers */}
                {["Coin", "Price", "24h Change", "Market Cap"].map((head) => (
                  <TableCell
                    style={{
                      color: "black",
                      fontWeight: "699",
                      fontFamily: "Montserrat",
                    }}
                    key={head}
                    align={head === "Coin" ? "" : "right"}
                  >
                    {head}
                  </TableCell>
                ))}
              </TableRow>
            </TableHead>

            <TableBody>
              {/* Render table rows */}
              {filterCoins()
                .slice((coinsPage - 1) * 10, (coinsPage - 1) * 10 + 10)
                .map((coin) => {
                  const profit = coin.price_change_percentage_24h > 0;
                  return (
                    <TableRow
                      onClick={() => navigate(`/coins/${coin.id}`)}
                      className={classes.row}
                      key={coin.name}
                    >
                      {/* Coin details */}
                      <TableCell
                        component="th"
                        scope="row"
                        style={{
                          display: "flex",
                          gap: 14,
                        }}
                      >
```

```
              <img
                src={coin?.image}
                alt={coin.name}
                height="50"
                style={{ marginBottom: 9 }}
              />
              <div
                style={{ display: "flex", flexDirection: "column" }}
              >
                <span
                  style={{
                    textTransform: "uppercase",
                    fontSize: 22,
                  }}
                >
                  {coin.symbol}
                </span>
                <span style={{ color: "darkgrey" }}>
                  {coin.name}
                </span>
              </div>
            </TableCell>
            {/* Price */}
            <TableCell align="right">
              {symbol}{" "}
              {numberWithCommas(coin.current_price.toFixed(2))}
            </TableCell>
            {/* 24-hour change */}
            <TableCell
              align="right"
              style={{
                color: profit > 0 ? "rgb(14, 203, 129)" : "red",
                fontWeight: 500,
              }}
            >
              {profit && "+"}
              {coin.price_change_percentage_24h.toFixed(2)}%
            </TableCell>
            {/* Market cap */}
            <TableCell align="right">
              {symbol}{" "}
              {numberWithCommas(
                coin.market_cap.toString().slice(0, -6)
              )}
              M
            </TableCell>
          </TableRow>
        );
      })}
    </TableBody>
  </Table>
 )}
</TableContainer>

{/* Pagination */}
<Pagination
 count={(filterCoins()?.length / 10).toFixed(0)}
 style={{
   padding: 19,
   width: "99%",
   display: "flex",
   justifyContent: "center",
```

```
        }}
        classes={{ ul: classes.pagination }}
        onChange={(_, value) => {
          setCoinsPage(value);
          window.scroll(0, 449);
        }}
      />
    </Container>
  </ThemeProvider>
 );
}
```

Лістинг CustomAlert.js:

```
// Import necessary dependencies and components
import { Snackbar } from "@material-ui/core";
import MuiAlert from "@material-ui/lab/Alert";
import { CryptoState } from "../CryptoContext";

// Alert component displays a Snackbar with a message and type based on the alert state
const CustomAlert = () => {
 const { alert, setAlertMessage } = CryptoState();

 // Function to handle closing the alert Snackbar
 const handleCloseCustomAlert = (event, reason) => {
  if (reason === "clickaway") {
   return;
  }

  // Close the alert by updating the alert state
  setAlertMessage({ open: false });
 };

 return (
  <Snackbar
   open={alert.open}
   autoHideDuration={2999}
   onClose={handleCloseCustomAlert}
  >
   {/* Snackbar content with the provided alert message and type */}
   <MuiAlert
    onClose={handleCloseCustomAlert}
    elevation={9}
    variant="filled"
    severity={alert.type}
   >
    {alert.message}
   </MuiAlert>
  </Snackbar>
 );
};

export default CustomAlert;
```

Лістинг Header.js:

```
// Import necessary dependencies and components
import {AppBar,Container,MenuItem,Select,Toolbar,Typography,} from "@material-ui/core";
import {createTheme,makeStyles,ThemeProvider,} from "@material-ui/core/styles";
import { useNavigate } from "react-router-dom";
import { CryptoState } from "../CryptoContext";
import AuthModal from "./authentication/AuthenticationModal";
import UserSidebar from "./authentication/UserSideBar";
```

```
// Custom styles for the header
const useStyles = makeStyles((theme) => ({
  headerTitle: {
    flex: 1,
    color: "#3535d4",
    fontFamily: "Montserrat",
    fontWeight: "bold",
    cursor: "pointer",
  },
}));

// Dark theme configuration
const darkTheme = createTheme({
  palette: {
    primary: {
      main: "#fff",
    },
    type: "dark",
  },
});

function Header() {
  const classes = useStyles();
  const { cryptoCurrency, setCryptoCurrency, user } = CryptoState();

  const navigate = useNavigate();

  return (
    <ThemeProvider theme={darkTheme}>
      <AppBar color="transparent" position="static">
        <Container>
          <Toolbar>
            {/* Title */}
            <Typography
              onClick={() => navigate(`/`)}
              variant="h6"
              className={classes.headerTitle}
            >
              Crypto Tracker
            </Typography>
            {/* Currency selection */}
            <Select
              variant="outlined"
              labelId="demo-simple-select-label"
              id="demo-simple-select"
              value={cryptoCurrency}
              style={{ width: 99, height: 39, marginLeft: 14 }}
              onChange={(e) => setCryptoCurrency(e.target.value)}
            >
              <MenuItem value={'USD'}>USD</MenuItem>
              <MenuItem value={'UAH'}>UAH</MenuItem>
              <MenuItem value={'EUR'}>EUR</MenuItem>
            </Select>

            {/* Render UserSidebar component if user is logged in, otherwise render AuthModal component */}
            {user ? <UserSidebar/> : <AuthModal/>}
          </Toolbar>
        </Container>
      </AppBar>
    </ThemeProvider>
  );
```

```
}

export default Header;
```

Лістинг firebaseConfig.js:

```
const firebaseConfig = {
    apiKey: "AIzaSyBqwaFzm9_FJ4WV1DFIjhJTitPUVWX_Cuc",
    authDomain: "crypt-tracker.firebaseapp.com",
    projectId: "crypt-tracker",
    storageBucket: "crypt-tracker.appspot.com",
    messagingSenderId: "353586689080",
    appId: "1:353586689080:web:76b64d85083f45fa5ee2cd"
  };

  export default firebaseConfig;
```

Лістинг CoinPage.js:

```
// Import necessary dependencies and components
import {Button,LinearProgress,makeStyles,Typography,} from "@material-ui/core";
import axios from "axios";
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import ReactHtmlParser from "react-html-parser";
import CoinInfo from "../components/CoinInformation";
import { SingleCoin } from "../config/api";
import { numberWithCommas } from "../components/CoinsTable";
import { CryptoState } from "../CryptoContext";
import { doc, setDoc } from "firebase/firestore";
import { db } from "../firebase";

const CoinPage = () => {
  // Fetching the coin ID from the URL parameters
  const {coinId} = useParams();
  // State to store the fetched coin data
  const [cryptoCoin, setCryptoCoin] = useState();

  // Retrieving data from CryptoState context
  const { cryptoCurrency, cryptoSymbol, user, setAlertMessage, cryptoWatchlist } = CryptoState();

  // Function to fetch coin data from the API
  const fetchCoin = async () => {
    const { data } = await axios.get(SingleCoin(coinId));

    setCryptoCoin(data);
  };

  // Checking if the coin is in the watchlist
  const inWatchlist = cryptoWatchlist.includes(cryptoCoin?.id);

  // Function to add the coin to the watchlist
  const addToWatchlist = async () => {
    const coinRef = doc(db, "watchlist", user.uid);
    try {
      await setDoc(
        coinRef,
        { coins: cryptoWatchlist ? [...cryptoWatchlist, cryptoCoin?.id] : [cryptoCoin?.id] },
        { merge: true }
      );

      setAlertMessage({
        open: true,
        message: `${cryptoCoin.name} Added to the Watchlist !`,
```

```
      type: "success",
    });
  } catch (error) {
    setAlertMessage({
      open: true,
      message: error.message,
      type: "error",
    });
  }
};

// Function to remove the coin from the watchlist
const removeFromWatchlist = async () => {
  const coinRef = doc(db, "watchlist", user.uid);
  try {
    await setDoc(
      coinRef,
      { coins: cryptoWatchlist.filter((wish) => wish !== cryptoCoin?.id) },
      { merge: true }
    );

    setAlertMessage({
      open: true,
      message: `${cryptoCoin.name} Removed from the Watchlist !`,
      type: "success",
    });
  } catch (error) {
    setAlertMessage({
      open: true,
      message: error.message,
      type: "error",
    });
  }
};

// Fetching the coin data on component mount
useEffect(() => {
  fetchCoin();
  // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

const useStyles = makeStyles((theme) => ({
  // Styling classes for different elements
  coinPageContainer: {
    display: "flex",
    [theme.breakpoints.down("md")]: {
      flexDirection: "column",
      alignItems: "center",
    },
  },
  coinPageSidebar: {
    width: "29%",
    [theme.breakpoints.down("md")]: {
      width: "99%",
    },
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
    marginTop: 24,
    borderRight: "2px solid grey",
  },
  coinPageHeading: {
```

```
    fontWeight: "bold",
    marginBottom: 19,
    fontFamily: "Montserrat",
  },
  coinDescription: {
    width: "99%",
    fontFamily: "Montserrat",
    padding: 24,
    paddingBottom: 14,
    paddingTop: 0,
    textAlign: "justify",
  },
  cryptoMarketData: {
    alignSelf: "start",
    padding: 24,
    paddingTop: 9,
    width: "99%",
    [theme.breakpoints.down("md")]: {
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
    },
    [theme.breakpoints.down("xs")]: {
      alignItems: "start",
    },
  },
}));

const classes = useStyles();

// Render a loading state until the coin data is fetched
if (!cryptoCoin) return <LinearProgress style={{ backgroundColor: "white" }} />;

return (
  <div className={classes.coinPageContainer}>
    <div className={classes.coinPageSidebar}>
      {/* Displaying coin image, name, and description */}
      <img
        src={cryptoCoin?.image.large}
        alt={cryptoCoin?.name}
        height="200"
        style={{ marginBottom: 19 }}
      />
      <Typography variant="h3" className={classes.coinPageHeading}>
        {cryptoCoin?.name}
      </Typography>
      <Typography variant="subtitle1" className={classes.coinDescription}>
        {/* Displaying a parsed version of the coin description */}
        {ReactHtmlParser(cryptoCoin?.description.en.split(". ")[0])}.
      </Typography>
      <div className={classes.cryptoMarketData}>
        {/* Displaying coin market data */}
        <span style={{ display: "flex" }}>
          <Typography variant="h5" className={classes.coinPageHeading}>
            Rank:
          </Typography>
             
          <Typography
            variant="h5"
            style={{
              fontFamily: "Montserrat",
            }}
```

```
      >
        {/* Formatting and displaying the coin's market cap rank */}
        {numberWithCommas(cryptoCoin?.market_cap_rank)}
      </Typography>
    </span>
    <span style={{ display: "flex" }}>
      <Typography variant="h5" className={classes.coinPageHeading}>
        Current Price:
      </Typography>
         
      <Typography
        variant="h5"
        style={{
          fontFamily: "Montserrat",
        }}
      >
        {/* Formatting and displaying the coin's current price */}
        {cryptoSymbol}{" "}
        {numberWithCommas(
          cryptoCoin?.market_data.current_price[cryptoCurrency.toLowerCase()]
        )}
      </Typography>
    </span>
    <span style={{ display: "flex" }}>
      <Typography variant="h5" className={classes.coinPageHeading}>
        Market Cap:
      </Typography>
         
      <Typography
        variant="h5"
        style={{
          fontFamily: "Montserrat",
        }}
      >
        {/* Formatting and displaying the coin's market cap */}
        {cryptoSymbol}{" "}
        {numberWithCommas(
          cryptoCoin?.market_data.market_cap[cryptoCurrency.toLowerCase()]
            .toString()
            .slice(0, -6)
        )}
        M
      </Typography>
    </span>
    {/* Render the watchlist button if the user is logged in */}
    {user && (
      <Button
        variant="outlined"
        style={{
          width: "99%",
          height: 39,
          backgroundColor: inWatchlist ? "#ad2a2a" : "#6d7be3",
        }}
        onClick={inWatchlist ? removeFromWatchlist : addToWatchlist}
      >
        {inWatchlist ? "Remove from Watchlist" : "Add to Watchlist"}
      </Button>
    )}
  </div>
</div>
```

```
 {/* Render the coin information component */}
     <CoinInfo coin={cryptoCoin} />
   </div>
 );
};

export default CoinPage;
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна_робота_ Назаркін.docx	Пояснювальна записка до дипломного проекту. Документ Word.
Кваліфікаційна_робота_ Назаркін.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Web-tracker.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_ Назаркін.ppt	Презентація дипломного проекту