Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система моніторингу та візуалізації вартості криптовалют*

Виконав: студент *IV* курсу, групи *СІ-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Демиденко А.О.
	(підпис)	(прізвище та ініціали)
Керівник		*Яцишин В.В.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є.В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		*Мацюк О.В.*
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Демиденку Антону Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система моніторингу та візуалізації вартості криптовалют*

Керівник роботи *Яцишин Василь Володимирович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *лютого* 2023 року № *4/7-238*

2. Термін подання студентом завершеної роботи *23.06.202 р.*

3. Вихідні дані до роботи *Дані про криптовалюти, їхня вартість, новини про трейдинг та види критовалютної торгівлі*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз сучасного стану досліджень у сфері аналізу криптовалюти автомобілів. 2. Аналіз технологій розробки веб-додатків та мов програмування 3. Реалізація програмного забезпечення комп'ютеризованої системи моніторингу та візуалізації вартості криптовалют. 4. Безпека життєдіяльності, основи охорони праці. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Пилипець М.І., д.т.н., проф. каф. МТ*		
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка та аналіз вимог технічного завдання*	*28.02-19.02.2023*	*виконано*
2	*Аналіз сучасного стану досліджень у сфері аналізу криптовалюти*	*19.02-05.03.2023*	*виконано*
3	*Аналіз технологій розробки веб-додатків та мов програмування*	*05.03-26.03.2023*	*виконано*
4	*Обґрунтування вибору апаратного забезпечення*	*26.03-01.04.2023*	*виконано*
5	*Реалізація програмного забезпечення комп'ютеризованої системи моніторингу та візуалізації вартості криптовалют*	*01.04-10.05.2023*	*виконано*
6	*Безпека життєдіяльності, основи охорони праці*	*10.05-18.05.2023*	*виконано*
7	*Оформлення кваліфікаційної роботи*	*18.05-06.06.2023*	*виконано*
8	*Попередній захист кваліфікаційної роботи*	*06.06-18.06.2023*	*виконано*
9	*Захист кваліфікаційної роботи*	*21.06-23.06.2023*	

Студент _____ *Демиденко Антон Олександрович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Яцишин Василь Володимирович*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп'ютеризована система моніторингу та візуалізації вартості криптовалют//Кваліфікаційна робота на здобуття освітнього ступеня бакалавр// Антон Демиденко //ТНТУ імені Пулюя, спеціальність 123 комп’ютерна інженерія // Тернопіль, 2023//с.- 72, рис. - 23, аркушів А1 - 4, бібліогр. - 19.

Ключові слова: комп'ютерна система, візуалізація, моніторинг, ціни, криптовалюти.

У кваліфікаційній роботі бакалавра розроблена комп'ютерна система візуалізації та моніторингу рівня цін криптовалют. Здійснений аналіз предметної області, в результаті чого побудовано графік даних та сформульовано функціональні вимоги до системи, що візуалізовані за допомогою діаграм варіантів використання. Для зберігання та обробки даних використана платформа для розробки мобільних та веб застосунків - Firebase. У роботі спроектовано архітектуру програмного забезпечення на тему комп'ютеризована система моніторингу та візуалізації вартості криптовалют. Реалізацію програмної складової комп'ютеризованої системи виконано за допомогою мови програмування JavaScript та TypeScript з використання фреймворку Vue 3, та серидовищем розробки було обрано WebStorm.

ANNOTATION

Computerized system for monitoring and visualization of cryptocurrency prices//Bachelor's degree qualification work//Anton Demidenko//Ternopil Ivan Pul'uj National Technical University, specialty 123 Computer Engineering//Ternopil, 2023//pp. - 72, fig. -23, sheets A1 - 4, bibl. - 19.

Keywords: computer system, visualization, monitoring, prices, cryptocurrencies.

In the bachelor's thesis, a computer system for visualization and monitoring of cryptocurrency prices has been developed. The domain analysis has been carried out, resulting in the construction of a data graph and formulation of functional requirements for the system, visualized using use case diagrams. Firebase, a platform for mobile and web application development, has been used for data storage and processing. The software architecture for the computerized monitoring and visualization system for cryptocurrency values has been designed. The implementation of the software component has been done using JavaScript and TypeScript programming languages, along with the Vue 3 framework, and the WebStorm development environment has been chosen.

ЗМІСТ

ВСТУП

У сучасному світі, де Інтернет надає безліч можливостей, електронна комерція постійно розвивається, а разом з нею зростає і сфера криптовалют. Інвестиційні фонди та трейдери активно займаються торгівлею на різних криптовалютних біржах, використовуючи високу волатильність цих цифрових активів для отримання прибутку. Однак, для досягнення стабільного доходу трейдерам потрібно мати чітке розуміння цін на криптовалюти. Тому прогнозування курсу крпитовалют є важливим завданням для професіоналів у галузі трейдингу.

Зважаючи на високі ризики, пов'язані з торгівлею криптовалютами, моніторинг-платформа стає незамінним програмним забезпеченням, яка допомагає трейдерам аналізувати ринок більш ефективно та оперативно. На сьогоднішній день ринок криптовалют має унікальну особливість - сильну залежність курсу від новин, які поширюються через популярні соціальні мережі та новинні веб-сайти. Враховуючи це, розробка системи у вигляді веб-додатку, спрямованого на візуалізацію та моніторинг рівня цін криптовалют є обґрунтованим дипломним проектом.

Дана система має на меті надати трейдерам зручний інструмент для аналізу та відстеження курсу криптовалют, що дозволить їм приймати обґрунтовані рішення. Крім того, вона забезпечує можливість моніторингу новин, що допомагає зрозуміти вплив факторів на курс криптовалютних активів. Даний проект присвячено розробці веб-додатку, призначеної для візуалізації та моніторингу рівня цін криптовалют.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕНЬ У СФЕРІ АНАЛІЗУ КРИПТОВАЛЮТИ

1.1 Аналіз особливостей ринку криптовалют

Завдяки постійному розвитку Інтернету та електронної комерції, аналіз криптовалютних даних стає все більш важливим. В царині криптовалют, дані включають інформацію про ціни та обсяги торгів, а також про проекти та команди, які стоять за криптоактивами. Крім того, значення має також місце проведення торгів, наприклад, криптовалютну біржу.

Сучасні трейдери активно використовують онлайн-сервіси криптовалютних бірж, таких як Binance, OKX, DOBI Exchange, для здійснення торгівлі на фінансових ринках, приклад біржі можна побачити на рис. 1.1. Цей спосіб торгівлі має декілька переваг, таких як низькі витрати на комісію для трейдерів, можливість заробити на високій ліквідності та використання автоматизованої машинної торгівлі за власним алгоритмом.

Криптовалютні ринкові дані охоплюють потокову інформацію, пов'язану з торгівлею. Вони включають ціни, котирування пропозицій і запитів та обсяги ринку. Торгові платформи надають звіти про різні активи та фінансові інструменти, які потім розповсюджуються серед трейдерів і фірм. Ринкові дані допомагають трейдерам оцінювати вартість різних криптовалют та приймати рішення щодо входу та виходу з торгів. Використання цих даних спрямоване на отримання якомога більшої кількості інформації про актив, з яким планується торгувати, для розрахунку ринкового ризику та оцінки впливу новин на ринок.

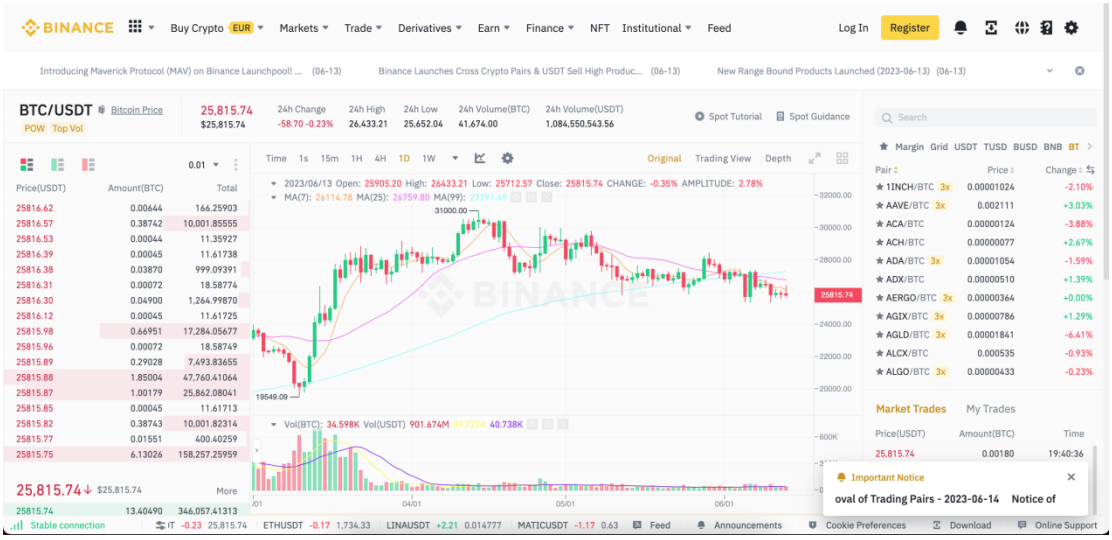

Рисунок 1.1 – Криптовалютна біржа Binance

Ринкові дані доступні в режимі реального часу, що дозволяє трейдерам приймати швидкі та обґрунтовані рішення. Крім того, історичні дані про ринок, включаючи графіки, є важливими для технічного аналізу та розробки стратегій торгівлі.

1.2 Аналіз криптовалют та токенів

Криптовалюта є цифровим активом з різними варіантами використання. Вона дозволяє передавати цінність між людьми у цифровому форматі, включаючи грошову цінність, право власності або голосування. Основною особливістю криптовалюти є її базування на блокчейн-технології та автоматизованій платіжній системі без необхідності централізованого управління.

На сьогоднішній день доступно понад дві тисячі криптовалют для торгівлі, серед яких найбільш відомі Bitcoin, Ether, Ripple, Bitcoin Cash і Litecoin. Станом на травень 2023 року загальна капіталізація цих криптовалют становила $1.15 трильйона доларів США, а середній обсяг торгівлі за 24 години склав $36.92 мільярди. Обсяг усіх операцій в децентралізованих фінансах (DeFi) наразі становить $2.60 мільярди, а обсяг стейблкоїнів склав $33.86 мільярди.

Зараз кожен має можливість створити власний токен, навіть без програмування, завдяки наявності різноманітних блокчейнів, таких як Binance Smart Chain, Polygon, Solana та Ethereum. Створення токена не вимагає великого досвіду і зусиль порівняно з створенням криптовалюти. Для створення власної монети потрібна компанія програмістів, але для створення токена можна використати інші блокчейни і зробити це за лічені хвилини.

Криптовалюти можна умовно поділити на дві категорії: монети і токени. Монети мають власний блокчейн, прикладом може бути – Bitcoin або Ether і Ethereum. Вони виконують багато функції у мережі, такі як оплата комісій, стейкінг або участь у голосуванні.

Блокчейн – це розподілена технологія, що дозволяє зберігати та передавати інформацію в безпечному та незмінному способі. Він базується на концепції ланцюжка блоків, де кожен блок містить дані та посилання на попередній блок, утворюючи таким чином послідовну структуру даних. Основна особливість блокчейну полягає в тому, що він працює на основі розподіленої мережі комп'ютерів, які називаються вузлами. Кожен вузол має копію повної історії транзакцій, які здійснюються в мережі, як показано на рис. 1.2. Це означає, що немає центрального сервера чи організації, яка контролює та володіє данними. Блокчейн забезпечує безпеку та незмінність даних за допомогою криптографічних алгоритмів. Кожен блок має свій унікальний ідентифікатор (хеш), який обчислюється на основі даних у блоку та хешу попереднього блоку. Це дозволяє виявити будь-які зміни в блоках, оскільки будь-яка зміна в одному блоку автоматично змінює хеш наступних блоків, що робить маніпуляції з даними майже неможливими.

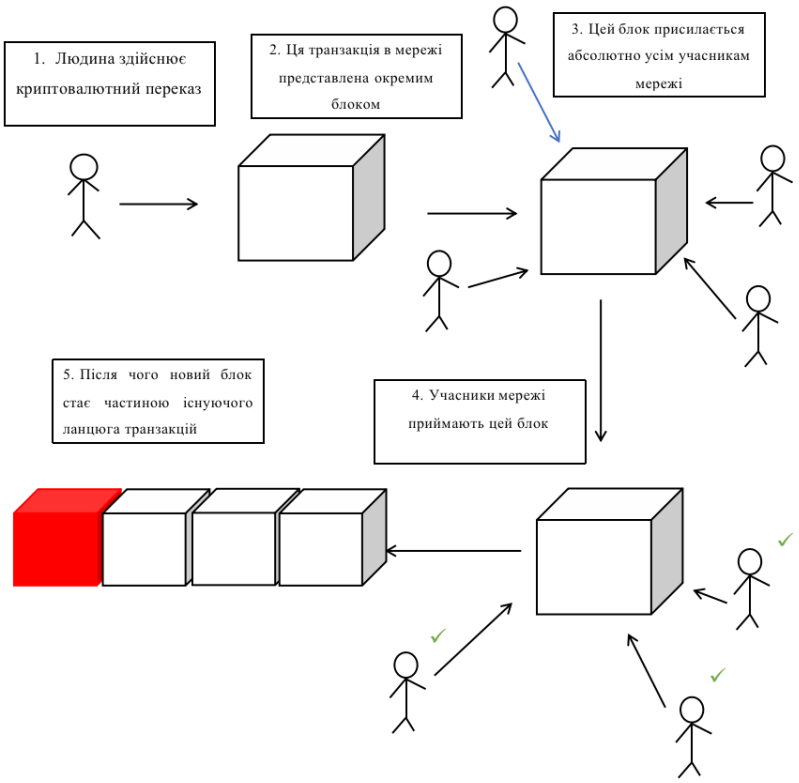

Рисунок 1.2 – Принцип блокчейну

Токени, натомість, створені на вже існуючих блокчейнах і використовуються переважно в межах своїх власних проєктів. Один з прикладів такого токена - CAKE від компанії PancakeSwap, який базується на блокчейні Binance Smart Chain.

Створення власного токена стало простішим завдяки наявності різних блокчейнів, таких як Ethereum, Binance Smart Chain. Різниця між монетами і токенами полягає в тому, що монети мають свій власний блокчейн, наприклад, Bitcoin або Ethereum.

1.3 Методи аналізу маркетингових даних

Інтерес до криптовалют обумовлений, можливістю вигідного переказу валюти по світу та спекулятивними чинниками. Хоча у більшості країн світу

криптовалюти ще не мають правового статусу, ці цифрові активи широко використовуються для проведення трейдингових спекулятивних операцій.

В Україні за 2022 рік спостерігався значний ріст прийняття криптовалют населенням. Згідно з дослідженням компанії Tripple A, відсоток громадян, які володіють криптовалютами в Україні, збільшився на понад 2% за рік. На початок 2022 року криптоактиви володіли 15,72% населення України.

Криптотрейдинг, або торгівля криптовалютами, є процесом купівлі, продажу та обміну цифрових валют на криптовалютних біржах. Трейдер має можливість торгувати як за фіатні гроші (традиційні валюти, такі як долари або євро), так і обмінювати одну криптовалюту на іншу.

У криптотрейдингу використовуються два основних підходи: фундаментальний аналіз та технічний аналіз. Технічний аналіз передбачає, що ціна відображає всю наявну інформацію і змінюється відповідно до тенденцій. Це означає, що шляхом аналізу історичних цін на криптовалюту можна передбачити її подальші рухи. Технічний аналіз включає в себе вивчення графіків цін наприклад, графіків японських свічок, виявлення рівнів підтримки та опору і прийняття рішень на основі цих даних.

Технічний аналіз криптовалют передбачає, що цінові рівні на графіку показують, за якими цінами покупці готові купувати, а продавці – продавати. Трейдер, успішно визначаючи ці рівні, може відкривати позиції вигідною напрямку, тобто купувати криптовалюту перед початком її зростання і продавати до зниження курсу. Цей аналіз базується на принципі, що історія циклічна, і спирається на аналіз психології трейдерів та виявлення закономірностей у руху цін. Аналіз криптовалют базується на трьох основних принципах:

– Історія повторюється: цей принцип виходить з припущення, що історичні дані про ціни криптовалют містять закономірності, які можуть повторюватись у майбутньому. Аналізуючи графіки та паттерни цінових рухів, трейдери намагаються передбачити майбутні зміни в цінах.

– Інформація в графіках: за цим принципом вважається, що всі фактори, які впливають на ціну криптовалюти, вже відображені у графіках. Технічні аналітики вивчають графіки цін та використовують інструменти, такі як індикатори та паттерни, щоб зрозуміти напрямок руху цін і прийняти торгові рішення.

– Трендовий рух: цей принцип стверджує, що ціна криптовалюти рухається в трендах або певних напрямках. Тренд може бути вищим (биковим), коли ціни зростають, нижнім (ведмежим), коли ціни падають, або бічним, коли ціни коливаються у певному діапазоні. Трейдери намагаються виявити поточний тренд і передбачити його зміну, щоб здійснити вигідні угоди.

Ці принципи технічного аналізу допомагають трейдерам розуміти цінові рухи на ринку криптовалют і приймати обґрунтовані торгові рішення.

Фундаментальний аналіз це метод аналізу, який використовується трейдерами для визначення внутрішньої вартості активів. Для точної оцінки цієї вартості вони ретельно досліджують усі фактори, щоб визначити чи є актив або бізнес переоціненим або недооціненим. Для визначення реальної ціни активів, фундаментальні аналітики в основному використовують бізнес-метрики, такі як показники дохідності, проектні показники та фінансові показники. Ці показники допомагають зрозуміти потенційні ризики та можливості, пов'язані з активами чи бізнесом.

Ончейн-метрики представляють собою характеристики, які можна отримати з даних блокчейну. Простим рішенням є збір інформації з сайтів або API. Наприклад, аналіз даних біткоїна на CoinMarketCap надає велику кількість інформації. Основні блокчейн-метрики включають:

– кількість транзакцій - показник вказує на активність мережі;

– вартість транзакції - цей показник показує сумарний обсяг транзакцій за певний період;

– активні адреси - це блокчейн-адреси, які були активними протягом встановленого періоду;

– комісії - розмір комісій може бути показником попиту на певний актив;

– хешрейт та кількість монет у стейкінгу - метрики допомагають зрозуміти стан мережі та витрати на майнінг.

Важливою відмінністю між технічним та фундаментальним аналізом є те, що фундаментальний аналіз призначений для пошуку довгострокових інвестиційних можливостей, тоді як технічний аналіз зазвичай фокусується на короткострокових коливаннях цін. Як фундаментальний, так і технічний аналіз можна зробити самостійно або спільно. Деякі аналітики використовують обидва методи аналізу, а інші дотримуються лише одного.

РОЗДІЛ 2 АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ - ДОДАТКІВ ТА МОВ ПРОГРАМУВАННЯ

2.1 Вибір веб-додатку в якості типу програмного забезпечення

Реалізація бізнес-ідеї може бути реалізована за допомогою програмування різними способами. Розробник повинен чітко розуміти вимоги користувачів та мати навички для розробки програмного забезпечення. Правильний підхід до впровадження програмного забезпечення є важливою складовою проектування, розберем переваги веб-додатків.

– легка масштабовність: веб-додатки мають гнучкість та здатність до масштабування, завдяки серверному середовищу і хмарним технологіям, їх можна легко масштабувати для виконання зростаючих потреб користувачів;

– простота розповсюдження та оновлення: веб-додатки можуть бути розповсюджені через веб-сервери та доступні користувачам через Інтернет. Оновлення веб-додатків також відбуваються централізовано на сервері, що полегшує впровадження нових версій та виправлення помилок;

– кросплатформенність: веб-додатки можуть працювати на різних платформах і операційних системах без необхідності великої переробки коду. Вони можуть запускатися на Windows, macOS, Linux і багатьох мобільних пристроях.

Щодо недоліків веб-додатків, можна додати наступне:

– залежність від доступу до Інтернету: веб-додатки потребують постійного підключення до Інтернету для доступу до сервера та обміну даними;

– питання безпеки: оскільки дані зберігаються на сервері, веб-додатки можуть мати певний ризик щодо безпеки, особливо якщо не вжиті відповідні заходи щодо захисту даних;

– обмеження браузерів: старі версії браузерів можуть не підтримувати останні технології та стандарти, що використовуються на сайті.

Вибір реалізації програмного застосунку у вигляді сайту з використанням ділення на клієнтську та серверну частини і сервісу Firebase є розумним рішенням, ось деякі переваги та недоліки такого підходу:

– кросплатформенність: веб-додатки можуть бути доступні з будь-якого пристрою з веб-браузером, незалежно від операційної системи, що дозволяє досягти широкої аудиторії користувачів;

– простота розгортання: веб-додатки не вимагають встановлення на кожен клієнтський пристрій, оскільки вони доступні через веб-браузер. Це спрощує процес розгортання та оновлення програмного забезпечення;

– централізоване керування: серверна частина веб-додатка дозволяє зберігати та керувати даними, виконувати бізнес-логіку та керувати безпекою додатка;

– широкий вибір технологій: веб-розробка має велику кількість мов програмування, фреймворків та інструментів, що надають розробникам багато варіантів для вибору і оптимізації розробки.

2.2 Технологої розроблення back end частини ПЗ

При розробці програмного проекту важливо вибрати мови програмування та технології, які найкраще підходять для реалізації конкретного продукту. Для веб-розробки, зокрема для back-end частини веб-додатків, на початку 2022 року найпоширенішими мовами програмування були:

– JavaScript основна мова для веб-розробки [12]. Вона підтримується всіма сучасними браузерами і має широкий набір фреймворків та бібліотек, зокрема React, Angular і Vue.js, які допомагають в розробці складних веб-додатків.

– PHP одна з популярних мов для розробки веб-додатків. Вона має потужні фреймворки, такі як Laravel і Symfony, які полегшують процес розробки і підтримують широкий спектр функцій.

– Java - ця мова програмування використовується не тільки для веб-розробки, але й для розробки платформонезалежних додатків. У веб-розробці Java використовується у поєднанні з фреймворками, такими як Spring і JavaServer Faces (JSF).

– Python ця мова програмування набуває все більшої популярності в веб-розробці. Python має такі фреймворки, як Django і Flask, які дозволяють швидко розробляти веб-додатки з мінімальними зусиллями.

На кінець 2023 року, щодо веб-розробки, фреймворки Flask та Django були найбільш популярними для розробки на мові Python, а для JavaScript основними фреймворками були Express.JS, Socket.io та Meteor.JS, які базувались на платформі Node.js. Одним з факторів порівняння між мовами програмування є швидкодія програми, тобто те, наскільки швидко застосунок відповідає на дії користувачів [3].

Також, останнім часом популярності набирають хмарні сервіси. Хмарні сервіси є інноваційною технологією, яка пропонує хороший спосіб збереження, обробки та доступу до даних. Вони базуються на використанні віддалених серверів, які знаходяться в розподіленому хмарному середовищі. Це дозволяє користувачам зберігати свої дані та запускати програми на цих серверах, отримуючи доступ до них через Інтернет. Один з найпопулярніших прикладів хмарних сервісів – Firebase, який надає широкий спектр функціональності для розробки веб- та мобільних додатків. Хмарні сервіси дозволяють користувачам масштабувати свої додатки, веб-сайти, мобільні додатки або настільні програми. Вони забезпечують зручний спосіб зберігання та резервного копіювання даних, доступ до них з різних пристроїв та місць, а також захищеність і надійність.

Використання хмарних сервісів дозволяє зменшити витрати на обладнання та інфраструктуру, оскільки не потрібно мати власний серверний парк. Крім того, вони забезпечують гнучкість та швидкість розгортання нових функцій та оновлень, що є важливим для сучасних бізнесів. Одним із основних переваг

хмарних сервісів є можливість простого масштабування, коли попит на ресурси зростає, хмарні сервіси автоматично можуть адаптуватися і збільшувати їх обсяг, що дозволяє забезпечити незмінну продуктивність та доступність для користувачів. Такі технології, як Firebase, відкриває безліч можливостей для розробників, підприємств та індивідуальних користувачів. Вони дозволяють зосередитися на розробці додатків, забезпечуючи швидку та надійну інфраструктуру для їх розгортання та ефективну роботу з даними.

Firebase, як зазначено, є платформою розробки застосунків, яка надає широкий спектр інструментів та сервісів для розробки back-end та front-end компонентів веб- та мобільних додатків. Firebase BaaS (Backend as a Service) надає розробникам можливості хостингу, зберігання даних, аутентифікації, реального часу, обліку подій, зберігання файлів та інші сервіси веб-додатка. Використання Firebase дозволяє розробникам зосередитись на функціональності додатку, не витрачаючи багато часу на конфігурування та керування серверами. Основні функції Firebase BaaS включають:

– Firebase Authentication - аутентифікація користувачів;

– Cloud Firestore – NoSQL база даних;

– Cloud Functions – створення та розгортання функцій відповідно до подій у додатку;

– Cloud Storage – зберігання файлів в хмарі. Firebase також надає інші корисні сервіси, такі як аналітика, тестування, розсилки сповіщень та багато іншого.

У підсумку, хмарні сервіси є переважною технологією у сучасному світі, що надає зручний доступ до ресурсів, збереження даних та розробку програмного забезпечення. Вони сприяють інноваціям, економії витрат, забезпечують безпеку та надійність, та дають змогу швидко реагувати на зміни потреб користувачів. Firebase виступає чудовим прикладом хмарного сервісу, який використовується для розробки сучасних додатків з високою функціональністю та ефективністю.

2.3 Технології розроблення front end частини ПЗ

Розроблення front-end є процесом створення користувацького інтерфейсу веб-додатків, який включає в себе розмітку сторінок, стилізацію, динамічну взаємодію та інші функціональні аспекти. Front-end розробники використовують різні технології та інструменти, щоб створити зручний та привабливий інтерфейс для користувачів. Один з ключових елементів розробки front-end - це HTML, мова розмітки, яка визначає структуру веб-сторінки. Розробники використовують теги для створення різних елементів, таких як кнопки, заголовки, форми, таблиці та інші, і визначають їх розмітку на сторінці. Для стилізації інтерфейсу використовується CSS. CSS дозволяє задавати кольори, шрифти, розміщення елементів, анімацію та багато інших аспектів візуального вигляду сторінок. За допомогою CSS, розробники можуть створювати привабливі та естетичні дизайни, а також забезпечувати адаптивність інтерфейсу під різні пристрої і розміри екранів. JavaScript є обов'язковою частиною розробки front-end. Ця мова програмування дозволяє створювати динамічну взаємодію на сторінках, оброблювати події, змінювати вміст сторінок, маніпулювати даними та багато іншого. Використовуючи JavaScript, розробники можуть створювати інтерактивні елементи, валідувати форми, виконувати асинхронні запити до сервера та реалізовувати різноманітні функціональність на стороні клієнта. Крім того, для розробки front-end використовуються різні фреймворки і бібліотеки, які полегшують та прискорюють розробку, вони надають готові рішення, структури та компоненти, які спрощують та прискорюють процес розробки.

Фреймворк - це набір інструментів, бібліотек та правил, що допомагають розробникам створювати програмне забезпечення швидше і ефективніше. Він надає структуру та шаблони для розробки, дозволяючи розробникам фокусуватись на основних функціях додатку, а не на деталях інфраструктури. Важливими концепціями для фреймворку є DOM та Virtual DOM.

DOM (Document Object Model) - це представлення структури HTML-документа у вигляді дерева об'єктів, доступне для маніпуляції та змін за допомогою JavaScript. DOM визначає інтерфейс, який дозволяє програмам читати і змінювати контент, стиль і поведінку веб-сторінок.

Віртуальний ДОМ (Virtual DOM) - це концепція оптимізації роботи з DOM, яка використовується у бібліотеках та фреймворках, таких як React, Vue і Angular. Він є абстрактним представленням реального DOM і зберігає його структуру і стан у пам'яті. Зміни, внесені в віртуальний DOM, порівнюються з реальним DOM і виконуються тільки необхідні оновлення, що забезпечує ефективну роботу з інтерфейсом користувача. Використання віртуального DOM може покращити продуктивність і швидкість роботи веб-додатків. Основна ідея полягає в тому, що у браузері є реальний DOM, який представляє структуру сторінки, але він може бути досить повільним і затратним на оновлення при зміні даних. Віртуальний DOM дозволяє уникнути безпосереднього взаємодії з реальним DOM, працюючи з його віртуальною копією. Коли відбувається зміна даних в програмі, фреймворк або бібліотека перераховує та оновлює віртуальний DOM, порівнюючи його зі старим станом. Після цього він виявляє лише ті частини, які змінилися, і оновлює їх у реальному DOM, як показано на рис. 2.1. Цей підхід дає змогу знизити кількість операцій з реальним DOM, що позитивно впливає на продуктивність додатка. Загалом, віртуальний DOM є потужним інструментом для оптимізації та прискорення роботи з інтерфейсом користувача. Він дозволяє забезпечити швидке та ефективне оновлення сторінки, зменшити кількість маніпуляцій з реальним DOM.

Бібліотеки, з іншого боку, призначені для розширення можливостей мови програмування та надання додаткових функціональних можливостей. Наприклад, бібліотека jQuery спрощує маніпулювання DOM-елементами та взаємодію з AJAX-запитами. Axios бібліотека допомагає виконувати HTTP-запити до сервера. Day.js забезпечує розширені можливості роботи з датами і часом. Ці бібліотеки

можуть бути використані окремо або в поєднанні з фреймворками для забезпечення додаткової функціональності.

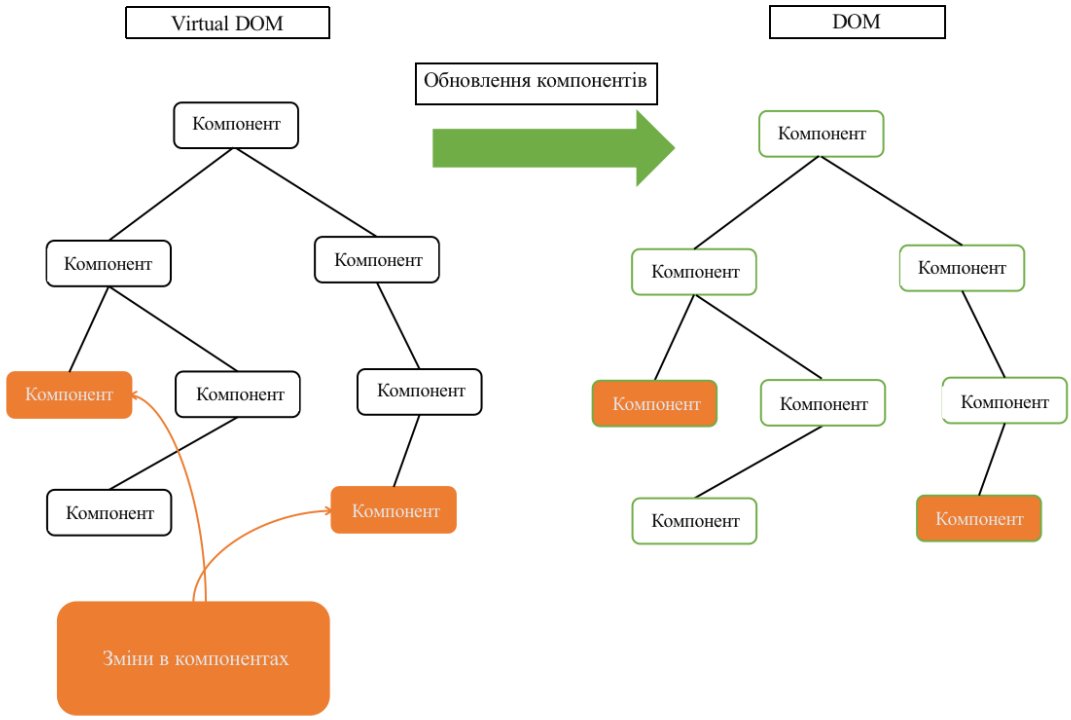

Рисунок 2.1 – Концепція Virtual DOM

Однією з популярних технологій для розробки front-end додатків є React. React - це бібліотека, яка дозволяє розробникам будувати інтерфейси, використовуючи компонентний підхід. Вона базується на концепції віртуального DOM, що забезпечує ефективне оновлення та рендеринг інтерфейсу. React також підтримує розширення через багато сторонніх бібліотек та фреймворків, таких як Redux для керування бізнес-логікою.

Іншим популярним фреймворком є Angular. Angular - це повноцінний фреймворк, який надає широкий набір інструментів для розробки веб-додатків. Він має вбудовану систему модулів, директив, сервісів, керування станом і підтримує двостороннє зв'язування даних, які сприяють організації та підтримці складних додатків.

Vue.js є ще одним популярним фреймворком front-end, який пропонує легкість використання та навчання. Vue.js базується на компонентній архітектурі,

яка дозволяє розробникам створювати реактивні інтерфейси з легкістю. Він також має вбудовану систему керування станом та підтримує директиви та додаткові функції для зручної маніпуляції DOM-елементами.

Препроцесори, як Sass, Less та Stylus, дозволяють розширити функціональність CSS та забезпечити більшу гнучкість при написанні стилів. Вони додають нові можливості, такі як змінні, міксіни, вкладені селектори та багато іншого, що допомагає покращити організацію та підтримку CSS-коду. Ці препроцесори компілюються в звичайний CSS, який браузер може розуміти. Основна різниця між ними полягає у синтаксисі та функціональності, але загальні принципи їх роботи схожі.

TypeScript є розширенням JavaScript, яке надає статичний типізацію та додаткові функції, що полегшують розробку складних проектів [11]. Вона надає можливість визначати типи змінних, параметрів функцій та інших елементів програми. TypeScript компілюється до чистого JavaScript, тому його код може запускатись у будь-якому середовищі, яке підтримує JavaScript.

UI бібліотеки, такі як Bootstrap, Element Plus, Tailwind CSS та інші, надають набір готових компонентів та стилів, що допомагають швидко побудувати красивий та функціональний інтерфейс. Ці бібліотеки мають заздалегідь розроблені компоненти, такі як кнопки, форми, таблиці, навігація та інші, які можна використовувати безпосередньо або налаштовувати під свої потреби. Вони також надають стилізацію та адаптивність, що дозволяє забезпечити єдиний та сучасний вигляд веб-додатку. На рис. 2.2 зображено, сучасні технології для розробки веб-додатків.

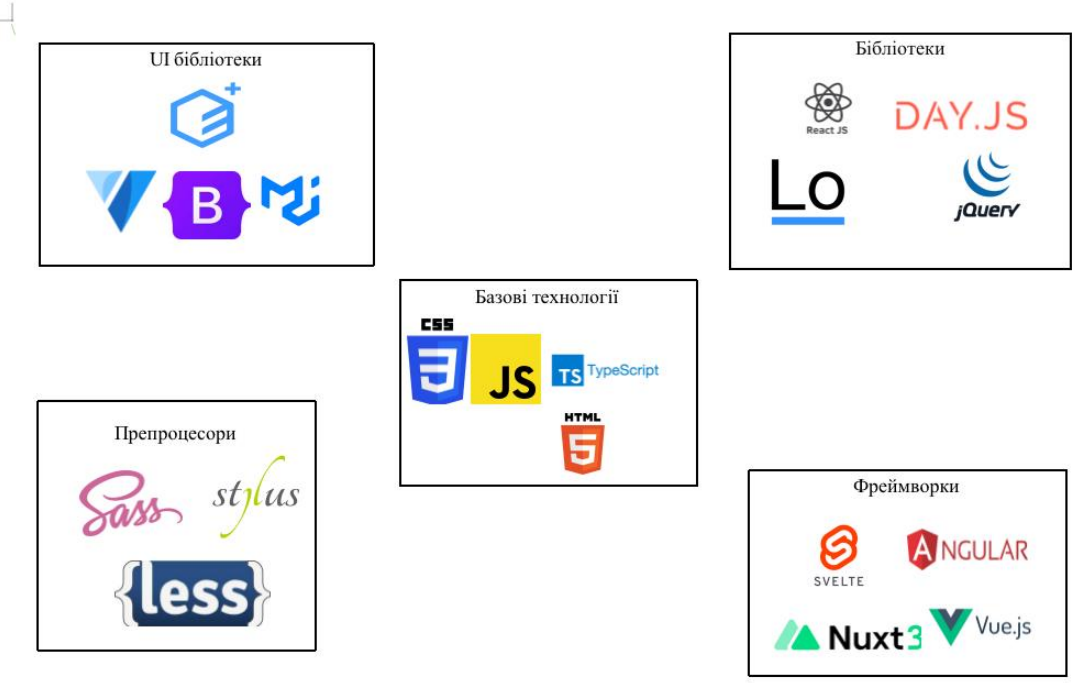

Рисунок 2.2 – Набір сучасних front end технологій

Комбінування цих технологій дозволяє розробникам створювати сучасні, інтерактивні та ефективні веб-додатки.

2.4 Обґрунтування вибору технологій для розробки ПЗ

Для дипломної роботи було проведено дослідження та оцінка різних технологій розроблення з метою вибору найбільш кращих для проекту. Вибір було зроблено на користь сучасного і потужного фреймворку Vue та Firebase платформи.

Комбінація Vue 3 та Firebase забезпечує потужний інструментарій для розробки веб-додатків. Vue 3 дозволяє створювати елегантний та інтерактивний інтерфейс, використовуючи компонентний підхід, а Firebase надає готові сервіси, що спрощують роботу з автентифікацією, базами даних та іншими складовими розробки. Таке поєднання дозволить ефективно реалізувати функціональні та нефункціональні вимоги до програмного забезпечення, забезпечуючи швидку розробку та надійну роботу додатку.

У Vue 3 компоненти використовуються для побудови веб-додатків. Компоненти – це незалежні, повторно використовувані блоки коду, які об'єднують HTML-розмітку, CSS-стилі та JavaScript-логіку, пов'язану з відображенням.

Основною ідеєю компонентного підходу є розділення веб-сторінки на малий набір незалежних компонентів, які можна легко керувати окремо. Компоненти можна розглядати як будівельні блоки, з яких складається сайт, як зображено на рис. 2.3. Вони забезпечують зручність, модульність та повторне використання коду[8].

У фреймворку компоненти описуються за допомогою синтаксису Single File Components (SFC), де HTML-розмітка, CSS-стилі та JavaScript-логіка розташовані в одному файлі з розширенням .vue. Завдяки компонентній архітектурі Vue 3, розробники можуть створювати чистий, організований та легко розширюваний код. Компоненти дозволяють розділити складний інтерфейс на простіші частини, сприяючи покращенню тестування та розробки веб-додатку.

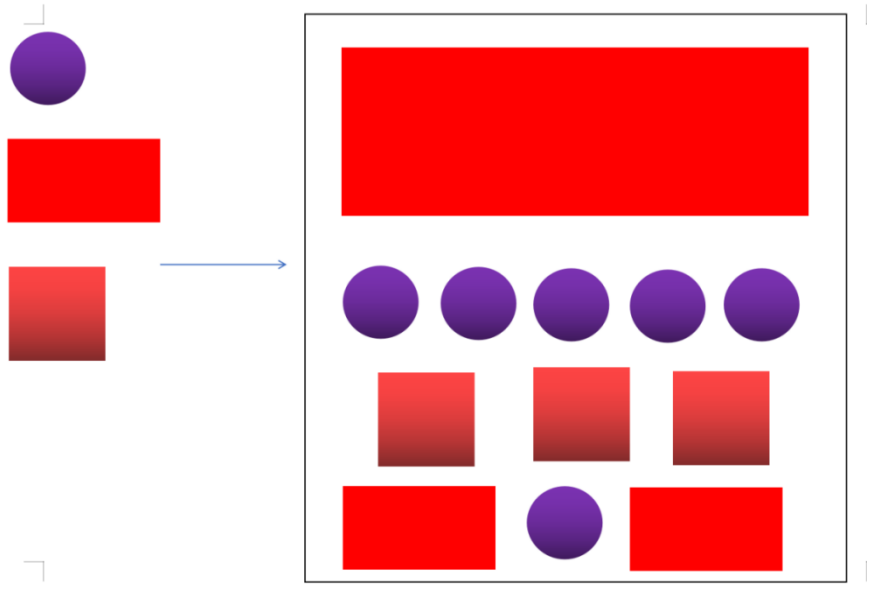

Рисунок 2.3 – Принцип компонентів

Vue – це сучасний і популярний фреймворк для розробки веб-інтерфейсів. Його вибір зумовлений кількома перевагами. По-перше, Vue пропонує просту та

інтуїтивно зрозумілу синтаксичну модель, що сприяє швидкому навчанню та ефективній розробці. Він також має широкий набір інструментів та розширень, що дає змогу збільшити функціонал та забезпечує гнучкість у роботі. Крім того, Vue 3 має високу продуктивність, що забезпечує плавну роботу веб-додатків. Основними особливостями Vue є простота використання, прогресивний підхід до розробки, швидкість. Фреймворк дозволяє починати з маленьких та простих проектів, а поступово додавати складніші функції та розширювати функціональність. Фреймворк використовує ефективну систему віртуального DOM, яка забезпечує оптимальне оновлення компонентів і покращує продуктивність додатку. Vue 3 також має широкий набір інструментів та розширень, які допомагають розробникам покращити продуктивність та зручність роботи. Наприклад, Vue Router дозволяє легко налаштовувати маршрутизацію в додатку, а Pinia забезпечує управління бізнес-логікою, а саме для зберігання та спільного використання даних. Технічно Vue використовує шаблон MVVM [8]. Він з’єднує представлення та модель двостороннім зв’язуванням даних. MVVM є абревіатурою, що означає Model-View-ViewModel, і є одним з популярних шаблонів архітектури програмного забезпечення. У шаблоні MVVM модель представляє собою джерело даних, таке як база даних або сервер, яке зберігає інформацію про додаток. Вона відповідає за доступ до даних і виконання бізнес-логіки. Модель може бути представлена класами або об'єктами, які включають методи для роботи з даними. Представлення (View) є інтерфейсом, через який користувач взаємодіє з програмою. Це може бути графічний інтерфейс користувача (GUI), веб-сторінка або інший спосіб візуалізації інформації. Представлення відображає дані, отримані з моделі, і передає введення користувача до ViewModel. ViewModel виступає посередником між моделлю і представленням. Він отримує дані з моделі і підготовлює їх для відображення в представленні. ViewModel також обробляє введення користувача та виконує дії, пов'язані зі зміною стану програми або взаємодією з моделлю. Він надає методи та властивості, до яких можна зв'язати представлення, щоб забезпечити двосторонню

зв'язок даних. Це означає, що зміни, внесені в даних в представленні, автоматично відображаються в ViewModel, і навпаки. Це спрощує синхронізацію даних між компонентами та робить код більш прогнозовуваним та легким для розуміння. Шаблон допомагає відокремити бізнес-логіку від представлення, що сприяє покращенню модульності та обслуговуваності коду.

Зв'язування даних і команди – це два важливих концепти, які використовуються в шаблоні MVVM. Зв'язування даних (data binding) визначає автоматичну синхронізацію даних між ViewModel і представленням. Це означає, що зміни в даних однієї сторони наприклад, у ViewModel автоматично відображаються в іншій стороні наприклад, у представленні і навпаки, також цей концепт дозволяє зручно прив'язувати значення властивостей ViewModel до елементів відображення, таких як текстові поля, чекбокси або списки, щоб автоматично оновлювати їх при зміні даних. Команди (commands) використовуються для реагування на дії користувача, такі як натискання кнопок або введення клавіш [1,2]. Вони дозволяють ViewModel виконувати певну логіку або викликати методи моделі при спрацюванні події в представленні. Команди дозволяють розділити логіку обробки подій від представлення і забезпечити керовану взаємодію між ViewModel і інтерфейсом користувача. в'язування даних і команди разом допомагають створити потужну взаємодію між ViewModel і представленням, схему можна глянути на рис. 2.4. Наприклад, у додатку з формою реєстрації користувача, ViewModel може мати властивості для збереження імені, електронної пошти та пароля. Ці властивості можуть бути зв'язані з відповідними полями вводу в представленні, так що будь-які зміни в полях автоматично оновлюватимуть відповідні властивості ViewModel. Таким чином, зв'язування даних і команди є потужними інструментами в шаблоні MVVM, які допомагають забезпечити зручну і ефективну взаємодію між ViewModel і представленням у програмах.

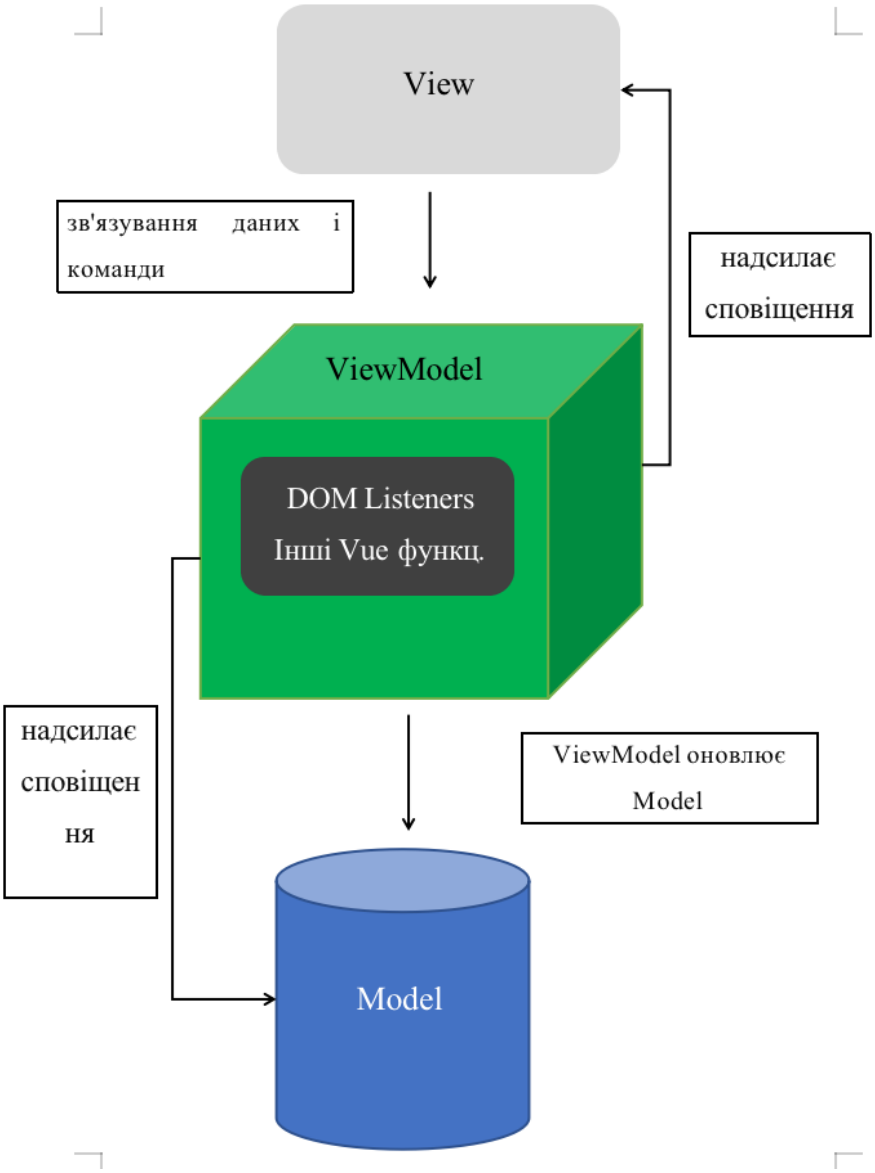

Рисунок 2.4 – Принцип шаблону MVVM в Vue

Загалом, Vue 3 є потужним та гнучким фреймворком, який дозволяє розробникам швидко створювати інтерактивні та ефективні веб-інтерфейси. Його простота використання, прогресивний підхід, швидкість та розширюваність роблять його популярним вибором для розробки сучасних веб-додатків.

Firebase, з іншого боку, є хмарною платформою, яка надає різноманітні сервіси для розробки та підтримки веб-додатків. Вибір Firebase також має свої причини. Платформа забезпечує швидке та просте впровадження аутентифікації,

бази даних в режимі реального часу, зберігання файлів, хостинг, пуш-сповіщення та багато інших функцій. Це дозволяє розробникам швидко створювати та масштабувати веб-додатки без необхідності власноручно налаштовувати. На рис. 2.5 зображено роботу авторизаційної системи у Firebase.

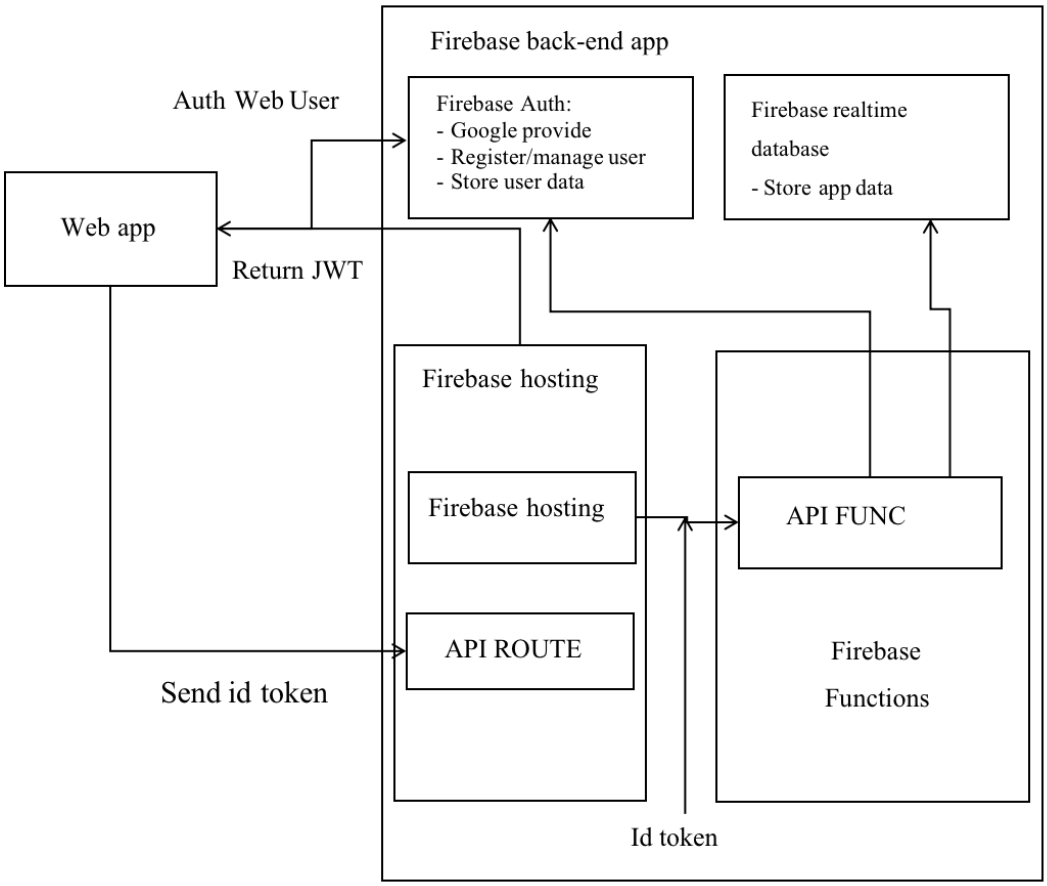

Рисунок 2.5 – Авторизаційна система у Firebase

Користувач вводить свої облікові дані електронну пошту та пароль в додатку або на веб-сторінці, додаток передає ці дані до Firebase Authentication, в свою чергу Firebase Authentication перевіряє, чи існує користувач з такими обліковими даними, якщо користувач існує і введені облікові дані є правильними, API Route Firebase генерує JWT (JSON Web Token) токен для цього користувача [9]. JWT є стандартом відкритого формату для передачі інформації між двома сторонами в безпечному та компактному вигляді. Цей токен повертається до

додатку. Додаток зберігає цей JWT в локальному сховищі (наприклад, у веб-браузері використовуючи localStorage або в мобільному додатку використовуючи SharedPreferences). Додаток використовує цей JWT для авторизації в майбутньому. При кожному запиті до Firebase API додаток надсилає цей JWT як частину заголовка запиту. Firebase Authentication перевіряє цей JWT, щоб підтвердити ідентичність користувача і дозволити або обмежити доступ до запиту. Якщо JWT є дійсним і ідентичність користувача підтверджена, Firebase Authentication відповідає на запит або виконує необхідну операцію. Це загальний процес авторизації за допомогою Firebase [9]. Під капотом Firebase використовує різні криптографічні техніки для захисту облікових даних, генерує та перевіряє JWT, а також забезпечує безпеку підключення між додатком і Firebase-серверами.

При розробці frontend частини веб-додатку необхідно враховувати різні фактори, такі як розмір та роздільна здатність екрана, різні браузери, операційні системи та пристрої, на яких буде відображатись веб-сайт. Це вимагає від розробника уважного планування та тестування, щоб забезпечити, що веб-сторінка виглядає та працює належним чином на різних пристроях та платформах.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ МОНАТОРИНГУ ТА ВІЗУАЛІЗАЦІЇ ВАРТОСТІ КРПИТПОВАЛЮТ

3.1 Структура проекту

Структура проекту є важливим аспектом дипломної роботи, оскільки вона впливає на організацію коду, його читабельність та підтримку проекту в майбутньому. Добре організована структура дозволяє забезпечити ефективну розробку, тестування та супровід програмного забезпечення. Для правильної побудови сткруктури необхідно дотримуватись наступних правил:

1) Розділення на модулі, потрібно розділітити проект на логічні модулі або компоненти. Кожен модуль повинен відповідати конкретній функціональності або розділу додатку. Це допоможе зробити проект більш організованим та легким для розширення.

2) Групування за типами файлів, варто розділити файли за їх типами або функціональністю. Наприклад, окремі папки для компонентів, маршрутів, сервісів, стилів, тестів тощо. Це полегшить знаходження потрібного файлу та зменшить плутанину в структурі проекту.

3) Іменування файлів, використовування імен, які ясно вказують на їх призначення. Наприклад, назви файлів можуть відображати назви компонентів або функціональності, яку вони виконують.

4) Вкладеність папок, використовування вкладеності папок, коли це має сенс допомагає структурувати проект, як показано на рис. 3.1.

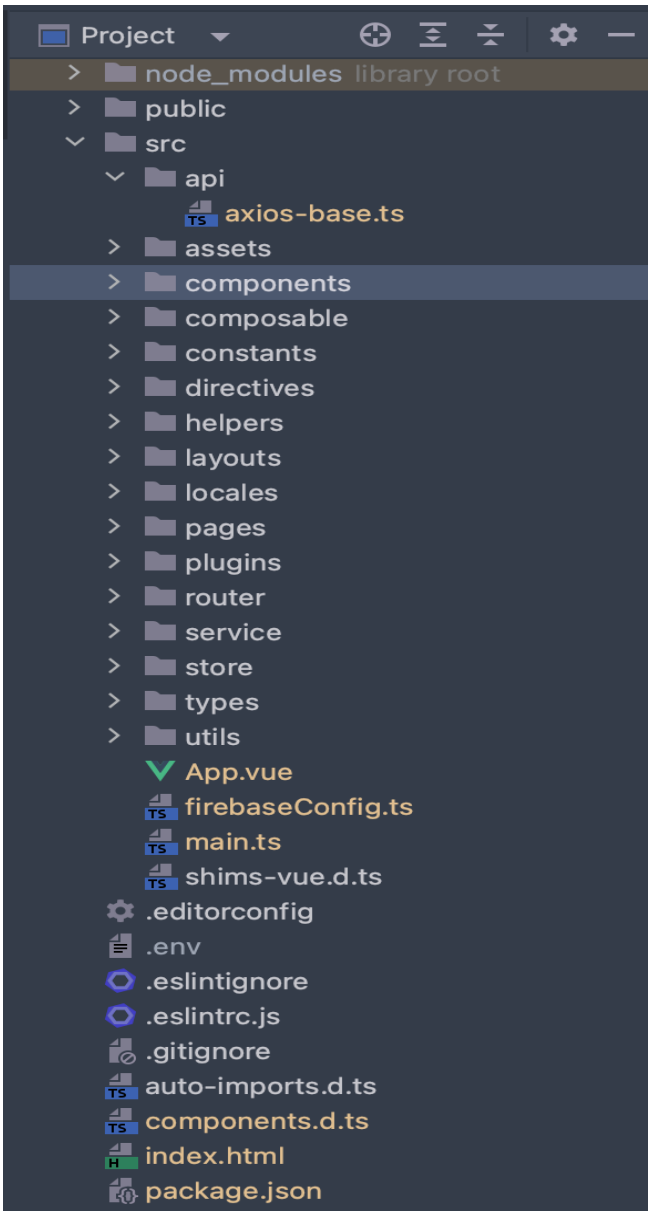

Рисунок 3.1 – Структура дипломного проекту

Проект складається з кількох папок та конфігураційних файлів, розглянути сткруктуру проекту можна на блок-схемі (рис. 3.2).

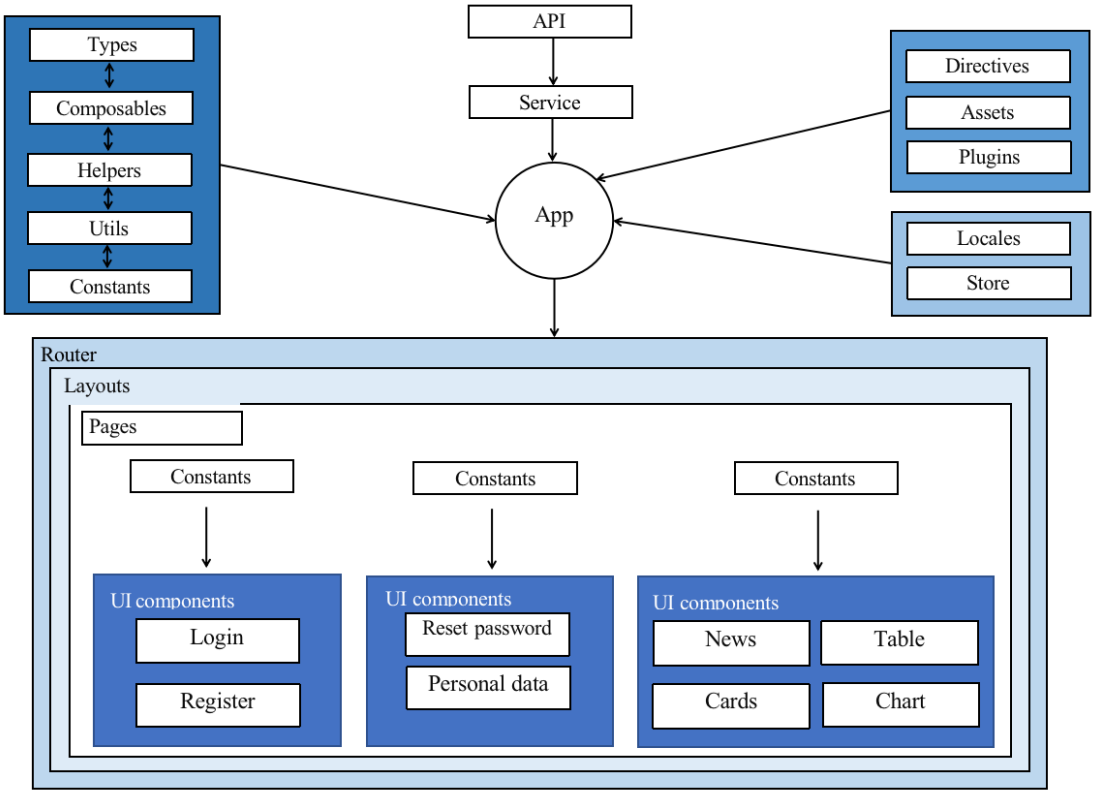

Рисунок 3.2 – Блок-схема веб-додатку

Папка node_modules – це папка, в якій знаходяться всі залежності, необхідні для проекту. Вона містить готові пакети, які використовуються в проекті. Ця папка генерується автоматично під час встановлення залежностей за допомогою пакетного менеджера, такого як npm або yarn.

Папка public – це папка, в якій можна зберігати публічні ресурси сайту або додатку, такі як фавіконка, зображення, статичні файли тощо. У цій папці можна розміщувати файли, які будуть доступні напряму зі сторони клієнта, без обробки сервером. Увесь код зібраний у головній папці src, де є ще додаткові під категорії.

Api – папка з єдиним файлом, використана для організації коду, пов'язаного з взаємодією з API. Вона містить основні екземпляри Axios для взаємодії з API, які відповідать за виконання запитів до сервера, обробку відповідей та управління даними, пов'язаними з API, приклад використання можна побачити у лістингу 3.1.

Лістинг 3.1 – Створення екземплярів за допомогою бібліотеки Axios

```
import axios from "axios";
axios.defaults.headers.common["Content-Type"] =
"application/json";

export const axiosCoinRanketInstance = axios.create({
  baseURL: import.meta.env.VITE_COIN_RANKET_API_URL,
  headers: {
    "x-access-token":
import.meta.env.VITE_COIN_RANKET_API_TOKEN,
    "Access-Control-Allow-Origin": "*",
  },
});
export const axiosCryptoCompareInstance = axios.create({
  baseURL: import.meta.env.VITE_CRYPTO_COMPARE_API_URL,
  headers: {
    authorization: `Apikey
${import.meta.env.VITE_CRYPTO_COMPARE_API_TOKEN}`,
  },
});
export const axiosBinanceInstance = axios.create({
  baseURL: import.meta.env.VITE_BINANCE_API_URL,
});
export const axiosByBitInstance = axios.create({
  baseURL: import.meta.env.VITE_BYBIT_API_URL,
});
export const axiosCoinGeckoInstance = axios.create({
  baseURL: import.meta.env.VITE_COIN_GECKO_API_URL,
  headers: {
    "Access-Control-Allow-Origin": "*",
  },
});
```

Assets – це папка в якій зібрані усі стилі та картинки, що використовуються на сайті. Стилі написані на препроцесорі SCSS з використанням спеціальної бібліотеки для скороченння стилів Tailwind. Tailwind CSS - це популярний CSS-фреймворк, який пропонує набір готових утилітних класів для створення стилів, його основна ідея полягає в тому, щоб спростити процес розробки користувацьких інтерфейсів, надаючи широкий набір готових стилів [7]. Структура папки assets, зображена на рис. 3.3 та приклад використання у лістингу 3.2.

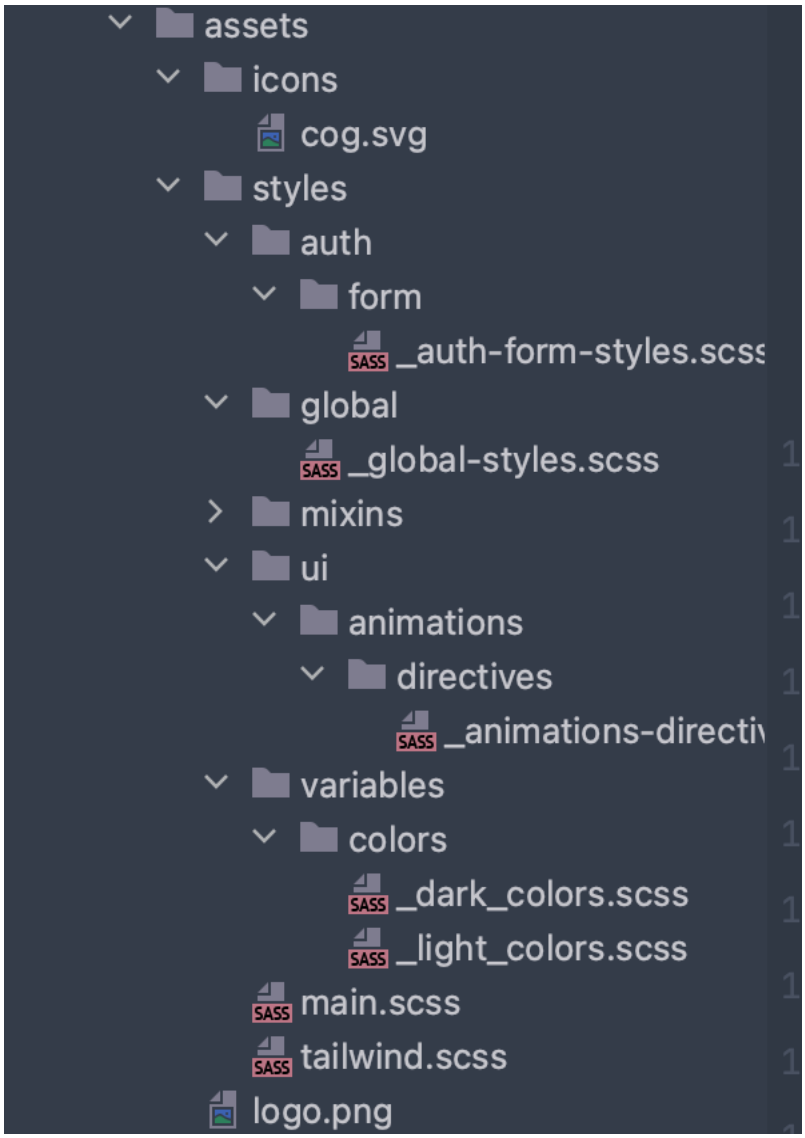

Рисунок 3.3 – Структура папки assets

Лістинг 3.2 – Приклад написання стилів з використання препроцесора SCSS та бібліотеки Tailwind

```
.auth-form {
  @apply bg-white relative flex-center flex-col py-0 px-[50px]
gap-2 h-full text-center;
  &__button {
    @apply rounded-[20px] mt-3 border border-[#ff4b2b] text-
white text-[12px] font-bold bg-[#ff4b2b];
    padding: 12px 45px;
    letter-spacing: 1px;
    text-transform: uppercase;
    transition: transform 80ms ease-in;
  }

  &__button:active {
    transform: scale(0.95);
  }
```

```
  &__button:focus {
    outline: none;
  }
}
```

Папка components – створена для зберігання компонентів. Компонент це частина коду HTML, CSS, VUE, який можна перевикористовувати. Компонент у VUE 3 мають мати строгу стркуктуру, яка складається з 3 частин:

- блоку коду для HTML, обгорнутий у спеціальний тег <template>;
- блок коду для Vue;
- блок коду для стилів.

Лістинг 3.3 – Приклад компонента

```
<template>
<p class="text-xs absolute load-fade text-red-500 text-start pl-
2 bottom-[-15px]"
  >
    <span>{{ errors }}</span>
  </p>
</template>

<script setup lang="ts">
defineProps<{ errors: string }>();
</script>
<style lang="scss">
.load-fade {
  animation: fade-screen 1s 1 ease-in;
  animation-fill-mode: forwards;
}
@keyframes fade-screen {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
</style>
```

У Vue 3 компонент є основною будівельною одиницею для розробки веб-додатків, приклад коду зображено у лістингу 3.3. Він дозволяє структурувати та організовувати код, розділяти логіку і представлення, а також повторно використовувати компоненти в різних частинах додатку. У Vue 3 компоненти

можна створювати за допомогою синтаксису Vue 3 Composition API або за допомогою синтаксису Options API. В данній дипмлоній роботі був обраний Composition API як основний синтаксис для фреймворку Vue 3. Composition API є новим підходом до створення логіки компонентів у Vue 3. Вона надає можливість групувати пов'язані логіку та функції у комбіновані функції (composables), які можуть бути повторно використані в різних компонентах.

Composables – це папка в структурі проекту Vue, яка використовується для зберігання Vue-композицій. Композиції є способом організації та повторного використання логіки в Vue-компонентах. Вони дозволяють винести спільну функціональність з компонентів до окремих модулів, які можуть бути використані в багатьох компонентах. Кожен файл може містити одну або декілька композицій, які виконують певні функції або надають певну функціональність. Vue-композиції - це функції, які можуть використовуватись в компонентах для надання певної функціональності. Вони можуть містити стан (state), обчислювальні властивості (computed properties), методи (methods) та інші логічні блоки коду, які можуть бути використані в компонентах. Використання папки "composables" допомагає зберігати логіку проекту в окремих модулях, що полегшує її управління, розширення та повторне використання

Constants – папка в контексті проекту відноситься до структури, в якій зберігаються константи, тобто значення, які залишаються постійними протягом проекту. Ці константи можуть бути використані в різних частинах програми і забезпечують єдність та уніфікацію значень.

Лістинг 3.4 – Приклад використання константи

```
export const LAYOUT_NAMES = Object.freeze(<Record<string,
LayoutNames>>{
  DEFAULT: "default",
  AUTH: "auth",
});
```

У вище наведеному лістингу 3.4 константи пишутся у camel case стилі, щоб в коді виділити, що це саме є константа, а також задля забезпечення безпеки, і

неможливості випадкової зміни змінної, використовується спеціальна функція freeze глобального обєкта Object. Метод Object.freeze() в JavaScript використовується для "заморожування" об'єкта. Це означає, що він перешкоджає додаванню нових властивостей до об'єкта, видаленню існуючих властивостей з об'єкта та зміні значень існуючих властивостей.

Directives – у Vue директиви - це спеціальні атрибути, які можуть додаватись до HTML елементів в шаблоні компонента. Вони використовуються для розширення функціональності елементів, надаючи їм певну поведінку або здатність реагувати на зміни в даних або подіях. Vue має вбудовані директиви, такі як v-if, v-for, v-bind, v-on та інші.

Helpers – ця папка в проекті Vue може використовуватись для зберігання допоміжних функцій, утиліт або модулів, які використовуються в різних частинах додатку. Основна ідея папки helpers полягає в тому, щоб розмістити функції або модулі, які надають загальні функціональні можливості, які можуть бути використані в багатьох компонентах або середовищах додатку.Ці файли можуть містити функції, які імпортуються та використовуються в різних компонентах або інших частинах додатку.

Layouts – використовується для зберігання шаблонів (layouts) сторінок, які визначають загальну структуру та розмітку для декількох сторінок додатку. Шаблони в папці layouts містять загальні компоненти, такі як заголовок, навігаційне меню, бічна панель, тощо, які повторюються на декількох сторінках додатку. Вони надають основну структуру та розмітку сторінок, а також можуть містити зв'язані дані та функціональні елементи, які необхідні для багатьох сторінок. Основна ідея полягає в тому, щоб зосередити загальну логіку та розмітку в одному місці, щоб уникнути дублювання коду та спростити підтримку додатку. За допомогою шаблонів, ви можете визначити структуру сторінки один раз і використовувати її для декількох сторінок.

Locales – використовується для зберігання локалізаційних файлів, які містять текстові ресурси для різних мов або регіональних налаштувань. У Vue локалізаційні файли можуть бути завантажені та використані за допомогою

спеціальних бібліотек або плагінів, таких як vue-i18n, які надають зручний спосіб керування локалізацією додатка.

Лістинг 3.5 – Приклад використання локалізованого файлу

```
{
  "global": {
    "english": "English",
    "ukrainian": "Ukrainian",
    "dashboard": {
      "news": "Latest News",
      "cards": {
        "top": "Top Gainers Crypto",
        "lose": "Top Losers Crypto",
        "trend": "Top Trend Crypto",
        "watchlist": "My Watchlist",
        "watchlist_empty": " You don't have any crypto in your
watchlist",
        "add": "Add to Watchlist",
        "remove": "Remove from Watchlist"
      }
    }
  }
}
```

Дані зберігаються у форматі JSON (JavaScript Object Notation) - це легкий формат обміну даними, який широко використовується для передачі структурованої інформації в інтернеті, приклад можна побачити у лістингу 3.5. Він базується на синтаксисі мови JavaScript , на колекції пар "ключ-значення", де ключі є рядками, а значення можуть бути різних типів даних, таких як рядки, числа, булеві значення, масиви, об'єкти або навіть null. Всі ці типи даних можуть бути вкладеними один в одного, що дозволяє створювати складні структури даних.

Pages – використовується для організації сторінок веб-додатку. Кожен файл у папці "pages" представляє окрему сторінку додатку і містить компонент Vue, який відповідає за рендеринг та функціональність цієї сторінки.

Plugins - можна розмістити додаткові плагіни, які розширюють функціональність додатку. Плагіни є зручним способом інтеграції сторонніх бібліотек або налаштування глобальних налаштувань для додатку, зокрема вищезгаданий плагін для локалізації мов vue-i18n

Router – в цій папці розміщується файл, що відповідає за налаштування маршрутизації веб-додатку. Маршрутизація визначає, які компоненти відображаються на різних шляхах (URL) в додатку. Основним файлом, що міститься в папці "router", є index.js або router.js. У цьому файлі визначається маршрути додатку, які відповідають різним URL, глянути приклад використання можна у лістингу 3.6.

Лістинг 3.6 – Використання router

```
const routes: Array<RouteRecordRaw> = [
  {
    path: "/user-auth",
    name: "AuthPage",
    component: AuthPage,
    meta: {
      layoutComponent: LAYOUT_NAMES.AUTH,
    },
  },
  {
    path: "/",
    name: "DashboardPage",
    component: DashboardPage,
    meta: {
      requiresAuth: true,
      layoutComponent: LAYOUT_NAMES.DEFAULT,
    },
  },
  {
    path: "/account-settings",
    name: "AccountPage",
    component: AccountPage,
    meta: {
      requiresAuth: true,
      layoutComponent: LAYOUT_NAMES.DEFAULT,
    },
  },
  {
    path: "/crypto-page/:pair",
    name: "CryptoPageSlug",
    component: CryptoPageSlug,
    meta: {
      requiresAuth: false,
    },
  },
  {
    path: "/preload-page",
    name: "PreloadPage",
    component: PreloadPage,
```

```
    meta: {
      layoutComponent: LAYOUT_NAMES.AUTH,
    },
  },
];
```

Service – в Vue.js розміщуються файли, пов'язані з взаємодією з веб-сервером або зовнішніми API. Ці файли виконують функцію сервісного шару, який відповідає за виконання запитів до сервера, обробку відповідей і забезпечення даних для компонентів вашого додатку. Файли сервісів можуть містити різні методи, які виконують різні види запитів, такі як GET, POST, PUT, DELETE - для цього існує спеціальний термін - CRUD. CRUD - це абревіатура, що означає "створення" (Create), "читання" (Read), "оновлення" (Update) та "видалення" (Delete), приклад зображено в лістингу 3.7. Це основні операції, які виконуються з даними в багатьох системах керування базами даних (СКБД) та веб-додатках. Кожна з цих операцій виконується з метою керування та маніпуляції даними.

Лістинг 3.7 – Приклад CRUD

```
import { AxiosInstance, AxiosRequestConfig } from "axios";
export default abstract class Api {
  private getUrl(url: string) {
    return `${url}`;
  }

  private axiosInstance(instance: AxiosInstance) {
    return instance;
  }

  protected post<T>(
    instance: AxiosInstance,
    url: string,
    data?: T,
    config?: AxiosRequestConfig
  ) {
    return
this.axiosInstance(instance).post<T>(this.getUrl(url),    data,
config);
  }

  protected    put<T>(url:    string,    data?:    T,    config?:
AxiosRequestConfig<T>)
```

```
{
    return   this.axiosInstance.put<T>(this.getUrl(url),   data,
config);
  }

  protected get<T>(
    instance: AxiosInstance,
    url: string,
    config?: AxiosRequestConfig
  ) {
    return
this.axiosInstance(instance).get<T>(this.getUrl(url), config);
  }

  protected       delete<T>(url:       string,       config?:
AxiosRequestConfig<T>) {
    return       this.axiosInstance.delete<T>(this.getUrl(url),
config);
  }
}
```

Store – в контексті Vue, "store" це централізоване сховище даних, яке використовується для керування станом додатку. Store дозволяє зберігати, оновлювати і отримувати дані, які використовуються в різних компонентах додатку. В данній дипломній роботі використовується стейт менджер Pinia - це становий менеджер для Vue, який пропонує альтернативний підхід до керування станом додатку. Він був розроблений з урахуванням простоти, типізації та швидкодії. Pinia працює як плагін для Vue.js і забезпечує зручний і зрозумілий інтерфейс для створення та використання сторінок.

17. Types – тут знаходитяться файли, які визначають різні типи даних або інтерфейси, які використовуються в програмі. Використання типів дозволяє забезпечити типову безпеку та покращити роботу з кодом, допомагаючи виявляти помилки на етапі компіляції. Це можливо завдяки TypeScript - це мова програмування, яка є надмножиною JavaScript [11]. Вона надає статичну типізацію, розширені можливості орієнтованого на об'єкти програмування і підтримує розвиток складних додатків, приклад на можна побачити у лістингу 3.8. TypeScript компілюється до чистого JavaScript і може бути використаний для розробки на стороні клієнта (фронтенд) та на стороні сервера (бекенд).

Лістинг 3.8 – Приклад використання TypeScript

```
import { Component } from "vue";
export type StringOrNumber = string | number;
export type StringOrComponent = string | Component;
export type LocaleType = "en" | "ua";
export type LayoutNames = "default" | "auth";
export type SizeType = "large" | "default" | "small";
```

Utils – у папці "utils" (скорочення від "utilities" або "утиліти") зазвичай знаходяться допоміжні функції, класи або модулі, які мають загальне використання в проекті. Ці утиліти зазвичай містять невеликі фрагменти коду, які вирішують конкретні завдання, і можуть бути використані в різних частинах програми. Терміни "utils" і "helpers" часто використовуються із схожим значенням, і їх використання може варіюватися в залежності від конкретного проекту або команди розробників. Однак, можна розрізнити деяку різницю між ними. Утиліти (utils) зазвичай використовуються для зберігання функцій, класів або модулів, які мають загальне використання і надають допоміжний функціонал. Вони можуть містити різноманітні функції, які вирішують конкретні завдання і можуть бути повторно використані в різних частинах проекту. Файли utils часто орієнтовані на різноманітні функції та допоміжні інструменти, які не належать до жодної конкретної області функціоналу проекту. З іншого боку, "helpers" часто використовуються для зберігання функцій або методів, які надають допомогу певній області функціоналу проекту. Це можуть бути спеціалізовані функції, які забезпечують підтримку, обробку даних або виконання конкретних завдань у певному контексті.

App – є основним компонентом в Vue-проекті і представляє головний шаблон (layout) для додатку, зображено у лістингу 3.9. Він містить кореневий компонент додатку і об'єднує всі інші компоненти, які будуть використовуватись у проекті.

Лістинг 3.9 - Головний компонент App

```
<template>
  <app-layout>
    <router-view />
```

```
  </app-layout>
</template>

<script setup lang="ts">
import AppLayout from "@/layouts/AppLayout.vue";
</script>
```

Main.ts – є вхідною точкою Vue-додатку, він виконується при запуску додатку і виконує необхідні конфігурації та ініціалізацію:

– імпорт необхідних залежностей;

– конфігурація додатку;

– підключення плагінів;

Також в проекті є ще кілька конфігураційних файлів, зокрема файли для настройки tailwind, firebase , eslint , vite.

3.2 Дизайн веб сторінок

Для реалізації дизайну та вмісту сторінок веб-додатку були використані HTML5, SCSS та Vue 3, TypeScript та UI бібліотека Element Plus [10,13]. Веб додаток представляє собою SPA - Singel Page Appication SPA (Single-Page Application) – це архітектурний підхід до розробки веб-додатків, де весь контент завантажується на одну HTML-сторінку, і взаємодія з сервером відбувається через AJAX-запити (асинхронні запити до сервера). У традиційному багатосторінковому веб-додатку при переході між сторінками браузер завантажує нову HTML-сторінку з сервера. У SPA підході цей процес відбувається асинхронно, без повного перезавантаження сторінки. Весь код, що відповідає за рендерінг і взаємодію зі сторінкою, вже знаходиться на клієнтському боці і виконується в браузері. Одна з основних переваг SPA полягає в тому, що завантаження сторінок відбувається швидше, оскільки не потрібно кожного разу передавати весь HTML код з сервера.

Веб додаток складається з трьох сторінок: авторизаційної сторінки, головної сторінки з усією інформацією про криптовалюти, та сторінки з настройками веб додатку і потрапити на ці сторінки можна лише авторизованим

користувачам, при спробі користувачем оновити кеш браузера або стерти дані з local storage, система сайту відправить користувача на сторінку з авторизацією.

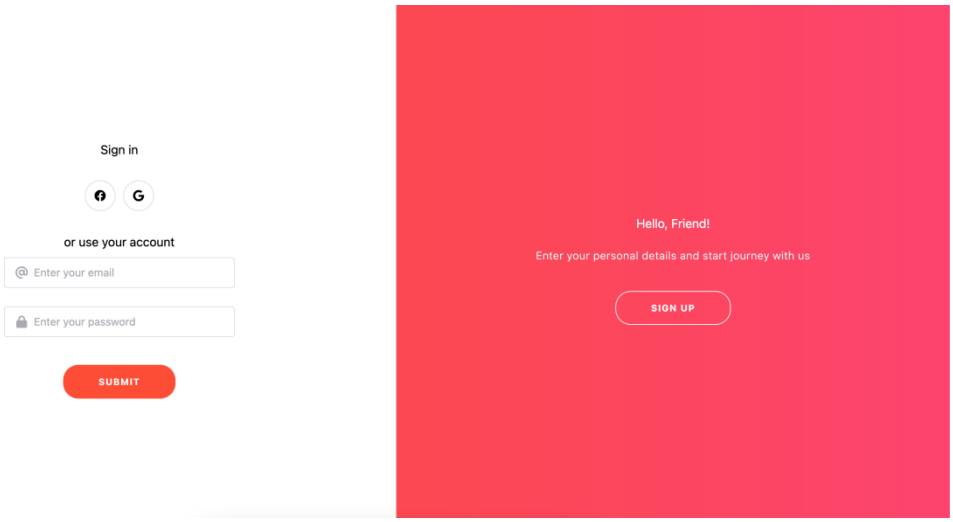

Рисунок. 3.4 – Авторизація у веб-додаток

Сторінка входу до системи зображена на рис. 3.4. На цій сторінці користувачу пропонується ввести пару значень, логін та пароль для входу до системи. Також реалізований функціонал авторизування через соціальну мережу Facebook або через Google.

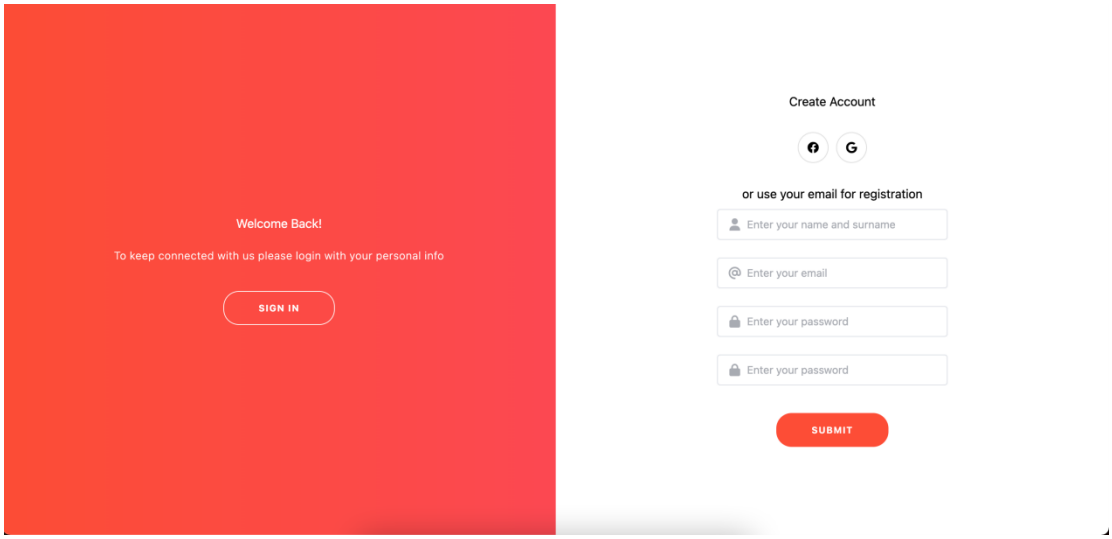

Рисунок 3.5 – Реєстрація у веб-додатку

Користувач при реєстрації повинен вказати своє ім’я, логін, та пароль, як на рис. 3.5, а також пітвердити його, усі поля є обовязковими , якщо користувач введе не правильний логін або пароль, то отримає зустрічне повідомлення нижче не валідного поля, зображено на рис. 3.6.

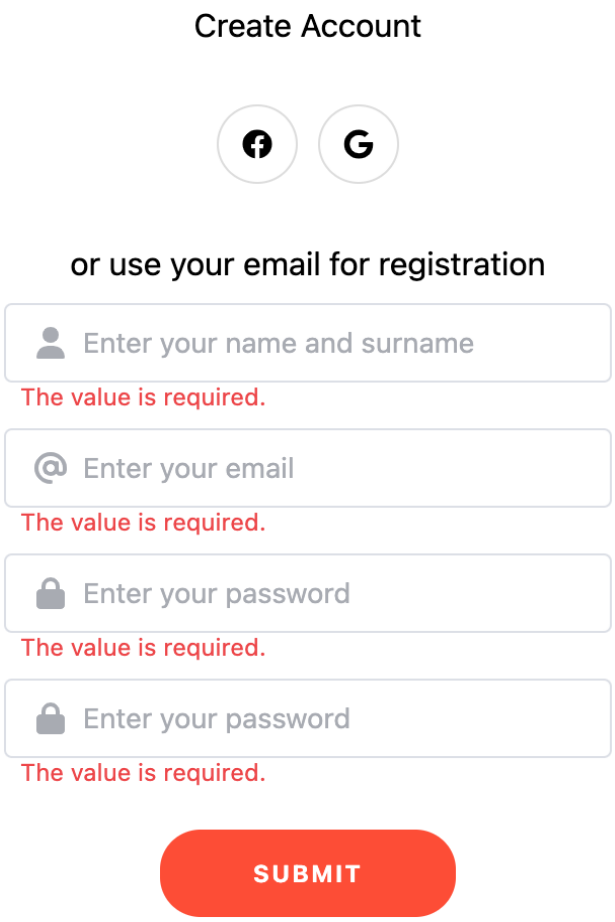

Рисунок 3.6 – Вивід про помилку не валідного поля

Після успішної реєстрації, користувача перенаправить на головну сторінку, де користувач може ознайомитись з поточними цінами на криптовалюту, а також останніми новинами.

Рисунок 3.7 – Головна сторінка

На рис 3.7 у верхній панелі, можна побачити поточний час та день, також можливість змінити мову сайту, його тему, а також кнопки, які переадресують користувача до сторінки з настройками або дозволяють користувачу розлогінитись.

Рисунок 3.8 – Головна сторінка на українській мові

На рис. 3.8 під верхньою панеллю є 4 карточки, кожна з яких є слайдером і користувач може гортати вибрану карточку. Перша карточка дає інформацію про список гірших в данний момент криптовалют, карточка має назву криптовалюти, назву токена її ціну та графік, друга карточка дає інформацію про список найкращих криптовалют, третя карточка про криптовалюту яка зараз у всіх на

слуху, четверта карточка це список криптовалют, які користувач добавив у свій список відслідковуваних. Також добавлено логіка, додавання до списку вибраних криптовалют, лише на клік по іконці зірочки, рис. 3.9.

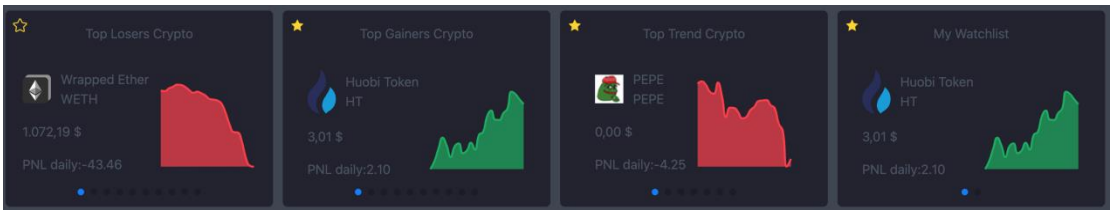

Рисунок 3.9 – Добавлення криптовалют в список відслідковуваних

Криптовалюти напряму залежать від новин на ці самі криптовалюти, тому для користувача є можливість переглядати останні новини про крипто світ, як на рис. 3.10.

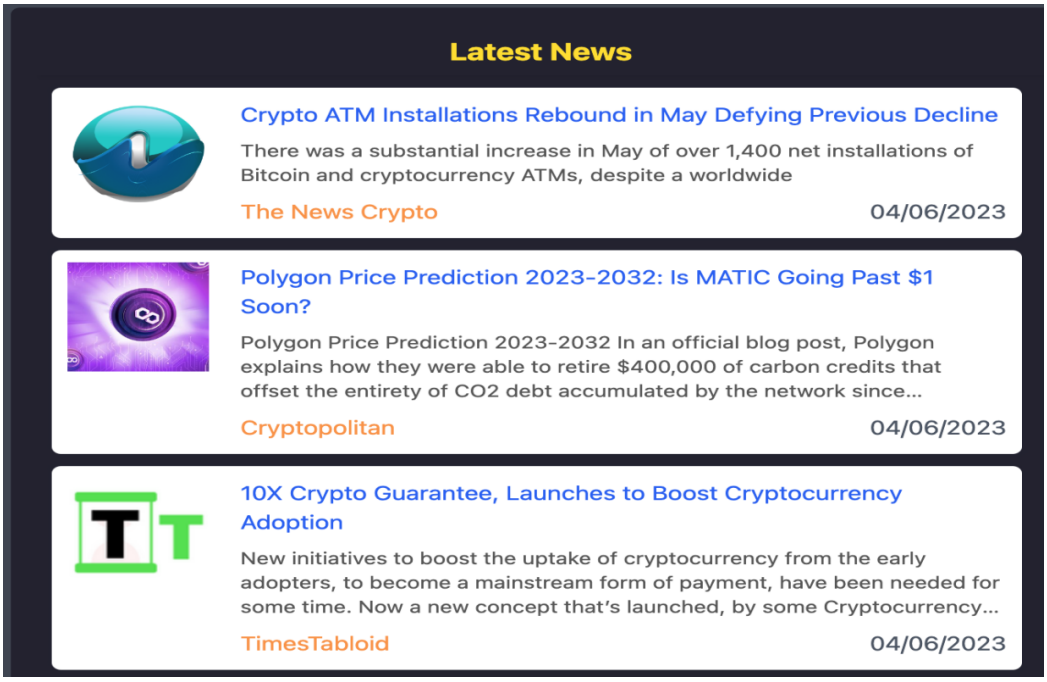

Рисунок 3.10 – Останні новини про криптовалюти

При кліці на назви статей, користувача перенаправить на сайт з новиною.

Графік є багатофункціональним і дозволяє дивитися за крпитовалютою у часовому проміжку на рис. 3.11, можна вибрати часовий проміжок:

1) 1 секунда;

2) 15 хвилин;

3) 1 година;

4) 4 години.

Рисунок 3.11 – Лінійний графік

Також дозволяється обрати спосіб або лінійним графік, див. рис. 3.11 або у вигляді японських свічок, рис. 3.12, варто зазначити, що графік оновлюється щосекунди.

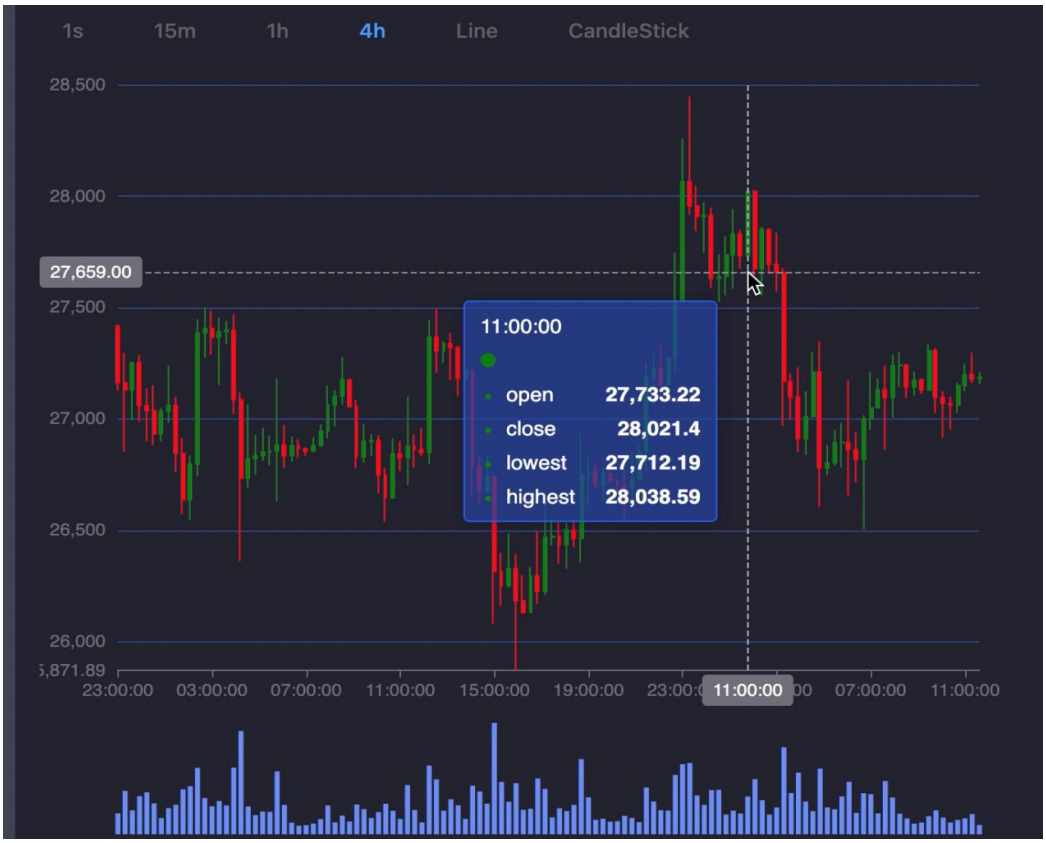

Рисунок 3.12 – Графік японські свічки

Нижче під графіком, користувач може побачити таблицю з криптовалютами, рис. 3.13, в данному випадку виводяться найбільші крпитовалюти за об’ємом, цю таблицю можна сортувати, а також шукати потрібну користувачу криптовалюту за назвою у спеціальному для цього полі вводу.

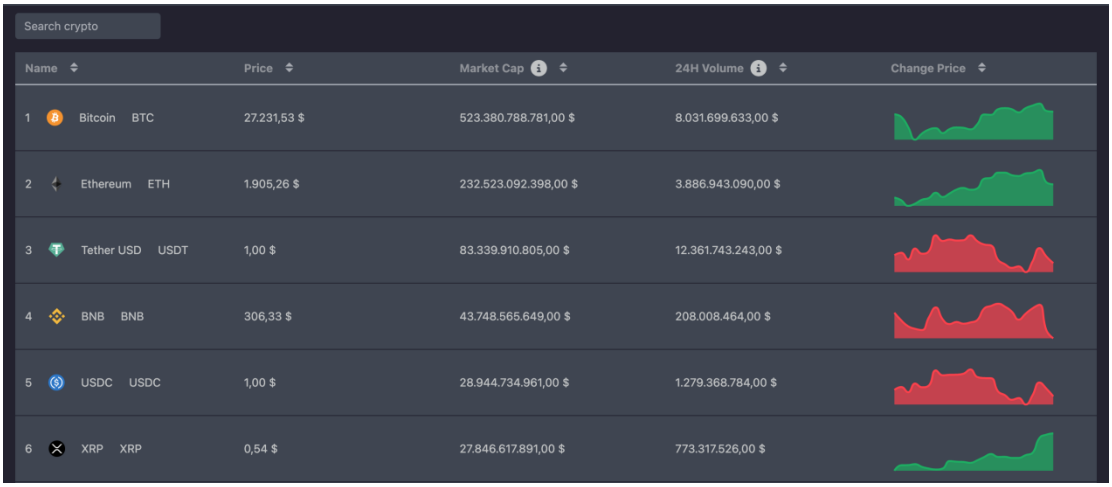

Рисунок 3.13 – Таблиця криптовалют

Таблиця досить велика, тому для комфорту користувача була добавлена пагінація. Пагінація веб-сторінок - це процес розбиття великої кількості контенту на окремі сторінки або блоки з метою полегшення навігації для користувачів, переглянути пагінацію можна на рис. 3.14.

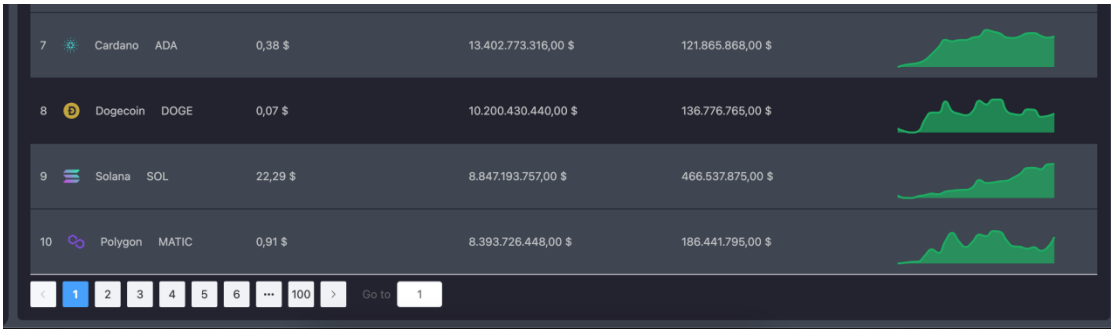

Рисунок 3.14 – Пагінація

Зліва на сайті розміщений сайдбар зображений на рис. 3.15, який дозволяє переміщатись на сайті, перша кнопка - це головна сторінка, друга - це настройки користувача, третя - це вихід з сайту.

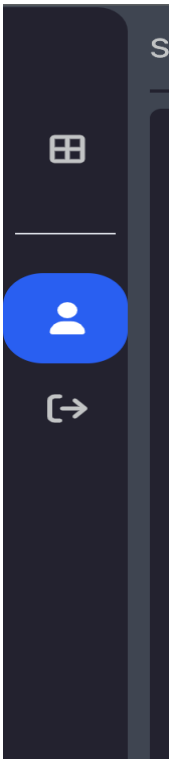

Рисунок 3.15 – Сайдбар

На сторінці користувацькі налашнування є дві кнопки, рис 3.16, які дозволяють переключатися між контентом, на першій табі можна змінити дані про користувача, зокрема ім’я, логін, номер телефону.

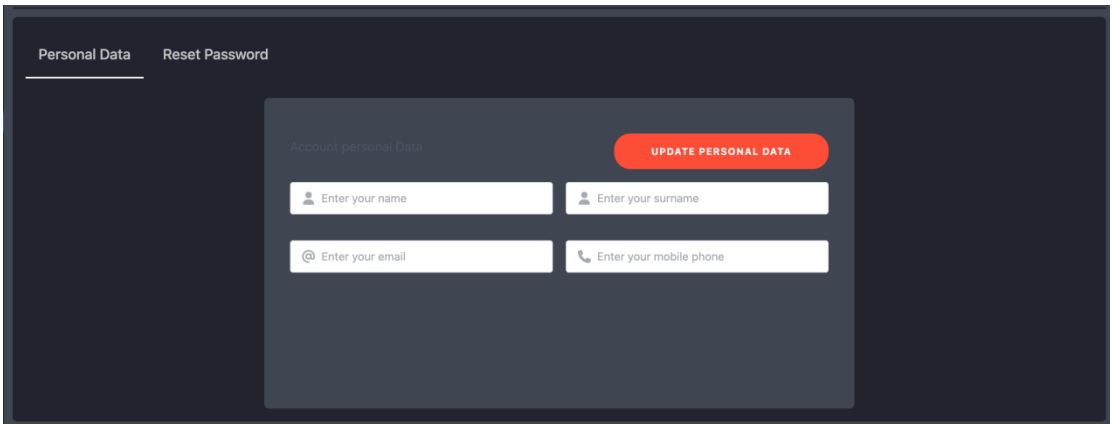

Рисунок 3.16 – Зміна даних про користувача

При кліці по кнопці “Reset Password”, користувача, переадресує на форму із можливістю змінювати пароль, як на рис. 3.17.

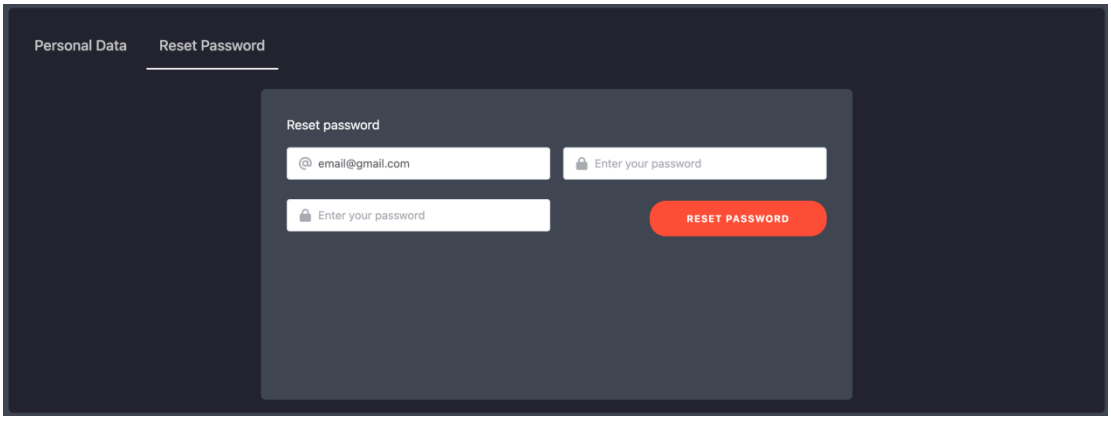

Рисунок 3.17 – Зміна паролю

Це увесь дизайн сайту. Варто зазначити, що оновлення графіку, даних в карточках чи в таблиці, відбувається щоразу коли користувач заходить на сайт, тобто якщо користувач перейде на іншу вкладку в браузері і вернеться назад, то відбудеться оновлення усіх даних.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Соціальне значення охорони праці

Охорона праці відіграє важливу соціальну роль, оскільки ніякі трудові здобутки не можуть відновити втрачене здоров'я або життя. Нещасні випадки та аварії на робочому місці мають великий соціальний вплив, оскільки поряд з робітниками та службовцями, на них впливають їх родичі, друзі та сім'я. Саме тому створення сприятливих та безпечних умов праці є необхідним фактором економічного та соціального розвитку держави.

Процес вступу України до Європейського союзу, ставить досить складні економічні, політичні, соціальні та інші завдання, серед яких важливим є формування відносин з працівниками на засадах соціальної відповідальності, а це означає, що ми повинні навчитися вибудовувати ефективні, демократичні та правові соціально-трудові відносини, які б, сприяли відновленню та зростанню економіки країни, а з іншого боку, надавали кожній особі почуття власної значимості, гідності, дотримання прав та свобод, гарантували гідний рівень життя та створювали умови для всебічного розвитку особистості. Необхідним для цього є запозичення та впровадження досвіду європейських країн у сфері використовуваних ними стандартів охорони праці. Соціальне значення охорони праці полягає в сприянні зростанню ефективності суспільного виробництва шляхом безперервного вдосконалення і поліпшення умов праці, підвищення її безпеки, зниження виробничого травматизму і захворюваності, зменшенні робочих місць, які не відповідають вимогам охорони праці [16].

У зв’язку з цим соціальне значення охорони праці виявляється у трьох основних показниках: – зростанні продуктивності праці в результаті збільшення

фонду робочого часу за рахунок скорочення внутрішньозмінних простоїв; – скорочення цілоденних втрат робочого часу; збереження трудових ресурсів і підвищення професійної активності працівників завдяки поліпшення їх стану здоров'я; – підвищення професійного рівня; збільшення валового національного продукту за рахунок поліпшення зазначених вище показників і їх компонентів [16]. Сама ж охорона праці відіграє важливу роль як соціальний чинник, оскільки якими б вагомими не були виробничі результати, вони не можуть компенсувати людині втрачене здоров'я, а тим більше життя. Соціальне значення охорони праці передбачає забезпечення всебічного соціального розвитку кожної працюючої особи, захист особи, також передбачає визнання пріоритету життя та здоров'я людини у процесі виробничої та трудової діяльності. До соціального змісту охорони праці належить запобігання шкідливим наслідкам, до яких може призвести ігнорування вимог техніки безпеки та гігієни праці на виробництві. До сфери соціального партнерства з охорони праці входять досягнення спільного рішення з питань застосування найманої праці та надання освітніх послуг із дотриманням техніки безпеки, вимог до охорони здоров'я у трудовому процесі під час навчання, забезпечення нормального режиму діяльності й відпочинку, соціального страхування, встановлення порядку проведення колективних переговорів, вирішення колективних трудових спорів тощо. Вищенаведені фактори чинять певний моральний і матеріальний тиск на роботодавця, що змушує його постійно й систематично займатися питаннями охорони праці. Однак цей тиск, як і приписи державних інспекцій та численні нормативні акти, не матимуть ефекту, якщо роботодавець не створюватиме гідні умови праці, особливу атмосферу довіри, чесності, вірності слову, порядності в стосунках із персоналом, зневажатиме закони і традиції, тобто певні неписані правила поведінки і дій. У такій ситуації на перший план виступає проблема підвищення рівня соціальної відповідальності всіх суб'єктів освітньої діяльності – держави, роботодавців, працівників і споживачів освітніх послуг відповідно до їх повноважень. Здатність відчувати відповідальність не лише за власне життя, й за життя тих, хто поруч, – доволі непросте завдання. Але якщо суспільство справді

прагне розвиватися відповідно до декларованих європейських стандартів, то повинне щоденно діяти за найвищими соціальними принципами толерантності, взаємодопомоги й ініціативності [15]. На даний час в Україні склалися досить скептичні погляди на євроінтеграцію.

Покращення національної системи соціально-трудових відносин та соціальної політики шляхом застосування європейських стандартів у сфері охорони праці надасть змогу сформувати відносини з працівниками на засадах соціальної відповідальності, створити гідні умови праці, забезпечити соціальний захист працівників та їх сімей. В свою чергу це позитивно вплине на ефективність роботи підприємств та їх фінансові результати діяльності, а відтак буде сприяти економічному зростанню країни, її просуванню вперед на шляху до євроінтеграції та виходу країни на рівень високорозвинених держав. Для українських підприємств застосування європейських стандартів охорони праці надасть можливості сформувати відносини з працівниками на засадах корпоративної соціальної відповідальності та соціального партнерства; підвищать відповідальність підприємств перед всіма зацікавленими сторонами, такими як: споживачі, підприємці, працівники, громада, держава та іншими; допоможе закріпити свої позиції на ринку праці, підвищить конкурентоспроможність підприємств; підвищить продуктивність праці робітників, покращить зацікавленість персоналу в діяльності підприємства та збільшить результативність їх роботи.

4.2 Менеджмент безпеки життєдіяльності

Діяльність людини завжди була пов’язана із майстерністю розумного управління виробничим процесом. Одним з найголовніших аспектів управління є безпека людини. Менеджмент системи безпеки – це система наукових знань та досвіду забезпечення безпеки, втілених у діяльності професійних управлінців для досягнення цілей системи безпеки через використання праці, інтелекту та мотивів поведінки людей.

Основна мета менеджменту безпеки полягає в тому, щоб створити безпечне та здорове робоче середовище, де ризики мінімізовані, а працівники можуть працювати ефективно та продуктивно. Для досягнення цієї мети, менеджмент безпеки включає такі основні етапи:

- оцінка ризиків: Визначення потенційних небезпек та оцінка ймовірності їх виникнення. Це включає ідентифікацію потенційних загроз, аналіз їх впливу та визначення рівня ризику.

- розробка стратегій безпеки: Визначення конкретних заходів та політик, спрямованих на запобігання та зменшення ризиків. Це може включати впровадження стандартів безпеки, навчання та навчальні програми, застосування безпечних технологій та обладнання, а також розробку екстрених планів дій;

- виконання та контроль: Реалізація запланованих заходів та постійний моніторинг їх ефективності. Це включає навчання працівників з питань безпеки, проведення інспекцій та аудитів безпеки, а також збір та аналіз даних щодо нещасних випадків та випадків порушення правил безпеки;

- постійне вдосконалення: Оцінка результатів та внесення необхідних змін для покращення системи безпеки. Це включає аналіз нещасних випадків та інцидентів, здійснення корекційних заходів, впровадження нових технологій та методів безпеки, а також впровадження культури безпеки серед працівників;

Робота менеджерів безпеки полягає у тому, щоб поєднати та скоординувати використання зазначених ресурсів (людські, фінансові, фізичні, інформаційні) для досягнення цілей забезпечення безпеки, ОГ.

До законодавчої та науково-методичної бази менеджменту безпеки належить перш за все Конституція України, законодавчі й підзаконні акти, що регламентують правовідносини у сфері підприємництва і безпеки. Такі як: Закон України “Про захист інформації в автоматизованих системах”, Закон України “Про підприємства в Україні”, закон України “Про службу безпеки України”. А також норми і положення статуту підприємства, що стосуються безпеки діяльності.

Системи менеджменту охорони праці і промислової безпеки базуються на стандартах, які чітко визначають процес досягнення безперервного поліпшення роботи з охорони праці і здоров'я, а також виконання вимог законодавства. СМОПіПБ відповідно до вимог OH5A5 18001 - це система менеджменту, що дозволяє оцінити виробничі небезпеки, ідентифікувати пов'язані з ними ризики і ефективно управляти ними. Унаслідок впровадження СМОПіПБ можливості виникнення аварійних ситуацій зводяться до мінімуму, знижуються виробничі ризики, забезпечується належний рівень охорони здоров'я персоналу і дотримання техніки безпеки на робочих місцях. OH5A5 18001 є дійсно світовим стандартом в тому сенсі, що його застосування не обмежується тільки організаціями в економічно високорозвинених країнах. У багатьох країнах керівництва компаній дійшли висновку, що такий стандарт є важливим для компанії і для її взаємин з суспільством і урядом, оскільки дозволяє створити систему управління безпекою. [15,17]

Створюючи систему, засновану на принципах OH5A5 18001, організація не зазнає труднощів в дотриманні правил і знижує ризик бути оштрафованою або піддатися судовому розгляду в разі виникнення травм, професійних захворювань і нещасних випадків. Правильне впровадження і підтримка в робочому стані системи управління охороною здоров'я і безпеки персоналу може бути частиною стратегії належної виробничої практики, яка є ефективним довгостроковим вкладенням засобів у майбутнє компанії [16]. Це, у свою чергу, веде до того, що компанії, які отримали сертифікати на системи управління охороною здоров'я і безпекою персоналу, вимагають від своїх субпідрядників, щоб вони також контролювали процеси і управляли ризиками у сфері охорони здоров'я і безпеки персоналу. Тому поява менеджменту безпеки як окремого напрямку менеджменту є досить логічною та абсолютно зрозумілою. Він допомагає запобігати нещасним випадкам, зберігати здоров'я та життя працівників, знижувати витрати, пов'язані зі здоров'ям та безпекою, та створювати позитивну репутацію організації в очах співробітників та громадськості.

ВИСНОВКИ

Комп'ютеризована система моніторингу та візуалізації вартості криптовалют є важливим інструментом для інвесторів, трейдерів та всіх зацікавлених осіб, які мають інтерес до криптовалютного ринку. Ця система забезпечує можливість в реальному часі стежити за цінами і рухами різних криптовалют, а також аналізувати їхній графік та історичні дані. Метою цього дипломного проекту було розроблення веб-додатку для моніторингу курсу цін та візуалізації криптовалют. Попередній аналіз засобів розроблення такої програмної системи у вигляді веб-додатку показав доцільність створення системи у такому форматі. Розроблена програмна система має наступні особливості:

– надає візуалізацію цін криптовалют Bitcoin, Etherereum та Litecoin;

– повідомляє про останні новини у світі криптовалют;

– має зрозумілий та зручний дизайн, придатний для різних браузерів та пристроїв.

Програмне забезпечення реалізовано повністю, всі функціональні та нефункціональні вимоги виконані. Використання розробленої системи надасть трейдерам та криптовалютним інвесторам хороший інструмент.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yatsyshyn V., Pastukh O., Palamar A., Zharovskyi R. Technology of relational database management systems performance evaluation during computer systems design. Scientific Journal of TNTU.Tern.: TNTU. 2023. Vol 109. No 1. P. 54–65.

2. Yatsyshyn V., Pastukh O., Zharovskyi R., Shabliy N. Software tool for productivity metrics measure of relational database management system. Mathematical Modeling. No 1 (48). 2023. P. 7-17.

3. Yatsyshyn V., Pastukh O., Lutskiv A., Tsymbalistyy V. ,Martsenko N. A Risks management method based on the quality requirements communication method in Agile approaches. CEUR Workshop Proceedings. 2022. Vol. 3309. Information Technologies: Theoretical and Applied Problems (ITTAP 2022) : proc. of the 2nd Intern. Workshop,Ternopil, Ukraine, November 22-24, 2022. pp. 1-10

4. Yatsyshyn V., Kharchenko O., Lutskiv A. Maturity Requirements Model for Software Requirements with the Implementation of ISO/IEC 25010 Recommendations. International Journal "Information Models and Analyses" Volume 9, Number 2, 2020 p. 126-143.

5. Yatsyshyn V., Pundyk V., Medvid I. Using Onlizer as efficient and productive tool at the software life cycle stages. VI Międzynarodowa konferencja studentów oraz doktorantów „Inżynier XXI wieku”.2016. P. 201-209.

6. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп’ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

7. Документація Tailwind CSS. [Електронний ресурс] – Режим доступу: https://tailwindcss.com/docs/installation.

8. Документація Vue 3. [Електронний ресурс] – Режим доступу: https://vuejs.org/guide/introduction.html.

9. Документація Firebase. [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/auth.

10. Документація Scss. [Електронний ресурс] – Режим доступу: https://sass-lang.com/documentation/.

11. Документація TypeScript. [Електронний ресурс] – Режим доступу: https://www.typescriptlang.org/docs/.

12. Документація JavaScript. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

13. Документація Element plust. [Електронний ресурс] – Режим доступу: https://element-plus.org/en-US/guide/quickstart.html.

14. Керб Л. П. Основи охорони праці: навч. посібник. Вид. 2-ге, без змін [Електронний ресурс]. К.: КНЕУ. 216 с. Режим доступу: masters.donntu.edu.ua/2011/iem/zemlyakova/diss/indexe.htm.

15. Криворучко Н. Соціальна відповідальність як основа розвитку українського суспільства. [Електронний ресурс]. Режим доступу: http://politiko.ua/blogpost59132.

16. Скобло Ю.С., Соколовська Т.Б., Морозенко Д.І. та ін. Безпека життєдіяльності. Навчальний посібник для вищих навчальних закладів 3-4 рівнів акредитації. – К.: Кондор, 2003. 424 с.

17. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці. К.: Каравела, 2007. 408 с.

18. Грибан В.Г., Негодченко О.В. Охорона праці. – К.: Центр учбової літератури, 2009. 209 с.

19. Бедрій І.Я., Нечай В.Я. Безпека життєдіяльності. Навчальний посібник. Львів: Манголія 2006, 2007. 499 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"**Затверджую"**

Завідувач кафедри КС

_____ Осухівська Г.М.

"___" _____2023 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА МОНІТОРИНГУ ТА ВІЗУАЛІЗАЦІЇ ВАРТОСТІ КРИПТОВАЛЮТ

ТЕХНІЧНЕ ЗАВДАННЯ

на _6_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»

Керівник кваліфікаційної роботи

_____ к.т.н., доцент Яцишин В.В.

«____» _____ 2023 р.

«ВИКОНАВЕЦЬ»

Студент групи СІ-41

_____ Демиденко А. О.

«____» _____ 2023 р.

Тернопіль 2023

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютеризована система візуалізації та вартості криптовалют».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.039.00.00

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Демиденко Антон Олександрович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-238 від 28.02.2023 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 28.02.2022 р.

Плановий термін завершення виконання кваліфікаційної роботи – 23.06.2022 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Система моніторингу та візуалізації, що розробляється призначена для:

– Забезпечення моніторингу за криптовалютами,
– Забезпечення візуалізації криптовалют,
– Зберігання обраних криптовалют.

2.2 Мета створення системи

Метою кваліфікаційної роботи є розробка системи моніторингу та візуалізації криптовалют на базі цифрових технологій з використанням технололій Vue та Firebase.

2.3 Характеристика об'єкту

Об'єкт, для якого розробляється дана система, є веб-додаток. Згідно з планом система має можливість зберігати дані про користувача та обрані ним криптовалюти, а також відображати їх у веб-додатку.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

3.1.1 Вимоги до структури та функціонування системи

Комп’ютерна система моніторингу та візуалізації повинна складатись з:

- Таблиці;
- Графіку;
- Можливості входити та виходити з свого акаунту.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Основна вимога, яка ставиться до способів та засобів інформаційного обміну – це їх узгодженість.

3.1.3 Перспективи розвитку, проектування системи

Дана система може бути розширена завдяки передбачененій технології Firebase.

3.2 Показники призначення

Система повинна передбачати можливість масштабування. Можливості масштабування повинні забезпечуватися засобами використовуваного базового програмного і технічного забезпечення.

3.2.1 Вимоги до надійності

Система повинна забезпечувати працездатність та відновлення своїх функцій при виникненні наступних ситуацій:

– при втраті зв’язку з API бірж;

3.3 Вимоги до захисту інформації від несанкціонованого доступу

Система повинна забезпечувати захист від несанкціонованого доступу Компоненти підсистеми захисту повинні забезпечувати:

– ідентифікацію користувача;

– перевірку повноважень користувача при роботі з системою;

Рівень захищеності від несанкціонованого доступу засобів обчислювальної техніки, що здійснюють обробку конфіденційної інформації, повинен відповідати вимогам класу захищеності згідно вимогам документу “Засоби обчислювальної техніки.

3.3.1 Вимоги по стандартизації і уніфікації

Система повинна відповідати вимогам ергономіки і зручності користування за умови комплектування високоякісним обладнанням (ЕОМ, монітор і інше обладнання), що має необхідні сертифікати відповідності і безпеки.

3.3.2 Вимоги до функцій (завдань), що виконуються системою:

– забезпечення якісного зображення;
– забезпечення умов зберігання обраних криптовалют;
– забезпечення високої швидкодії
– забезпечення постійного оновлення даних;

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;
– графічного матеріалу:

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка та аналіз вимог технічного завдання	28.02-19.02.2023
2	Аналіз сучасного стану досліджень у сфері аналізу криптовалюти	19.02-05.03.2023
3	Аналіз технологій розробки веб-додатків та мов програмування	05.03-26.03.2023
4	Обґрунтування вибору апаратного забезпечення	26.03-01.04.2023
5	Реалізація програмного забезпечення комп'ютеризованої системи моніторингу та візуалізації вартості криптовалют	01.04-10.05.2023
6	Безпека життєдіяльності, основи охорони праці	10.05-18.05.2023
7	Оформлення кваліфікаційної роботи	18.05-06.06.2023
8	Попередній захист кваліфікаційної роботи	06.06-18.06.2023
9	Захист кваліфікаційної роботи	21.06-23.06.2023

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Фрагменти лістингу програмного забезпечення комп'ютеризованої системи та візуалізації вартості криптовалют

```
<template>
      <div class="flex bg-mako flex-col gap-3">
        <Suspense>
          <dashboard-crypto-cards />
          <template #fallback>
            <div class="flex gap-4 w-full">
              <u-wrapper-card v-for="i in 4" :key="i" class="w-1/2">
                <u-skeleton-card />
              </u-wrapper-card>
            </div>
          </template>
        </Suspense>
        <dashboard-real-time-chart class="w-[calc(46%-_-17px)]" />
        <div class="flex gap-2 w-full items-start">
          <dashboard-real-time-chart class="w-1/2" />
          <DashboardCryptoNews class="w-1/2 h-[614px]" />
        </div>
        <u-wrapper-card class="!w-full">
          <dashboard-table />
        </u-wrapper-card>
      </div>
    </template>

    <script lang="ts" setup>
    import { defineAsyncComponent } from "vue";
    import USkeletonCard from "@/components/ui/skeleton/USkeletonCard.vue";
    import USkeletonTable from "@/components/ui/skeleton/USkeletonTable.vue";
    import UWrapperCard from "@/components/ui/UWrapperCard.vue";
    import ULoader from "@/components/ui/ULoader.vue";
    import DashboardCryptoNews from
    "@/components/dashboard/news/DashboardCryptoNews.vue";

    const DashboardCryptoCards = defineAsyncComponent({
      loader: () =>
        import("@/components/dashboard/crypto-cards/DashboardCryptoCards.vue"),
      errorComponent: USkeletonCard,
    });
```

```
const DashboardTable = defineAsyncComponent({
  loader: () => import("@/components/dashboard/table/DashboardTable.vue"),
  errorComponent: USkeletonTable,
});

const DashboardRealTimeChart = defineAsyncComponent({
  loader: () =>
    import("@/components/dashboard/chart/DashboardChartInRealTime.vue"),
  errorComponent: ULoader,
});
</script>
<template>
  <div class="flex items-center w-full justify-between dashboard-cards-
swiper">
    <template v-for="(swiperCard, index) in swiperCardsData"
:key="swiperCard">
      <swiper
        v-if="swiperCard"
        :pagination="{ clickable: true }"
        :modules="[Pagination]"
        class="!w-[25%] max-w-[330px] h-[240px]"
      >
        <swiper-slide v-for="card in swiperCard" :key="card?.name">
          <dashboard-swiper-crypto-card
            :title="swiperCardsTitle[index]"
            :card="card"
          />
        </swiper-slide>
      </swiper>
      <div v-else class="!w-1/2 h-[240px]">
        <u-wrapper-card
          class="flex flex-col h-full justify-center items-center relative w-
full items-center gap-4"
        >
          <u-loader />
        </u-wrapper-card>
      </div>
    </template>
  </div>
</template>

<script setup lang="ts">
```

```
import { computed, defineAsyncComponent, onMounted } from "vue";
import { Swiper, SwiperSlide } from "swiper/vue";
import { Pagination } from "swiper";
import "swiper/css";
import "swiper/css/pagination";
import { storeToRefs } from "pinia";
import { useCryptoWatchListStore } from "@/store/useCryptoWatchListStore";
import USkeletonCard from "@/components/ui/skeleton/USkeletonCard.vue";
import ULoader from "@/components/ui/ULoader.vue";
import UWrapperCard from "@/components/ui/UWrapperCard.vue";
import { useQueryCryptoCoinData } from "@/composable/useQueryCryptoCoinData";
import { useI18n } from "vue-i18n";

const DashboardSwiperCryptoCard = defineAsyncComponent({
  loader: () =>
    import("@/components/dashboard/crypto-
cards/DashboardSwiperCryptoCard.vue"),
  errorComponent: USkeletonCard,
});

const cryptoWatchListStore = useCryptoWatchListStore();
const { cryptoWatchlist } = storeToRefs(cryptoWatchListStore);

const { data: trending } = useQueryCryptoCoinData().queryGetTopTradesCoins();
const { data: losers } = useQueryCryptoCoinData().queryLosersCoins("losers");
const { data: gainers } =
useQueryCryptoCoinData().queryGainersCoins("gainers");

const swiperCardsData = computed(() => [
  losers?.value,
  gainers?.value,
  trending?.value,
  cryptoWatchlist.value,
]);

const { t } = useI18n();

const swiperCardsTitle = computed(() => {
  return [
    t("global.dashboard.cards.lose"),
    t("global.dashboard.cards.top"),
    t("global.dashboard.cards.trend"),
```

```
      t("global.dashboard.cards.watchlist"),
    ];
  });
  onMounted(async () => {
    await cryptoWatchListStore.fetchWatchList();
  });
  </script>

  <style lang="scss" scoped>
  .dashboard-cards-swiper {
    .swiper {
      margin: 0;
      &-pagination {
        bottom: 0 !important;
        &-bullet {
          background: var(--silver-sand);
        }
      }
    }
  }
  </style>
```

```
<template>
    <u-wrapper-card
      class="flex h-full flex-col relative w-full items-center gap-4"
    >
      <p>{{ title }}</p>
      <el-tooltip v-if="Object.keys(card).length" effect="light"
  placement="top">
        <template #content>
          <p v-if="!isInWatchList">{{ t("global.dashboard.cards.add") }}</p>
          <p v-else>{{ t("global.dashboard.cards.remove") }}</p>
        </template>
        <u-toggle-icons
          :active-icon="Star"
          :inactive-icon="StarFilled"
          :value="isInWatchList"
          @active-icon-event="addItemsToWatchList"
          @inactive-icon-event="removeItemsFromWatchList"
```

```
      />
    </el-tooltip>
    <div v-if="!Object.keys(card).length">
      <p class="text-center mt-2">
        {{ t("global.dashboard.cards.watchlist_empty") }}
      </p>
    </div>
    <div v-else class="flex items-center gap-3 justify-between pb-[20px]">
      <dashboard-crypto-card-info
        :name="card?.name"
        :symbol="card?.symbol"
        :price="card?.price"
        :change="card?.change"
        :icon-url="card?.iconUrl"
      />
      <u-small-line-chart
        class="w-[120px] h-[120px]"
        :data="card?.sparkline"
        :change-price="card?.change"
      />
    </div>
  </u-wrapper-card>
</template>

<script setup lang="ts">
import DashboardCryptoCardInfo from "@/components/dashboard/crypto-
cards/DashboardCryptoCardInfo.vue";
import USmallLineChart from "@/components/ui/charts/USmallLineChart.vue";
import UWrapperCard from "@/components/ui/UWrapperCard.vue";
import { useToggle } from "@/composable/useToggle";
import { useCryptoWatchListStore } from "@/store/useCryptoWatchListStore";
import { CoinrankingResponse } from "@/types/api/coinRanking";
import { computed } from "vue";
import UToggleIcons from "@/components/ui/icons/UToggleIcons.vue";
import { Star, StarFilled } from "@element-plus/icons-vue";
import { useI18n } from "vue-i18n";

interface DashboardSwiperCryptoCardProps {
  title: string;
  card: CoinrankingResponse;
}
```

```
const { t } = useI18n();

const props = defineProps<DashboardSwiperCryptoCardProps>();

const [watchlistClick, toggleWatchlist] = useToggle(false);

const { checkIfItemIsInWatchList } = useCryptoWatchListStore();

const { addNewItemToWatchList, removeItemFromWatchList } =
  useCryptoWatchListStore();

const isInWatchList = computed(() => {
  return checkIfItemIsInWatchList(props?.card);
});

const addItemsToWatchList = () => {
  toggleWatchlist();
  addNewItemToWatchList(props.card);
};

const removeItemsFromWatchList = () => {
  toggleWatchlist();
  removeItemFromWatchList(props.card);
};
</script>
```