НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система для верифікації історичних подій»

Виконала:
студентка IV курсу, групи ІА-з91
Литвиненко Ольга Богданівна _____

Керівник:
Старший викладач кафедри ІСТ
Тимофєєва Юлія Сергіївна _____

Рецензент:
Асистент кафедри ОТ
Каплунов Артем Володимирович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студентці
Литвиненко Ользі Богданівні

1. Тема проєкту «Інформаційна система для верифікації історичних подій», керівник проєкту Тимофєєва Юлія Сергієвна старший викладач кафедри ІСТ, затверджені наказом по університету від «31» травня 2023 р. №2102-с

2. Термін подання студентом проєкту 12 червня 2023 р.

3. Вихідні дані до проєкту: мова програмування Rust, мова програмування TypeScript

4. Зміст пояснювальної записки: аналіз предметної області, постановка проблеми, огляд існуючих рішень, проектування системи, вибір технологій та реалізація та тестування.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): UML-діаграма станів, UML-діаграма класів, UML-діаграма використання, UML-діаграма послідовності.

6. Дата видачі завдання: 01.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Аналіз предметної області	03.02.2023	
2.	Огляд існуючих рішень	12.02.2023	
3.	Вибір технологій та особливості реалізації	20.02.2023	
4.	Розробка on-chain частини	26.02.2023	
5.	Розробка off-chain частини	07.03.2023	
6.	Тестування	02.04.2023	
7.	Виконання графічних документів	21.04.2023	
8.	Оформлення пояснювальної записки	17.05.2023	
9.	Подання ДП на основний захист	11.06.2023	

Студент Ольга ЛИТВИНЕНКО

Керівник Юлія ТИМОФЄЄВА

АНОТАЦІЯ

Литвиненко О.Б. Інформаційна система для верифікації історичних подій КПІ ім. Ігоря Сікорського, Київ, 2023.

Проєкт містить 77 с. тексту, 13 рисунків, 4 таблиці, посилання на 29 літературних джерела, додатки та 4 конструкторські документи.

Ключові слова: історична подія, доказ історичної правди, архів, Solana, Blockchain, dApp.

Об’єктом розробки є історична правда.

Мета розробки — покращення системи верифікаціїї правдивості історичних подій за допомогою створення універсального інструменту, який надаватиме можливість швидкого доступу до інформації про певні історичні події та пов'язані з ними докази правдивості.

В результаті виконання дипломного проєкту поставлена мета була виконана: був проведений детальний аналіз проблеми, прописані алгоритми верифікації подій та розроблена двокомпонентна інформаційна система для верифікації історичних подій: on-chain та off-chain частини. Дана система може бути використана в едукаційних цілях та в науковій сфері для архівації та перевірки правдивості історичних подій, також був наданий функціонал для інтеграцій інших продуктів в систему, що надасть можливості розширення екосистеми.

SUMMARY

Information system for verification of historical events of Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023.

The project contains 77 pages of text, 13 figures, 4 tables, references to 29 literary sources, appendices and 4 design documents.

Keywords: historical event, proof of historical truth, archive, Solana, Blockchain, dApp.

The object of development is historical truth.

The purpose of the development is to improve the system of verification of the truthfulness of historical events by creating a universal tool that will provide quick access to information about certain historical events and related evidence of truthfulness.

As a result of the thesis project, the goal was achieved: a detailed analysis of the problem was conducted, event verification algorithms were prescribed, and a two-component information system for verifying historical events was developed: on-chain and off-chain parts. This system can be used for educational purposes and in the scientific field to archive and verify the veracity of historical events, and the functionality for integrating other products into the system was provided, which will provide opportunities to expand the ecosystem

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІАз91.010БАК.004 ПЗ	Пояснювальна записка	77		
6	А3	ІАз91.010БАК.004 Д1	Інформаційна система для	1		
7			верифікації історичних подій.			
8			Діаграма класів			
9	А3	ІАз91.010БАК.004 Д2	Інформаційна система для	1		
10			верифікації історичних подій.			
11			Діаграма використання			
12	А3	ІАз91.010БАК.004 Д3	Інформаційна система для	1		
13			верифікації історичних подій.			
14			Діаграма станів			
15	А3	ІАз91.010БАК.004 Д4	Інформаційна система для	1		
16			верифікації історичних подій.			
17			Діаграма послідовності			
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІАз91.010БАК.004 ТП		
Розроб.		Литвиненко			Інформаційна система для верифікації історичних подій. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Тимофєєва				Т	
Затв.						КПІ ім. Ігоря Сікорського Група ІАз91	

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система для верифікації історичних подій»

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

Термін	Пояснення
ПК	Приватний ключ
PDA	Program Derived Address — це тип акаунтів в Solana, який створений контрактом і належить йому, а не конкретному користувачеві або акаунту.
lamports	Мінімальна одиниця власного токена Solana SOL, котра дорінює 0.000000001 SOL
dApps	Застосунок, котрий базується на Blockchain
ID	Унікальний ідентифікатор
UI	User Interface — користувацький інтерфейс
IDL	Мова опису інтерфейсів, надалі під IDL матиметься на увазі саме файл із інтерфейсами
Деплой	Розгортання програмного забезпечення
Валідаційна нода	Вузол мережі Solana, надалі може використовуватися в скороченій формі – «нода»
Валідація	Перевірка заданих вимог ПЗ
ПЗ	Програмне забезпечення
Ініціалізація	Створення програмного об’єкту
Препроцессінг	Попередня обробка даних
Сериалізація	Переведення даних в послідовність байтів
Маппінг	Зв’язок між елементами різних наборів даних
Флоу	Послідовність виконання інструкцій або операцій в програмі
Дата акаунт	Тип акаунтів в Solana, котрі є невиконуваними та створені для збереження інформації

ВСТУП

Кожного дня ми стикаємося з потребою пошуку інформації, тож у сучасному світі, завдяки широкого доступу до мережі інтернет, пересічна людина кожного дня обробляє шалену кількість інформації, що надходить з різних джерел. Іноді важко сформувати думку щодо конкретної проблеми, вирішити, якому інформаційному джерелу довіряти, зважувати об'єктивність та правдивість отриманої інформації.

Частіше за все у зв’язку з обмеженістю часу та енергії при пошуку інформації пересічний користувач, не зважаючи на кількість доступних ресурсів, обирає з них певну кількість, якій сумлінно довіряє, для того, щоб не перевантажувати себе зайвими даними. Такий підхід дозволяє швидше розібратися в певній темі, але водночас звужує інформаційне поле, що дуже часто негативно впливає на правдивість отриманої інформації. Часом така стратегія може завести користувача в пастку — адже джерела інформації можуть по різному трактувати події, зважаючи на свої мотиви та погляди, таким чином прищеплюючи людині спотворене бачення фактів.

Дана проблема хвилює мене в контексті сприйняття суспільством певних історичних подій. Дуже часто основним джерелом пізнання історичних явиш є книжки, статті, портали з новинами і т.д. Обробляючи їх людина сумлінно вірить прочитаному, не перевіряючи інформацію на правдивість. Немає точної гарантії, що інформація, надана джерелом має високий ступінь правдивості, навіть якщо її авторство належить видатному вченому. Адже кожна подія може інтерпретуватися по різному, в залежності від поглядів, впливу оточення та мотивації автора, але існує багато протилежних думок інших вчених, котрі могли б перекреслити доказовість наданої інформації.

На жаль у швидкому темпі життя сучасної людини дуже рідко знайдеться час, щоб посвятити себе повністю пошуку істини щодо певного історичного явища, котрим вона цікавиться. Тож в умовах сьогодення доводиться покладатися обране коло джерел, сподіваючись на їх сумлінну правдивість.

Одним зі способів розв'язання цієї проблеми є створення інформаційної системи, що буде універсальним архівом для збереження конкретних історичних фактів, доказів та думок різних експертів щодо певної історичної події.

Перевагою цієї системи є наявність всієї інформації в одному місці — користувачам не доведеться шукати інформацію в великій кількості різних джерел та перевіряти їх на правдивість, адже уся інформація буде надана в одному ресурсі, це допоможе зберегти значну кількість часу і збільшить ймовірність отримання доказової інформації.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення та класифікація історичних подій

Згідно «Словнику української мови» історична подія — це певне явище, що відбувалося в минулому, невигадане, котре існувало насправді та відповідає реальній дійсності [2][3]. Воно має вагому історичну значущість і впливає на подальший розвиток подій у певному часовому періоді. Історичні події можуть бути пов'язані з політикою, економікою, соціальними аспектами, науковими відкриттями, війнами, конфліктами та іншими суспільними явищами.

Історична подія не є незалежною одиницею історії вона складається з послідовних пов’язаних між собою історичних фактів. Історичні факти необхідні для виявлення і визначення історичних зв'язків та компонування у загальні історичні явища. Обробляючи та систематизуючи зібрані історичні факти можливо досконально відтворити процеси певної епохи. Історичні події будуються з історичних фактів, а сукупність цих подій формує історичні процеси, явища, котрі впливають на подальший розвиток людства і є невіддільною частиною пізнання історії.

Раніше вважалося, що сам по собі факт не може бути визначений без наявного доказового носія, котрий містить докази правдивості та чітко визначене місце в хронології. Таким носієм є історичні джерела. Історичне джерело — це таке джерело інформації, що може бути використане для одержання інформації про минуле людського суспільства [4].

Історичним джерелом може бути будь-який матеріал (письмовий, архівний, археологічний, монетний, фотографічний тощо), який містить інформацію про події, які відбулися в минулому. Ці матеріали є доказами того, що деякі історичні факти дійсно сталися, і вони є важливими джерелами для реконструкції історії.

Але часом наукова спільнота може ставити під сумність правдивість певного історичного факту, наводячи докази фальсифікації певного історичного факту. В такому випадку обґрунтування неправдивості авторитетними експертами із

посиланнями на інші події минулого може вважатися спростуванням історичного факту, що саме по собі теж може бути історичним фактом.

Так склалося, що дуже багато фактів залишилися не зафіксованими — вони не мали матеріального обґрунтування або ж ці докази не збереглися до наших днів. Історія має досить багато прогалин, не підкріплених носіями історичних джерел. В XX столітті трактування поняття історичного факту змінилося, відтепер сприйняття історичного факту стало більш багатогранне — воно включає не тільки підкріплені матеріальними доказами факти, але й ті факти, котрі виникли в процесі дослідження певної невивченої області історії. Під час поглиблених досліджень історичної епохи вчені можуть запропонувати теоретичний конструкт, котрий хронологічно та за наслідками міг би заповнити пропущену прогалину історії. Такий конструкт може бути затверджений науковою спільнотою, а отже може вважатися історичним фактом.

Історичні факти можуть бути досить теоретичними, не відтворювальними минулу реальність достеменно, але вони мають виконувати одну функцію — реконструювати історичну подію таким чином, щоб її критичні кульмінаційні події та результат події відповідав історичній правді. Враховуючи вищесказане, можна класифікувати історичні факти на:

– об’єктивні події минулих часів, що мають точні просторово-часові рамки та матеріальні докази свого існування;

– інтерпретації цих фактів науковою спільнотою (адже часом ми не можемо гарантувати справжність події, тож думка експерта щодо певного джерела теж має значення);

– наукові факти, що з’являються в процесі історичних досліджень [5];

– варто зазначити, що в даній роботі я звузила масштаб пізнання історичного минулого до сприйняття його як сукупності хронологічно послідовних та пов’язаних історичних подій — поняття епох, періодів і т.д були упущені — весь плин історії був декомпонований в розрізі однієї одиниці – події;

Історична подія є свого роду узагальненням — об’єднанням сукупності історичних фактів, джерел та експертних думок. Говорячи про структуру даної

інформаційної системи, можна сказати що така інкапсуляція набору даних, котрі слугують доказом історичної правди, буде потребувати детального планування архітектури системи, адже необхідно буде представити усі сутності таким чином, щоб один історичний факт міг слугувати підтвердженням дня однієї чи більше історичних подій, при цьому важливо виключити дублювання цього факту.

1.2 Спотворення історичної правди

Історична правда — це об'єктивне існування фактів і подій, що відбулися в минулому та мають доказову базу в теперішньому часі. Це означає, що історична правда не залежить від особистих переконань, емоцій або ідеології дослідника, а базується на об'єктивних свідченнях, що можуть бути перевірені або спростовані [6]. Однак історична правда часто переплітається з історичними наративами, які є суб'єктивними інтерпретаціями історичних подій. На ці наративи можуть впливати різні фактори, такі як політичні, релігійні та націоналістичні інтереси.

Історичні викривлення виникають, коли історичні свідчення або наративи змінюються відповідно до особистих або групових інтересів, що включає дезінформацію та брехню. Існує кілька способів спотворення історичної правди:

– вибіркове представлення фактів: зосередження уваги на конкретних фактах чи подіях, ігноруючи інші, може призвести до упередженого або спотвореного розуміння історії. Це може бути зроблено навмисно для підтримки певного наративу або ненавмисно через обмеженість інформації чи знань;

– неправильне тлумачення фактів: історичні факти можуть бути неправильно витлумачені, навмисно або випадково;

– маніпуляції з доказами: докази, такі як документи, артефакти абосвідчення, можуть бути підтасовані, сфабриковані або знищені для створення спотвореної версії історичної правди;

– вплив особистих або групових інтересів: політичні, релігійні танаціоналістичні інтереси можуть впливати на спосіб представлення історії, що призводить до упередженого або спотвореного наративу;

Одним із прикладів впливу історичних наративів на міжнародні відносини є відмінності у трактуванні геноциду вірмен турками та вірменами [7]. Інший приклад — контрастні наративи палестинських арабів та ізраїльтян щодо арабо-ізраїльського конфлікту [8].

Історичну правду, у розумінні Зиґмунда Фройда, можна визначити як втрачену частину життєвого досвіду суб'єкта, яка доступна лише через роботу конструювання. Таким чином, з психоаналітичної точки зору, історична правда — це істина послідовності, а не моменту; вона вимагає реконструкції етапів, що ведуть до конституювання елементу, який може претендувати на статус істини [9]. А отже сприйняття історичної правди не має бути одновимірним — доказовість історичної правди має покладатися на сукупність фактів, думок експертів та доказів правдивості.

Тож історична правда — це об'єктивна розповідь про минулі події, тоді як історичні наративи — це суб'єктивні інтерпретації, на які можуть впливати різні чинники, що призводить до викривлень. Для збереження цілісності історичної правди дуже важливо усвідомлювати ці потенційні викривлення і дотримуватися принципів, які сприяють об'єктивному і заснованому на фактах розумінню історії.

1.3 Дослідження критеріїв правдивості історичної події

Доведення правдивості історичної події може бути складним процесом, оскільки він залежить від наявності достовірних джерел і свідчень про цю подію. Історик, який досліджує певну історичну подію, зазвичай спирається на різноманітні джерела, такі як архівні документи, листування, спогади, газетні статті та інші. Установи часто використовують історичні методи дослідження для збору фактів, хронологічних даних та іншої інформації, що відповідає їхнім інтересам. Вони оцінюють історичні події та забезпечують належний контекст, всебічно використовуючи первинні та вторинні джерела.

Іншим способом доведення правдивості історичної події є порівняння даних з різних джерел та перевірка їх узгодженості. Якщо дані з декількох джерел

збігаються, то імовірність правдивості історичної події збільшується. Історичні методи дослідження також включають процедури роботи з суперечливими джерелами. Коли два джерела не погоджуються з певним питанням, історик віддасть перевагу джерелу з найбільшим "авторитетом" — тобто джерелу, створеному експертом або очевидцем. Якщо два незалежно створені джерела збігаються в думках з певного питання, надійність кожного з них помітно підвищується.

Загалом, доведення правдивості історичної події є складним процесом, історики зазвичай використовують різні методи для перевірки достовірності джерел і даних, а також звертають увагу на контекст та інші фактори, які можуть впливати на інтерпретацію події. Важливо розрізняти історичну правду та історичні наративи. Історична правда є об'єктивною і може бути виявлена шляхом емпіричного дослідження, в той час, як історичні наративи є суб'єктивними інтерпретаціями подій, на які може впливати точка зору групи або нації [14].

Підтвердження історичної події як правдивої передбачає поєднання критики джерел, історичної аргументації, врахування авторського порядку денного, доказових артефактів і джерел, а також міждисциплінарних підходів. Це зазвичай здійснюється науковими інституціями, такими як університети, історичні музеї, архіви, дослідницькі центри, історичні товариства та інші подібні організації.

Зазвичай вони проводять дослідження історичних джерел, архівних документів, археологічних розкопок та інших джерел, щоб встановити правдивість певної історичної події. Також урядові організації, такі як історичні комісії або департаменти культури, можуть займатися доведенням правдивості історичних подій, які мають важливе значення для держави або місцевої громади.

У багатьох країнах є національні інститути, які займаються збереженням та дослідженням історичної спадщини країни, які можуть бути також відповідальні за доведення правдивості історичних подій.

Варто зазначити, що будь-яка установа має право проводити історичні дослідження, але вірогідність того чи дане дослідження буде котуватися в науковій спільноті, чи вважатиметься пророблена наукова робота об’єктивною та

ступінь довіри до авторів роботи напряму залежить від авторитетності даної організації в науковому світі.

1.4 Постановка проблеми

Дуже важливою навичкою сьогодення став аналіз та розуміння інформації з точки зору її об'єктивності, доказовості та значимості — тобто навичка критичного мислення. Цей процес можна розвивати шляхом вдосконалення своїх вмінь фільтрування та оцінки прочитаних або побачених фактів з різноманітних джерел з різними позиціями щодо інформаційної проблеми. Але на шляху людини до критичного аналізу стоїть саме проблема збору інформації.

На початку 21 століття була зроблена оцінка, що кожен день людство виробляє близько 2.5 ексабайт інформації, а ця кількість зростає з кожним роком [1]. Наш мозок має обмежену кількість робочої пам'яті, тобто кількість інформації, яку можна зберегти і одночасно обробляти. Коли перевищується цей обсяг, мозок стає перевантаженим, і це може призвести до зниження продуктивності та якості прийнятих рішень. Тож, при пошуку інформації пересічний користувач, не зважаючи на кількість доступних ресурсів, обирає з них певну кількість, якій сумлінно довіряє, для того, щоб не перевантажувати себе зайвими даними. Іноді такий підхід є вдалим, адже людина економить свій час і енергію, обмежуючи коло ресурсів, з яких черпає інформацію, вона не перевантажує себе зайвим, тож має менше аналізувати, фільтрувати отримані дані, а також перевіряти їх на правдивість.

Опитування, проведене в червні 2020 року, показало статистику перевірки людьми прочитаної інформації в мережі: частіше за все люди перевіряли правдивість інформації, шукаючи доказів в інших медіа(43%), коментарях (27%) або намагалися знайти оригінальну статтю автора (39%) або інші статті з того ж самого джерела (32%), проте варто зазначити, що перевірка фактів (27%) та дослідження іх правдивості (4%) використовувалися рідше [15].

Але в розрізі пізнання історії такий підхід є контрпродуктивним — адже економлячи час і енергію на пошуки інформації ми жертвуємо правдивістю. Як було зазначено в попередніх розділах історія сама по собі є досить неоднозначною наукою, історичні події можуть бути інтерпретовані по-різному залежно від перспективи, з якої їх досліджувати. Одна і та ж подія може мати різні описи залежно від культурного, соціального, економічного, політичного та іншого контекстів. Це може призводити до пізнання подій на основі неповного або некоректного інформаційного поля, адже маючи обмежену кількість довірених ресурсів, людина складає свою суб'єктивну думку на основі того обмеженого обсягу прочитаного, що може бути інтерпретоване різними способами і не завжди бути правдивим.

На жаль, попри доступність розмаїття інформаційних джерел, які зарекомендували свою експертизу в науковій спільноті, або ж першоджерел, основна маса користувачів інтернету частіше за все шукає доказів історичної правди на сайтах з новинами та історичних порталах, у вікіпедії, форумах та соціальних мережах.

Згідно зі звітом про дослідження споживання новин і медіаграмотності в усьому світі, телебачення було найбільш типовим джерелом, з якого споживачі отримували новини та інформацію про поточні події в усіх країнах-учасницях. . На рисунку 1.1 можна побачити статистику найбільш уживаних джерел пошуку інформації щодо подій за 2020 рік [16]. Незважаючи на те, що споживання новин різнилося в усьому світі, соціальні мережі були названі джерелом новин для понад 50 відсотків споживачів з кожної країни в дослідженні, а онлайн-газети, сайти новин або програми також були популярним вибором.

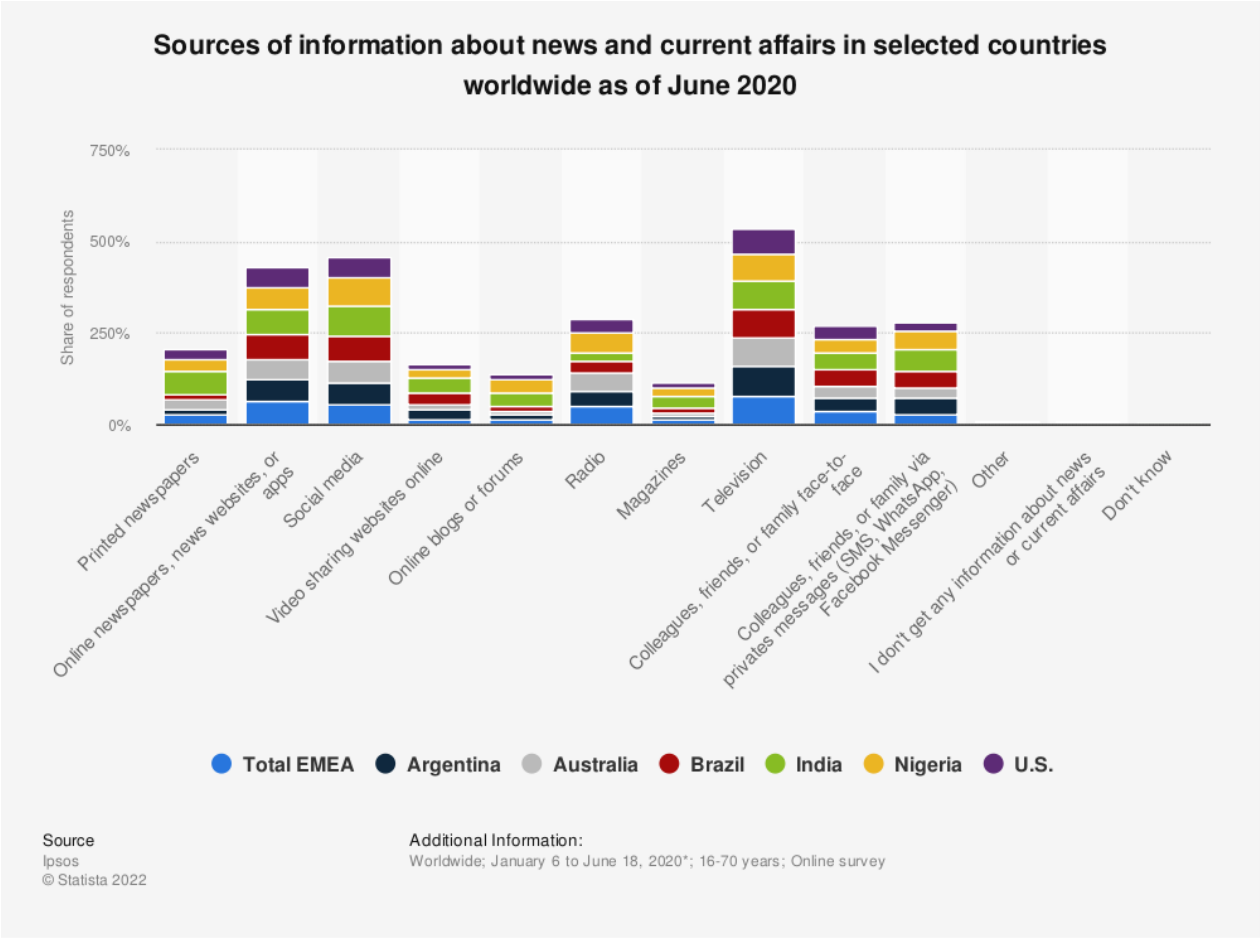

Рисунок 1.1 — Статистика найбільш уживаних джерел для пошуку інформації щодо подій [16]

Вищезгадані джерела є менш надійні, оскільки часто залежать від політичних та економічних інтересів, що може призвести до спотворення історичних фактів. Важливо користуватися різноманітними джерелами та перевіряти інформацію на правдивість, користуючись допомогою документальних та інших авторитетних джерел, наприклад історичні архіви, книги, журнали та наукові статті від визнаних експертів.

Найбільш продуктивним методом для пошуку історичної істини була б обробка усіх можливих джерел та думок щодо неї. Але у більшості людей немає часу та мотивації займатися таким глибоким пошуком. Навіть застосовуючи технології OSINT доведеться вкласти велику кількість часу для засвоєння та аналізу шуканої інформації. А зараз ціна часу занадто велика, тож людина продовжує звужувати спектр інформаційних джерел і формує свій світогляд на

основі думок невеликого кола спеціалістів, не перевіряючи правдивість, незалежність та доказовість їх думок. Як наслідок, історична правда залишається досить химерною, адаптованою для кожного по-своєму, спотворена призмою поглядів інших людей. Тож досить складним та актуальним є питання встановлення історичної правди в такому різноманітті ресурсів.

Одним зі способів розв'язання цієї проблеми я бачу створення інформаційної системи, що буде універсальним архівом для збереження конкретних історичних фактів, доказів та думок різних експертів. Таким чином в одному місці можна зібрати усі історичні необхідні джерела, тож читач матиме змогу проаналізувати обмежений обсяг даних від авторитетних джерел, економлячи свій час на пошуки інформації та знаючи, що дана інформація не є викривленою сторонніми факторами, а походить безпосередньо від експертного джерела.

1.5 Огляд готових рішень

Існують різні програмні рішення для архівації історичних подій. Ось кілька прикладів з основних категорій:

Спеціалізовані історичні системи управління даними: існують програмні рішення, спеціально розроблені для архівації історичних подій і даних. Наприклад, PastPerfect є популярною системою управління колекціями для музеїв, архівів та історичних організацій. Вона дозволяє зберігати інформацію про артефакти, фотографії, документи та інші матеріали, пов'язані з історичними подіями. Ця система допомагає зберегти цифрові копії об'єктів, фотографії, документів та іншої інформації, забезпечуючи довготривалу доступність інформації і полегшуючи спільну роботу з колекціями. Загалом, PastPerfect є потужним інструментом для організації, управління та дослідження колекцій культурної спадщини і використовується в багатьох установах по всьому світу.

Переваги:

– організація та доступ до даних: PastPerfect допомагає структурувати колекцію, надаючи можливість класифікувати об'єкти за різними категоріями, такими як тип об'єкту, рік створення, донор, місце знаходження тощо;

– інтеграція фотографій та документів: PastPerfect дозволяє додавати фотографії, скановані зображення та документи до записів про об'єкти. Це дозволяє зберігати та відображати візуальну інформацію, що допомагає в ідентифікації та дослідженні об'єктів;

Недоліки:

– централізоване керування: PastPerfect дозволяє централізовано зберігати та керувати інформацією про колекції. Мінус такого підходу — вірогідність повної втрати даних при відсутності бекапів у випадку видалення локального сховища, також він має обмежені можливості для роботи через веб-інтерфейс. Це може бути незручним, особливо якщо необхідно мати дистанційний доступ до системи або спільно працювати з колегами;

– вартість: PastPerfect є комерційним програмним забезпеченням, і його вартість може бути високою, особливо для менших організацій з обмеженим бюджетом;

– підтримка та оновлення: залежно від версії та підписки, підтримка та оновлення PastPerfect можуть бути обмеженими, до того ж він не має зворотньої сумісності;

Цифрові архівні системи: ці системи спеціалізуються на зберіганні та управлінні цифровими матеріалами, такими як електронні документи, відео, фотографії тощо. Наприклад, Archivematica є відкритим програмним забезпеченням для автоматизованої обробки цифрових архівних записів. Деякі цифрові архівні системи також надають функції для опису та метаданих, що дозволяють зберігати інформацію про історичні події.

Переваги:

– автоматизація процесів: Archivematica надає інструменти для автоматизації процесів зберігання, обробки та доступу до цифрових архівних матеріалів. Вона дозволяє автоматично виконувати приймання, перевірку

цілісності, розпізнавання форматів, обробку метаданих, нормалізацію форматів, створення архівних пакетів, створення резервних копій, забезпечення доступу до архівів, що спрощує процес управління архівами;

– розширені можливості зберігання: Archivematica підтримує різні формати архівних матеріалів та надає можливість зберігати їх, відповідаючи вимогам стандартів довготривалого зберігання;

Недоліки:

– високі вимоги до обладнання та інфраструктури: Archivematica вимагає потужного обладнання та добре налаштованої інфраструктури для ефективної роботи;

– потребує експертизи та навчання: використання Archivematica вимагає розуміння архівної практики та знання технічних аспектів цифрового зберігання;

– централізоване керування: Archivematica переважно призначена для роботи на одному комп'ютері або в локальній мережі. Це може обмежувати можливості спільної роботи та доступу до архівних матеріалів з різних місць, а також існує небезпека втрати даних;

Електронні журнали та Blockchain: Деякі організації використовують електронні журнали та Blockchain для архівації історичних подій. Наприклад Everipedia — це децентралізована онлайн-енциклопедія, яка використовує Blockchain технологію. Вона дозволяє користувачам створювати та редагувати статті, а всі зміни записуються на Blockchain, що забезпечує незмінність історичних даних. Everipedia не має централізованого контролю над своїм вмістом, і будь-який користувач може редагувати або додавати статті без обмежень.

Переваги:

– децентралізованість: Everipedia використовує технологію Blockchain для забезпечення децентралізованого контролю та управління;

– глобальний доступ: Everipedia доступна для всіх користувачів, цедозволяє залучати різноманітність перспектив та знань з різних країн і культур;

– стимулювання користувачів: Everipedia використовує криптовалюту IQ Token для винагороди користувачів, які вносять внесок у створення та покращення контенту. Це може стимулювати активну участь та якість матеріалів;

Недоліки:

– відсутність стандартизованих джерел: Everipedia дозволяє редагування статей користувачами, що може призвести до впровадження недостовірної або неперевіреної інформації;

– залежність від активності користувачів: якщо недостатня кількість користувачів активно долучається до редагування та покращення контенту, це може призвести до відсутності актуальної та достовірної інформації;

Всі вищезгадані системи мають широкий функціонал, але здебільшого спеціалізуються саме на архівації подій. Враховуючи вищезазначену інформацію, дані системи мають низку недоліків, що не можуть гарантувати високий ступінь безпеки, незалежності та об’єктивності інформації.

1.6 Опис вимог та функцій системи

На основі наданої інформації можна сформулювати список вимог до системи:

– відсутність централізації: система не повинна залежати від одного керуючого органу;

– унеможливлення видалення інформації: система не повинна зберігатися на обмеженій кількості носіїв, повинна бути відкритою для завантаження бажаючими;

– доступність даних: будь який користувач мережі може читати дані з системи безкоштовно і без високих технічних вимог до обладнання;

– довіра авторитетним фахівцям: тільки науковці із відповідними досягненнями та авторитетом можуть вносити та редагувати дані про події;

– наявність механізму валідації внесених даних: відображення думки наукової спільноти щодо внесеної інформації в систему;

– механізми заохочення та можливість інтеграцій: наявність технічної бази для введення системи заохочення науковців та можливості інтеграцій інших рішень із даною системою;

На основі вищезазначених вимог можна сформулювати функції системи:

– створення асоціативної групи вчених;

– створення репортера;

– редагування репортера.

– створення фактів.

– редагування фактів;

– зв’язування події та факту;

– створення доказу;

– редагування доказу;

– оцінка об’єктивності доказу;

– демонстрація усіх даних про подію.

Висновки до розділу 1

В даному розділі було розглянуто основні поняття обраної предметної області, розглянуто проблему спотворення історичної правди та методи доведення її правдивості. Також була сформульована основна проблема — вибіркове сприйняття інформації людиною у зв’язку із обмеженістю ресурсів (часу, енергії і т.д) та доступних джерел, а також проблема відсутності інструментів фільтрації фіктивної інформації в області історичних досліджень. Був проведений огляд та аналіз існуючих рішень та сформульоване можливе рішення проблеми — створення єдиної децентралізаваної інформаційної системи для валідації історичних подій.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис сутностей

В цьому розділі буде розглянуто основні сутності ініформаційної системи, описано основні функції, складові та вимоги до неї. Враховуючи інформацію із попередніх розділів можна сказати, що для верифікації історичної події необхідний набір сутностей, котрий буде прописаний в цьому розділі.

2.1.1 Історична подія

Історична подія обов’язково має своє чітко визначене місце в просторі та часі, є унікальною за своїм характером і наслідками, оскільки кожна подія відбувається в певному контексті, з участю унікальних людей та обставин. Навіть якщо дві події можуть бути схожими в певних аспектах, їх можна розрізнити за специфічними деталями та відмінностями, що впливають на їхні наслідки та історичну значущість.

Для віддзеркалення повноти значення певної історичної події в історії людства необхідно мати розгорнутий опис цієї події, її передумови та наслідки. Історична важливість події може бути визначена з різних причин, таких як її вплив на суспільство, політичні або культурні наслідки, значення для розвитку науки та технологій, або значення для розвитку людських цінностей та моралі — все це має бути в описі події.

Узагальнюючи усе вищесказане можна визначити такі вимоги до історичної події:

– повинна мати чітко визначені часово-просторові рамки;

– кожна історична подія має бути унікальною;

– подія може бути визначена як історична тільки якщо має значущі наслідки в історії людства;

– подія повинна мати мінімум один доказ правдивості;

– право створення сутності історичної події може мати тільки експерт, котрий зарекомендував свою авторитетність в науковій спільноті

2.1.2 Історичний факт

Історичний факт буде слугувати базовою одиницею доказовості історичної правди. Так само як і подія, він має чітко визначені часово-просторові рамки та повинен бути унікальним. Він має містити змістовний опис сукупності явищ, котрі сформували передумови, наслідки та перебіг подій, котрі призвели до певного історичного факту.

Необхідно відмітити, що історичний факт може доводити правдивість не тільки однієї події, а і сукупності подій, адже історичні явища тісно переплітаються один з одним. Також варто зазначити, що історичні факти можуть заперечувати один одного, оскільки вони можуть тлумачитись та розумітись по-різному залежно від джерел, контексту, інтерпретації та культурного середовища, в якому вони були створені.

Наприклад, історики можуть мати різні точки зору на питання, пов'язані з подіями, які сталися в минулому, і їхні тлумачення фактів можуть відрізнятися або інформація, яка зберігається про певні історичні події, може бути неповною або неправильною, тому інтерпретації тих подій можуть давати різні результати. Тож необхідно брати до уваги не тільки сам історичний факт, але також огляди та оцінки інших науковців щодо нього — тобто на докази його правдивості. Враховуючи усе вищесказане можна сказати, що сутність наукового факту повинна відповідати таким умовам:

– повинна мати чітко визначені часово-просторові рамки;

– кожен факт має бути унікальним;

– певне явище може бути визначеним як історичний факт тільки тоді, коли він є частиною чогось глобального — тобто події (однієї або більше) ;

– факти можуть мати докази своєї правдивості/хибності від інших науковців;

– право створення сутності історичного факту може мати науковець, котрий зарекомендував свою авторитетність в науковій спільноті та той, що проводить дослідження історичних явищ пов’язаних із даним фактом.

2.1.3 Доказ історичного факту

Дана сутність створена для акумулювання думок різних експертів щодо певного історичного факту. Доказом може бути як підтвердження правдивості історичного факту, так і його заперечення. Адже дуже часто історичні факти народжуються в процесі дослідження певного історичного процесу, і часом вони можуть не збігатися з висновками інших експертів. Варто зауважити, що право створення доказу має надаватися науковцю, що довів свою експертизу в тій галузі, котра охоплює матеріал покладений в основу створення доказу. Це не означає, що фахівці інших галузей, не будуть мати право створювати доказ, але бажано, щоб науковець був компетентний в обраній тематиці, адже один із фактів, котрий буде визначати об'єктивність доказу — це авторитет та компетентність науковця, що його представив.

Тож доказ історичного факту повинен містити докладний опис причин, пов'язаних фактів та історичних явищ, котрі будуть слугувати доказом правдивості або ж хибності обраного історичного факту. Також важливо вказати, що не всі експерти можуть погоджуватися з представленими доказами, тож необхідно ввести лічильник оцінок об’єктивності доказу, котрий буде показувати кількість згодних науковців. Знаючи все вищесказане, можемо сформувати основні вимоги до сутності:

– доводити правдивість або ж хибність факту;

– мати чітку аргументацію доказу: опис причин, пов'язаних фактів та історичних явищ;

– бути унікальним та послідовним серед інших доказів;

– право створення сутності історичного факту може мати науковець, котрий зарекомендував свою авторитетність в науковій спільноті та той, що проводить дослідження історичних явищ пов’язаних із даним фактом.

2.1.4 Історичний зв’язок

Історичний зв'язок — це допоміжна сутність, створена для репрезентації історичної пов’язаності глобального явища певної історичної події та фактом, що доводить її правдивість. Дана сутність є необхідною з причин оптимізації, адже історичний факт не є прив’язаним до однієї історичної події, він може бути доказом правдивості одразу декількох подій.

На приклад є дві події: розпад СРСР та відродження незалежності України, також є факт — підписання Біловезької угоди, котрий констатує припинення існування СРСР, але в той же час є передумовою відродження незалежності України [23].

Тобто судячи з цього можна зробити висновок, що факт і подія це не взаємозалежні, а самостійні сутності, між якими існує зв'язок. Тож виділення цього зв’язку як окремої сутності є необхідним при побудові архітектури системи.

Також варто зауважити, що одна подія може бути пов’язана з великою кількістю фактів, тобто мати велику кількість історичних зв'язків, тож для упорядкування цих фактів варто ввести лічильник пов’язаних зв'язків у сутності події, а також мати унікальний ID у сутності історичного зв’язку, щоб легко знайти його серед інших. Також важливо вказати, що не всі експерти можуть погоджуватися з пов’язаністю певної події з фактом, тому необхідно ввести лічильник оцінок експертів, як і в історичному факті. Знаючи все вищесказане, можемо сформувати основні вимоги до сутності:

– мати можливість пов’язувати історичну подію та історичний факт між собою;

– бути унікальним та послідовним серед інших зв’язків.

2.1.5 Оцінка об’єктивності

Дана сутність створена для оцінювання об’єктивності історичного доказу щодо певного історичного факту або ж історичних зв’язків події з фактом. Оцінка може бути позитивною або негативною, що означає згоду з об’єктивністю доказу, або ж її заперечення. Логіка даного функціоналу дуже подібна до “лайку/дизлайку” в соціальних мережах, основна мета та ж — лаконічна репрезентація думок користувачів щодо конкретної інформації.

Оцінка доказу є досить обмеженою за функціональністю сутністю, але її наявність в системі необхідна. Адже важливо показати внутрішнє ставлення наукової спільноти до певного історичного доказу або зв’язку, бо аналізуючи подані оцінки кваліфікованих фахівців можна зробити висновки про правдивість певного доказу, тобто водночас підтвердити правдивість певного факту. Аналогічно із запереченнями — через негативні оцінки можна спростувати правдивість фактів і подій.

Дуже важливою є кількість цих оцінок — адже це розширює статистичну вибірку думок експертів і можна більш об’єктивно проаналізувати дані. Тому важливо ввести систему заохочування оцінювання (про це в наступному розділі). Знаючи все вищесказане, можемо сформувати основні вимоги до сутності:

– містити погодженість або ж незгоду із доказом;

– право створення сутності історичного факту може мати фахівець, котрий має відповідну експертизу, щоб свідчити про історичну правдивість та оперувати необхідними доказами.

2.1.6 Репортер

Це ключова сутність системи верифікації історичних подій, адже уся система перевірки правдивості заснована саме на думках експертів щодо фактів, подій та явищ, що відбувалися або діють і досі. Репортер — це абстракція сутності фахівця певної галузі, науковця, що має експертизу в сферах, котрі можуть бути

корисними при дослідженні фрагментів історії та доказів правдивості історичних явищ.

Ця сутність є критично важливою, адже покладаючись на думку експертів, користувач системи буде робити висновки про те, чи значущі певні події та факти, або ж вони зовсім не мали впливу на становлення світового порядку, або взагалі вони є фіктивними. Знаючи все вищесказане, можемо сформувати основні вимоги до сутності:

– бути репортером може незалежний фахівець з дослідженнями в історичних галузях. Тип цього репортера буде визначатись в залежності від експертизи науковця.

2.1.7 Асоціація

Основною мотивацією до включення даної сутності в систему була необхідність створення єдиної узагальнюючої сутності. Вводячи сутність асоціації можна показати, що репортери, котрі входять в якусь конкретну асоціацію і є коректними репортерами, їх думці можна довіряти. Відповідно власник асоціації буде мати право створювати репортерів всередині цієї асоціації, а також це корисна сутність, у випадку, якщо знадобиться бекап. — адже є можливість перенести дані на іншу асоціацію, без міграції абсолютно всіх акаунтів. Виходячи з цього можемо зробити висновок, що дана сутність має містити:

– унікальний ID асоціації;

– ПК власника асоціації;

Усі вищезазначені сутності є взаємопов’язаними та варто зазначити — що сутності доказу, оцінки та зв’язку не є незалежними. Вони залежать від пов’язаних із ними сутностей: доказ не може існувати без факту, оцінка без доказу та зв’язок без події та факту. На основі прописаних сутностей сформована діаграма класів та взаємозв’язків між ними в кресленику ІАз91.010БАК.004 Д1.

2.2 Опис процесів системи

Беручи до уваги усі вищеописані сутності, можемо сформулювати можливі процеси, що будуть відбуватися в системі. В креслинику ІАз91.010БАК.004 Д2 показана UML прецедентів системи, на котрій зображено усі можливі взаємодії учасників системи. Можна виділити 3 ключових акторів, котрі будуть взаємодіяти з системою:

– адмін;

– репортер;

– користувач;

Адмін: це користувач системи, котрий має право створювати асоціацію в системі та буде мати права для редагування даних всередині створеної асоціації. Відповідно він має право створювати репортера всередині асоціації та редагувати його статус.

Репортер: у сутностях, описаних вище, згадувались різні види спеціалістів: починаючи від рядових практикуючих науковців, закінчуючи всесвітньо визнаними експертами. Спираючись на компетенцію фахівця можна виділити певний діапазон дій, в котрих в нього буде експертиза та гарантія надійності. Наприклад науковий співробітник університету може мати недостатньо досвіду та знань, для глибокого аналізу та обґрунтування правдивості явищ, що призвели до певного історичного факту, але в нього вистачить експертизи для оцінки доказу зробленого більш компетентним науковцем. Таким чином можна виділити декілька категорій репортерів, кожна з яких буде мати обмежений діапазон дій, відповідний до їх компетенції:

– Validator — тип репортера, котрий може давати оцінку доказам історичних фактів. Статус такого репортера може отримати фахівець, що має науковий ступінь в області, котра охоплює дослідження історичного характеру, проводить активні дослідження в цій галузі та має авторитет серед фахівців вищого рівня;

– Connoisseur — тип репортера, котрий може створювати докази правдивості історичних фактів, а також давати їм оцінку. Статус такого репортера

може отримати науковець, що зарекомендував себе в науковому світі, зробив вагомий вклад в дослідження своєї галузі;

– Expert — найвищий тип репортера, котрий може робити всі можливі дії: створювати історичні події, факти та їх оцінку. Статус такого репортера може отримати тільки експерт з міжнародним визнанням, котрий зробив вагомий вклад в науку, компетенція котрого не ставиться під питання.

Досить важливим є процес заміни репортерів через певний проміжок часу, наприклад прибирати репортерів, що втрачають кваліфікацію і додавати нових, більш перспективних. Тож необхідно ввести поняття статусу репортера — репортер може мати 3 статуси:

– активний;

– дезактивований;

– заблокований.

При створенні репортеру надається статус активного. Після завершення зазначеного строку, котрий надається кожному репортеру, він деактивується. У випадку надання фіктивної інформації або негативних відгуків інших репортерів він може бути заблокований адміном.

Також досить важливим моментом є логіка покарання репортера за фіктивну інформацію. Часом репортери можуть навмисно чи ненароком викривлювати інформацію щодо історичного явища, тому важливо мати систему покарань. Тому можна ввести поняття штрафних балів. Якщо доказ або факт репортера набирає певну кількість негативних оцінок, то репортеру дається штрафний бал, якщо кількість цих балів перевищує вказану максимальну — репортер деактивується.

Описані механіки задають базові правила екосистеми, котрі допомагають регулювати діяльність діючих сторін. В кресленику ІАз91.010БАК.003 Д3 показана UML діаграма стану репортера, на котрій показано усі проміжні стани ініціалізованого репортера в системі під час виконання дій створення та модифікації асоціації. Перед виконанням команди репортер обов’язково має пройти такі етапи:

– етап ініціалізації — репортер підключає свої ключі до системи;

– перевірка статусу репортера — статус має бути активним, всі інші статуси не допускаються до подальшої взаємодії;

– перевірка штрафних очок — кількість штрафних очок не має перебільшувати вказану допустиму;

– визначення типу репортера — в залежності від цього буде наданий тільки певний діапазон дій.

Користувач: це людина, котра зацікавлена в пошуку інформації щодо певної історичної події. Доступна йому дія досить проста — пошук даних про історичну подію, але флоу всередині системи досить складний та послідовний. Якщо користувач хоче дізнатися точку зору наукового світу щодо історичної події він знаходить назву необхідної йому історичної події та шукає її в системі, в цей час мають відбуватися такі процеси:

– пошук даних про сутність події в системі;

– на основі отриманих даних пошук зв’язків події по унікальному ID на основі лічильника зв’язків;

– пошук відгуків про знайдені зв’язки;

– на основі даних знайдених зв’язків пошук пов’язаних фактів;

– пошук доказів знайдених фактів;

– пошук відгуків про знайдені докази.

Технічним завданням даної дипломної роботи є створення програмного продукту — двокомпонентної інформаційної системи для верифікації історичних подій. Було прийнято рішення декомпонувати систему на два окремих модулі, один з котрих буде реалізований в Blockchain . Основним аргументом на користь використання Blockchain застосунка можна назвати проблему архівів або ж інших інформаційних джерел — це можливість їх ліквідації. Відштовхуючись від цього система повинна складатися з двох частин: on-chain та off-chain частини.

On-chain частина повинна містити усі необхідні сутності, що були описані в розділах вище, котрі є критично важливими для доведення правдивості історичної події. Ця частина має бути детально спроектована та містити в собі усі необхідні перевірки.

Of-chain частина повинна слугувати для допоміжним інструментом саме для репортерів та містити увесь необхідний функціонал, котрий буде покривати увесь доступний спектр дій репортера. Ця частина буде клієнтом, котрий буде під'єднуватися до мережі Blockchain та надсилати необхідні транзакції.

2.3 Огляд та аналіз Blockchain платформ

Смарт контракт (також відомий як розумний контракт або smart contract) — це програмний код, який запускається на Blockchain і містить правила та умови для автоматичного виконання та контролю угод між сторонами без необхідності посередництва.

Нижче наведений перелік найпопулярніших платформ для смарт контрактів, які засвідчили свою ефективність та надійність в різних застосуваннях:

– Ethereum — це децентралізована Blockchain платформа першого рівня, тобто та, котра може проводити транзакції без участі іншої мережі. Це перший Blockchain, котрий дав можливість розробникам інтегрувати свої програмні додатки (DApps) в систему, яка запускає розумні контракти. Віртуальна машина Ethereum (EVM) — це середовище виконання цих смарт контрактів. EVM є повною 256-бітною віртуальною машиною Тьюринга. Смарт контракти на основі Ethereum написані за допомогою мови програмування Solidity, яка є Turing Complete, тобто вона дозволяє кодувати цикли і розгалуження. «Газ» є паливом розумних контрактів Ethereum, котрий прораховує кількість обчислювальної потужності, необхідної для виконання смартконтрактів через EVM. Він використовує концепцію Proof of Stake (PoS) — це різновид механізму консенсусу, який використовується для забезпечення безпеки Blockchain мереж — кожен блок мережі затверджується тільки тоді, коли достатня кількість валідаторів погоджується з тим, що кожна з транзакцій у блоці була виконана правильно[17];

– Near — це також Blockchain першого рівня, котрий використовує унікальну технологію шардинга, котра називається Nightshade для досягнення високого рівня масштабування. Він також надає можливість розробникам

інтегрувати свої програмні додатки в систему. Це мережа також на "Proof of Stake". Валідатори запускають обладнання, яке виконує функцію керування мережею, кожен з них отримує підтримку від стейкінг-пулу. Власники токенів з усієї екосистеми можуть делегувати свої токени будь-якому з цих пулів. Коли валідатори голосують за затвердження блоків, їхні голоси зважуються залежно від того, скільки токенів у них є в стейкінг-пулах. Варто зазначити, що екосистема Near розвивається дуже швидко і одна з основних цілей команди розробки платформи — зробити систему максимально доступною, тому вони розробляють “мости” до інших Blockchain [18];

– Solana — один з найшвидших Blockchain першого рівня — пропускна здатність мережі становить від 50 000 до 191 000 транзакцій на секунду. У Blockchain Solana є 8 основних інновацій, що відрізняють його від інших мереж:

1) Proof-of-History. Solana використовує комбінацію двох алгоритмів консенсусу: Proof-of-Stake (PoS) і розробленого спеціально для Solana Proof-of-History (PoH), в котрому різні ноди функціонують паралельно і незалежно одна від одної, але в один і той самий час синхронізуються між собою. Це дає змогу мережі підтримувати тисячі нод одночасно, пропорційно масштабуючи її пропускну здатність. Варто зазначити, що синхронізація нод — одна з основних проблем децентралізованих мереж. Щоб підтвердити операції, ноди повинні записати блок із ними в Blockchain. Для цього вони обмінюються даними, перевіряють транзакції і синхронізуються (досягають консенсусу). Заковика в тому, що на це йде багато часу, внаслідок чого страждає пропускна здатність мережі. Наприклад, максимальна швидкість Bitcoin — 7-10 операцій на секунду. Proof-of-History дозволяє нодам синхронізуватися набагато швидше. Так, якщо в інших Blockchain х консенсус досягається угодою майнерів або валідаторів, то у кожного валідатора в Solana є власний годинник. Proof-of-History являє собою децентралізований годинник, який синхронізує час у всіх нодах. Використання Proof-of-History служить доказом, що транзакція відбулася в конкретний момент часу в мережі. При цьому ноди Blockchain працюють за єдиним розкладом (Leader Schedule);

2) Tower BFT. Це оптимізована для Proof-of-History версія практичної візантійської відмовостійкості (pBFT, practical Byzantine fault tolerance), варіант алгоритму консенсусу на основі Proof-of-Stake (PoS). Тому формально Solana працює на PoS, але спирається на "децентралізований годинник" Proof-of-History. Це дає змогу досягати консенсусу, не витрачаючи часу і ресурсів на обчислення часових міток попередніх операцій і на обмін повідомленнями між усіма валідаторами;

3) протокол Gulf Stream. Це протокол, який пересилає транзакції наступним валідаторам-лідерам ще до завершення попереднього набору транзакцій, не переводячи їх до мемпулу (список транзакцій, які очікують підтвердження). У підсумку майбутні лідери можуть почати збирати транзакції до того, як почнуть виробляти блоки. Це збільшує швидкість роботи, а також скорочує розмір мемпула і самого Blockchain;

4) протокол Turbine — розбиває інформацію про операції на кілька частин;

5) алгоритм Sealevel — дозволяє паралельно виконувати кілька смарт контрактів;

6) конвеєризація (Pipelining) — прискорення перевірки транзакцій на рівні ядра графічного процесора. Це дає змогу швидко перевіряти інформацію про транзакції в усіх нодах мережі;

7) база даних облікових записів Cloudbreak — оптимізує використання пам'яті на пристроях валідаторів, не даючи змогу даним займати занадто багато місця;

8) архіватори — розподілене сховище. Валідатори завантажують дані в спеціальну мережу нод — так званих архіваторів (Archivers). Архіватори не беруть участі в досягненні консенсусу, оскільки кожен із них зберігає частину даних (batch), а не всі їх. Періодично архіватори повинні довести, що вони як і раніше зберігають свої дані[19];

Існують і інші Blockchain платформи, що дозволяють розробляти розумні контракти, такі як Wanchain, Aeternity, Ripple, Qtum, Ark, Neblio, Cardano, BSC,

Polcadot, і інші. Проте, мною було обрано саме ці враховуючи їхню актуальність, попит та нововведення, котрі вони принесли в криптоіндустрію.

Near, Solana та Ethereum — це Blockchain платформи, які пропонують унікальні функції та орієнтовані на різні сценарії використання. Ось порівняння цих платформ [20 - 22]:

В таблиці 2.1 наведені загальні характеристики: обраних Blockchain мереж, можна помітити, що по усім параметрам Solana має кращі результати, окрім кількості валідаторів, за рахунок того, що мержа досить молода та має специфічні вимоги до обладнання валідаторів.

Таблиця 2.1 — загальні характеристики Blockchain

Blockchain	Solana	Ethereum	Near
Транзакції за секунду	65 000	15	10 000
Плата за транзакцію	0.0015$	15$	0.01$
Маштабованість	Висока	Низька	Висока
Затримка транзакцій	0.4 сек	~ 5 хв	~ 5 сек
Кількість валідаторів	~ 2000	500 000 +	~ 300

Масштабованість:

– Near: протокол Near має на меті забезпечити високу масштабованість і низьку комісію за транзакції. Він використовує архітектуру розщепленого Blockchain для паралельної обробки великої кількості транзакцій;

– Solana: Solana розроблений для високої масштабованості та пропускної здатності. Її архітектура поєднує механізми Proof-of-History і Proof-of-Stake для досягнення швидкої обробки транзакцій і обробки тисяч транзакцій в секунду;

– Ethereum: Ethereum стикався з проблемами масштабування, особливо в періоди високого перевантаження мережі. Однак оновлення Ethereum 2.0, яке впроваджується поетапно, має на меті покращити масштабованість завдяки впровадженню механізму консенсусу Proof-of-Stake та шард-ланцюгів;

Механізм консенсусу:

– Near: протокол Near використовує механізм консенсусу Proof-of-Stake (PoS) під назвою "Nightshade" для захисту мережі і перевірки транзакцій;

– Solana: Solana поєднує механізми Proof-of-History (PoH) і Proof-of-Stake (PoS) для досягнення консенсусу і підтримки цілісності Blockchain;

– Ethereum: Наразі Ethereum використовує механізм консенсусу Proof-of-Work (PoW), але переходить на механізм консенсусу Proof-of-Stake (PoS) через оновлення Ethereum 2.0.

Функціональність смарт-контрактів:

– Near: Near Protocol підтримує смарт-контракти, написані на мовах програмування Rust і AssemblyScript;

– Solana: Solana підтримує смарт-контракти, написані на Rust та C++, і використовує мову транзакцій Solana (Solang) для розробки контрактів;

– Ethereum: Ethereum широко відомий своєю підтримкою смарт-контрактів. Він представив мову програмування Solidity, яка дозволяє розробникам писати контракти, що завершуються за Тьюрінгом.

Екосистема та впровадження:

– Near: протокол Near набув популярності серед розробників та проектів децентралізованих додатків (dApp). Він пропонує інструменти для розробників, бібліотеки та зростаючу екосистему додатків;

– Solana: Solana привертає увагу своєю швидкою обробкою транзакцій та масштабованістю. Вона набула поширення в різних сферах, включаючи додатки DeFi (децентралізовані фінанси) і ринки NFT (невзаємозамінних токенів) ;

– Ethereum: Ethereum має найбільшу екосистему та спільноту розробників серед трьох платформ. Він став платформою для численних dApps, ICO та зародження ринків DeFi та NFT.

Варто зауважити, що у кожної Blockchain платформи є свої переваги та недоліки, але враховуючи вищезазначену інформацію, можна сказати, що для обраної інформаційної системи найбільше підходить Solana, адже він швидкий, дешевий, надійний за рахунок механізмів консенсусу та здатності масштабуватись. Solana має не велику кількість валідаторів порівняно з Ethereum, але навідміну від нього є значно дешевшим та швидшим, також великою перевагою буде те, що Solana набирає популярність, тож кількість валідаторів буде збільшуватись із кожним днем.

2.4 Підбір технологій розробки

При виборі стеку технологій, на основі котрого буде написана інформаційна система, треба відштовхуватися від обраного Blockchain — від Solana. Смарт контракти у Solana називаються програмами. Rust C, C++ — це мови, що використовуються для створення програм, які розгортаються в мережі.

Для розробки смарт-контрактів на платформі Solana найпоширенішою мовою програмування є Rust. Solana надає офіційну підтримку для мови Rust і має добре розроблену екосистему для розробки смарт-контрактів на цій мові. Rust є мовою програмування з високою продуктивністю, низьким рівнем абстракції і вбудованими механізмами безпеки. Ці характеристики роблять її дуже популярною для розробки Blockchain додатків, включаючи смарт-контракти на Solana.

Rust має кілька переваг в розробці на платформі Solana:

– продуктивність: Rust є високопродуктивною мовою програмування, що дозволяє ефективно виконувати обчислення на Solana. Це особливо важливо для Blockchain додатків, де швидкість та ефективність грають важливу роль;

– безпека: Rust пропонує механізми безпеки на рівні мови, такі як власництво та позикування, які допомагають уникнути багатьох типових помилок програмування, таких як витоки пам'яті, перегони даних та недійсні вказівники, усі ці проблеми часто трапляються в C та C++. Це особливо корисно в контексті

розробки смарт-контрактів, де недоліки безпеки можуть призвести до серйозних наслідків, таких як втрата коштів або компрометація даних;

– системний рівень контролю: Rust надає розробникам прямий доступ до системних ресурсів, що дозволяє виконувати оптимізації та маніпулювати низько рівневими деталями. Це може бути корисним для розробки на Solana, де ефективне використання ресурсів мережі є важливим аспектом;

– екосистема та підтримка: Rust має активну спільноту розробників, що підтримує розвиток екосистеми інструментів та бібліотек для розробки на Solana;

– наявність прикладів в коді ядра: Solana написана на Rust і весь код добре задокументований, тому це є великою перевагою, адже можна черпати приклади із самої бібліотеки SDK;

Варто зазначити — розробка програм на Solana є досить складним процесом, адже варто враховувати усі можливі проблеми, прописувати усі перевірки, що обмежити діапазон дій людини, що викликає контракт, до того, що прописано в логіці. Тому доцільним є використання Anchor — це фреймворк для швидкого створення безпечних програм на Solana.

Anchor є високорівневою бібліотекою та фреймворком для розробки смарт-контрактів на платформі Solana. Використання Anchor має декілька переваг у контексті розробки Solana контрактів:

– простота використання: Anchor надає простий та зрозумілий інтерфейс для розробки смарт-контрактів. Він спрощує процес розробки, тестування та взаємодії з контрактами на Solana;

– високорівневість: Anchor дозволяє розробникам працювати на високорівневому рівні абстракції, приховуючи багато низькорівневих деталей розробки на Solana. Це полегшує розробку, зменшує кількість потенційних помилок і дозволяє швидше створювати функціональні контракти;

– тестування та симуляція: Anchor надає набір інструментів для тестування та симуляції смарт-контрактів на локальному середовищі. Це дозволяє швидко перевірити функціональність та коректність контрактів перед їх розгортанням на мережі Solana;

– підтримка: фреймвок підтримується командою розробників, ведеться активна розробка нових функцій та активна комунікація з розробниками екосистеми (будь яка людина може задати їм писання в Discord).

Тому буде доцільним обрати мову Rust з використанням фреймворку Anchor для on-chain частини на Solana для розробки інформаційної системи.

Взаємодіяти з вже розгорнутими програмами можна, написавши dApps за допомогою будь-якого з доступних клієнтських SDK (або CLI). На рисунку 2.1 показана взаємодія контрактів із клієнтськими доданками через JSON RPC API. Основою розробки додатків на Solana є JSON RPC API, який є рівнем комунікації, що дозволяє взаємодіяти з Blockchain. Solana Labs створила простий у використанні SDK solana-web3.js, який дозволяє спілкуватися з Blockchain і програмами Solana таким же чином, як і з будь-яким іншим API. Також снує багато сторонніх SDK, які також були створені на основі JSON RPC API, таких мовах як Rust, Java, C#, Python, Go, Swift, Dart-Flutter і Kotlin. На рисунку 2.1 зображена схема компонентів в Solana (також вказаний стек технологій, котрий може використовуватись) та механізми їх взаємодії [27].

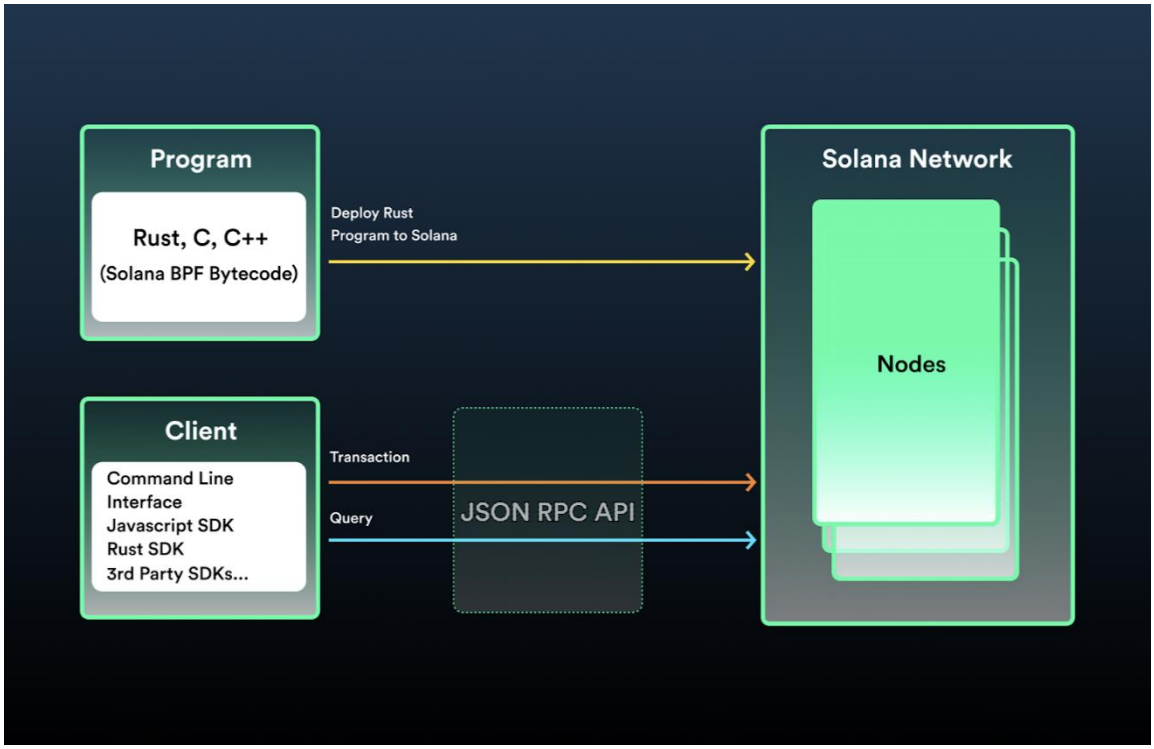

Рисунок 2.1 — Взаємодія контрактів та клієнтських додатків [27]

При виборі інструментів для написання on-chain частини варто враховувати наявність бібліотек для взаємодії із Solana (адже писати з нуля клієнта для JSON RPC API контрпродуктивно), а також на можливість написання зручного клієнтського інтерфейсу для репортерів. Найкращою мовою програмування для розробки dApps для взаємодії з смарт-контрактами на платформі Solana є TypeScript**,** адже Solana надає розширення Web3.js та Solana.js, які є офіційними бібліотеками для взаємодії з мережею Solana через JavaScript або TypeScript. Ці бібліотеки надають потужні інструменти для взаємодії з смарт-контрактами, включаючи підписання транзакцій, відправку запитів до мережі, отримання стану контрактів та багато іншого.

Відштовхуючись від мови, були обрані фреймворки, за допомогою котрих буде реалізовано користувацький інтерфейс.

Для швидкої та ефективної розробки інтерфейсу буде використовуватися React**,** оскільки він має:

– компонентний підхід: React працює на основі компонентів, що дозволяє розбити користувацький інтерфейс на невеликі, самодостатні частини;

– віртуальний DOM: React використовує віртуальний DOM, що дозволяє ефективно оновлювати тільки необхідні частини сторінки під час зміни стану. Це забезпечує швидку та ефективну роботу інтерфейсу, особливо при роботі зі складними та динамічними даними;

– розширюваність: React добре підходить для розробки великих та масштабованих проектів, оскільки має багату екосистему додаткових бібліотек та інструментів.

Для розробки UI частини буде використано Material-UI (MUI), котрий є популярною бібліотекою компонентів для розробки інтерфейсу користувача веб-додатків. MUI надає широкий набір готових компонентів, які можна використовувати для швидкої розробки інтерфейсу. Це включає кнопки, форми, таблиці, модальні вікна, картки, навігаційні елементи та багато іншого.

Також буде використовуватись Next.js є відкритим фреймворком розробки веб-додатків на базі React. Він надає потужні можливості для побудови сучасних інтерактивних додатків з високою продуктивністю. Переваги з Next.js:

– рендерінг на стороні сервера: HTML-код формується на сервері та відправляється клієнту. Це дозволяє отримати переваги щодо швидкості завантаження сторінок;

– статична генерація : HTML-файли для сторінок можуть бути попередньо згенеровані під час збірки додатка;

– клієнтський рендерінг: частину роботи виконує браузер. Це дозволяє створювати більш динамічні додатки, які здатні оновлювати інтерфейс без повного перезавантаження сторінки.

Висновки до розділу 2

В даному розділі були описані основні сутності, сценарії їх взаємодій та пов’язаність в системі. Також були прописані механіки контролю надання правдивих даних та створена система фільтрації репортерів, що надають фіктивну інформацію. Був проведений аналіз Blockchain систем та вибір стеку технологій. Для розробки on-chan частини буде використовуватись Rust із фреймворком Anchor, для off-chain частини буде використовуватись TypeScript із використанням фреймворків React, Material-UI та Next.js Надана інформація буде покладена в основу програмної реалізації системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

В цьому розділі буде описаний процес розробки та тестування двох частин системи.

3.1 Розробка on-chain частини

В цьому розділі буде спроектована та розроблений контракт на Solana та проведене дворівневе тестування.

3.1.1 Розробка акаунтів системи

В основу розробки on-chain частини було покладене проектування системи з минулого розділу. Вищезазначені сутності будуть представлені у вигляді десеріалізованих даних в акаунтах децентралізованої системи. За основу був обраний Blockchain Solana, котрий має свою специфічну модель акаунтів [25].

Смарт контракт, в котрому будуть описані вищезгадані сутності буде розгортатися на executable акаунт за допомогою системного акаунту System program. Скомпільований код контракту передається System Program (виконує системні операції та управляє базовими функціями мережі), котрий передає інструкції в системний акаунт BPF upgradable loader, котрий і буде створювати акаунт даних контракту, де і буде зберігатися байтовий код. Після створення акаунту контракту його фізичним власником і буде залишатися System Program, але логічним власником буде акаунт, котрий ініціалізував інструкцію розгортання контракту — іншими словами розгортання і модифікація контракта буде залишатися доступним для людини, котра розгортає контракт.

На рисунку 3.1 можна побачити три типи акаунтів – виконувані нативні програми, виконувані токен програми, а також невиконувані акаунти з даними або дата акаунти [25]. Дата акаунти — це саме той тип акаунтів, котрі будуть містити дані про вищеописані сутності.

Рисунок 3.1 — Типи акаунтів в Solana [25]

Варто зазначити, що в Solana немає view методів контракту, через унікальну систему збереження інформації — а саме не централізований збір усіх даних на одному дата акаунті, як в більшості EVM Blockchain х, зберігання даних проводиться в окремих дата акаунтах, що не прив'язані до однієї адреси. В випадку системи верифікації історичних подій буде використовуватися PDA акаунти — або ж program derived addresses.

Особливість цього типу акаунтів складається в тому, що вони не мають приватного ключа, усі їх транзакції підписуються контрактом, через котрий було створено PDA, варто знати, що вони не створюються технічно, а знаходяться серед ряду публічних ключів, що не мають коренів при передачі в хеш функцію.

PDA генеруються з комбінації seed фрази (наприклад, рядків бітових комбінацій, чисел) та ідентифікатора програми. Потім seed пропускається через хеш-функцію sha256, щоб побачити, чи генерує вона публічний ключ, який лежить на еліптичній кривій ed25519 чи ні. На рисунку 3.2 представлена графічна ілюстрація процесу знаходження PDA акаунту [26].

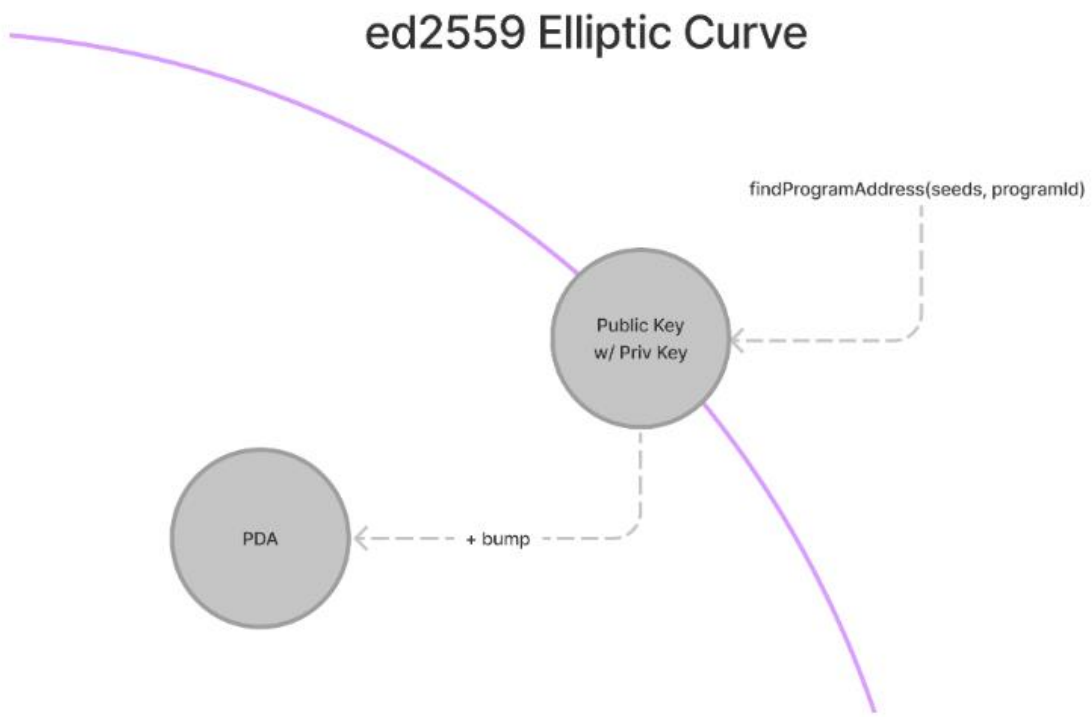

Рисунок 3.2 — Створення PDA [26]

Пропускаючи ідентифікатор програми та seed через хеш-функцію, існує ~50% ймовірність того, що можливо отримати дійсний відкритий ключ, який дійсно лежить на еліптичній кривій. У цьому випадку необхідно інкрементувати ідентифікатор, підтасувати вхідні дані, і запустити генерацію ще раз. Технічний термін для цього фактору підробки — "бамп". У Solana початковий bump = 255, якщо усі отримані рішення не призвели до отримання відкритого ключа — необхідно ітерувати бамп поки не отримаємо адресу, яка не лежить на еліптичній кривій [26].

Знайдені seed дають 2 основні можливості:

1) знаходження PDA акаунту серед інших акаунтів в мережі;

2) підписання транзакцій за допомогою заданих seed самим контрактом.

Такий підхід робить можливим загальнодоступність даних контракту (що і є метою даної роботи), а також робить більш дешевим зберігання даних на валідаторі. Враховуючи можливість децентралізованого зберігання даних, усі описані в попередньому розділі сутності будуть зберігатися окремо в кожному акаунті.

Відповідно до описаних сутностей були створені структури, котрі будуть десеріалізовані в дані PDA акаунтів. У зв'язку із тим, що кількість репортерів, подій, фактів і доказів не є константною, використання динамічно розширювальних контейнерів для зберігання цих типів (наприклад зберігання списку репортерів в асоціації) не буде оптимальним рішенням. Адже при ініціалізації акаунтів в Solana необхідно задати розмір пам'яті, котра буде виділена під дані акаунту, а також заплатити вказану ренту, котра вираховується відштовхуючись від заданого розміру пам'яті. З цього виходить, що пам'ять акаунту є статичною, тобто якщо з'явиться необхідність додавання нових даних, необхідно буде робити перерозподілення пам'яті акаунту, збільшуючи його на ту кількість байтів, яку необхідно ввести в дані акаунту, відповідно кожного разу треба буде витрачати lamports по-перше на оплату ренти за додаткову пам'ять, а по друге на реалокацію даних, що сама по собі є досить дорогою.

Тому було прийнято рішення використовувати модель Account Maps, яка використовується для організації та взаємодії з акаунтами без динамічного розширення даних на Solana. Account Maps дозволяють групувати облікові записи разом і визначати, як вони взаємодіють один з одним без зберігання їх в одному акаунті, замість цього цей метод з'єднує дані з окремих акаунтів мережі.

Мапа даної системи є абстракцією над логікою пов'язаності seed фраз акаунтів, за допомогою котрих можна знайти PDA в мережі. Основна ідея полягає в тому, щоб формувати зв'язок між акаунтами за рахунок прокидання в seed ID або ПК батьківської сутності, або ж пов'язаної сутності.

Дана система є досить ефективною, адже знаючи ID або ПК батьківської та ID дочірньої сутності, можна легко стягнути дані акаунту при цьому не витрачаючи обчислювальні потужності машини на пошук цього акаунту в

об’ємному масиві або векторі. Account Maps надає зручний спосіб організації даних і забезпечує ефективний доступ до акаунтів, що є важливим для швидкої та масштабованої обробки транзакцій у мережі Solana. Відштовхуючись від даних, наведених в діаграмі з кресленику ІАз91.010БАК.004 Д1 можна сказати, що:

1) Базовий клас асоціації має відношення до репортерів 1:N;

2) Клас репортера має відношення до події 1:N;

3) Клас репортера має відношення до факту 1:N;

4) Клас репортера має відношення до доказу 1:N;

5) Клас репортера має відношення до оцінки 1:N;

6) Подія має відношення до факту N:N;

7) Доказ до факту 1:N;

8) Оцінка до доказу і до зв’язку 1:N;

Під час створення акаунту, платником транзакції назначаю підписанта транзакцій, власником PDA — сам контракт, а розмір вираховую із пам’яті створеної структури + додаткові 32 байти у випадку, якщо необхідно буде розширити дані акаунту без додаткового реалокейту пам’яті.

Також варто зазначити, що кожен акаунт повинен мати унікальний ID, за допомогою котрого його можна було б знайти серед інших зв’язків. Виходячи з цього можна сказати, що в моделі Account Maps можна створити такий ланцюг послідовних зв’язків:

1) асоціація: незалежна сутність, seed містить унікальний ID (32 байтний масив — щоб була можливість закодувати назву) ;

2) репортер: залежить від асоціації та акаунту людини, котра і є репортером: seed містить ПК асоціації та ПК акаунту науковця;

3) подія: seed містить ПК асоціації та унікальний ID події (це має бути послідовний ID, тому варто зберігати лічильник подій в асоціації);

4) факт: seed містить ПК асоціації та унікальний ID події (теж послідовний ID, відштовхуючись від лічильника фактів в асоціації) ;

5) зв’язок: залежить від події та факту, seed містить ПК події та ПК факту, а також унікальний ID зв’язку (послідовний, залежить від лічильника зв’язків в події) ;

6) доказ: залежить від факту, seed містить ПК факту та послідовний ID доказу (залежить від лічильника доказів в факті);

7) оцінка доказу: залежить від підписанта (для того, щоб по-перше ідентифікувати репортера, а по-друге, щоб один репортер міг поставити тільки одну оцінку) та самого доказу, тож seed містить ПК репортера та ПК доказу;

8) оцінка зв’язку: залежить від підписанта (та ж логіка, що і в оцінці доказу) та самого доказу, тож seed містить ПК репортера та ПК доказу;

Дані про seed акаунтів будуть використані при ініціалізації та перевірках акаунтів при внесенні їх в контекст транзакції.

3.1.2 Створення контексту інструкцій

Формат зберігання даних визначений, але не менш важливим моментом є серіалізація цих даних. Однією із причин швидкої обробки транзакцій на контракті є препроцесінг акаунтів, котрі передаються в контекст транзакції. Розробник, котрий хоче виконати виклик до контракту має прописати усі акаунти, котрі будуть залучені в обробці транзакцій — системні в тому ж числі.

Контекст транзакції в Solana відноситься до інформації, яка доступна під час виконання транзакції у мережі, він містить важливі дані та функції, які допомагають взаємодіяти з мережею та забезпечують контекст для виконання операцій. Під час виконання система може перевіряти всі вхідні транзакції програми і перевіряти, чи перетинаються області пам'яті у аргументі транзакцій. Якщо ні, то середовище виконання може запустити ці транзакції паралельно, оскільки вони не конфліктують одна з одною.

За допомогою фреймворку Anchor при процесінгу транзакцій можна провести валідацію акаунтів, котрі в ньому знаходяться. Відповідно до цього можна прописати вимоги контекстів інструкцій.

Необхідно передавати в контекст такі акаунти:

– підписанта транзакції (той, хто створює акаунт);

– не ініціалізований акаунт (формується із seed та bump, що передається в аргументи інструкції);

– виконуваний акаунт контракту (тільки для створення асоціації);

– дата акаунт контракту, де зберігається байткод (тільки для створення асоціації);

– акаунт System Program (виконує системні операції та управляє базовими функціями мережі);

– додаткові PDA акаунти, що необхідні для перевірок логіки.

Необхідно проводити такі перевірки*:*

– перевірка seed та bump усіх PDA акаунтів — bump та необхідні аргументи для формування seed повинні зберігатися в акаунті (перевірка bump обов'язкова, акаунти з однаковими seed можуть мати різні bump, що дасть зловмисникам легко сфальсифікувати акаунт);

– перевірка підписанта — з акаунту контракту береться адреса дата акаунту програми, з котрого дістається upgrade authority, якщо він співпадає з акаунтом підписанта — перевірка пройшла (у випадку із створенням асоціації) ;

– при ініціалізації та модифікації репортера необхідно перевіряти, що підписант транзакції співпадає із ПК ключа, що створив асоціацію.

– при будь яких операціях, які виконує репортер необхідно перевірити, що цей репортер належить саме підписанту транзакції;

– при створенні події, факту та з'єднання необхідно перевірити, що репортер є Expert;

– при створенні доказу необхідно перевірити, що статус репортера не нижче Connosseur;

– оцінка доказів та зв'язків доступна для будь якого типу репортера;

– при виклику будь-якої інфомації від імені репортера необхідно перевіряти: що його статус активний він не заблокований і не дезактивований, та що кількість його штрафних очок не перевищує максимально допустиму;

– при оновленні події необхідно перевірити, що цю подію створив саме той репортер, котрий викликає інструкцію, а також що ця подія поки не має жодного з'єднання;

– при оновленні факту необхідно перевірити, що цей факт створив саме той репортер, котрий викликає інструкцію, а також що цей факт поки не має жодного доказу;

Логіка після обробки контексту:

– при ініціалізації асоціації створюються нульові лічильники подій та фактів. При створенні подій та фактів ці лічильники інкрементуються. Значення лічильника відповідає ID останнього створеного акаунту, адже вони послідовні;

– при ініціалізації події створюється нульовий лічильник зв'язків, логіка послідовності як і в попередньому пункті;

– при ініціалізації факту створюється лічильник доказів, послідовні ID;

– при ініціалізації репортера — лічильник штрафних очок;

– при ініціалізації доказу та зв'язку — лічильники згод та заперечень;

– при створенні оцінки (або доказу або зв'язку): інкрементуються лічильники згод та заперечень, якщо значення лічильника заперечень перевершує максимально допустиме значення — лічильник штрафних очок репортера інкрементується на 1.

3.1.3 Серіалізація даних

Формат зберігання даних визначений, але не менш важливим моментом є серіалізація цих даних. Однією із причин швидкої обробки транзакцій на контракті є препроцесінг акаунтів, котрі передаються в контекст транзакції. Розробник, котрий хоче виконати виклик до контракту має прописати усі акаунти, котрі будуть залучені в обробці транзакцій — системні в тому ж числі.

Одна транзакція в Solana може містити одразу декілька інструкцій (наприклад інструкція звернення до контракту і інструкція створення акаунту під PDA), кожна з цих інструкцій окрім метаданих має серіалізовані дані інструкції — назва інструкції, серіалізований список акаунтів з контексту та аргументи інструкції. Етапи серіалізації даних показані на рисунку 3.3 [28]. Серіалізація даних буде виконуватись за допомогою Borsh, що розшифровується як Binary Object Representation Serializer for Hashing (серіалізатор представлення двійкових об'єктів для хешування). Він призначений для використання у критично важливих для безпеки проектах, оскільки має пріоритети узгодженості, безпеки та швидкості і постачається з точною специфікацією.

Рисунок 3.3 — Серіалізація даних інструкції [28]

Серіалізовані дані відправляються на контракт, де десеріалізуються, проходять парсінг та необхідна інструкція виконується на валідаторі. Після цього отримані в результату виконання інструкції серіалізуються і запаковуються в новостворений акаунт. Серіалізація даних акаунту може відбуватися довільним серіалізатором, може використовуватися згаданий вище Borsh, або ж альтернативний серіалізатор Anchor, що має дуже зручний функціонал визначення типу структури акаунту. Це допомагає уникнути ручного розпакування та обробки бінарних даних та спрощує роботу з даними в контексті смарт-контрактів Solana.

Система ідентифікації акаунту полягає в формуванні певного числа — дискримінатора, котрий дає можливість визначити, яку конкретно структуру було серіалізовано (в нативній солані це треба робити вручну). Дискримінатор визначається першими 8 байтами хешу Sha256 Rust-ідентифікатора облікового запису — тобто назвою структурного типу — і гарантує, що жоден акаунт не може

бути замінений іншим. Це дозволяє Anchor дізнатися, до якого типу структури слід десеріалізувати дані. Це значно спростить обробку даних для користувача мережі, а також для розробника.

Тож серіалізація даних акаунту буде проведена десериализатором Anchor, котрий буде згенеровано за допомогою фреймворку Anchor індивідуально для кожної структури даних, а сериализація інструкцій буде проведена сериализатором Borsh.

3.1.4 Тестування контракту

Фреймворк Anchor надає зручні інструменти для тестування коду, але вони не підходять для тестування даного контракту, адже при запуску тестів Anchor запускає локальну ноду, завантажує контракт, але не створює дата акаунт контракту, він використовує Non Upgradable BPF Loader, котрий не дає можливості модифікувати код програми, тож при спробі витягнути дані акаунту контракту з’явиться помилка “Program data account is not initialized”. Для тестування коду дата акаунт програми необхідний, тому що він використовується при перевірці upgrade authority (це сутність в виконуваних акаунтах Solana, котра може модифікувати контракт) контракту при ініціалізації асоціації.

Щоб вирішити цю проблему перед тестами можна запускати локальний валідатор Solana, котрий використовує BPF Upgradable Loader, що створює дата акаунт контракту, компілювати код та розгортати його на локальну ноду зі заздалегідь згенерованими ключами (це необхідно, адже в контракті треба вказати ПК акаунту, на котрий розгортається контракт, якщо не вказати — ключ згенерується навмання і потім при виклику методів контракту з’явиться помилка несумісності дійсної адреси та тієї, що вказана в контракті), потім вже запускати тести Anchor, але необхідно задати опцію --skip-validator, котра гарантує, що внутрішній валідатор тестів не буде запущений, а самі тести будуть виконуватися на локальній машині. Тести будуть написані на мові TypeScript із використанням бібліотек Anchor.

Проектуючи тести, необхідно відштовхуватись від логіки, котра прописана у минулому розділі.

Тестування має включати такі юніт тести:

1) створення асоціації:

– невдалий: внесення невірного акаунту контракту;

– невдалий: внесення невідповідного дата акаунту контракту;

– невдалий: підписант транзакції не є upgrade authority контракту;

– успішний: коректно створена асоціація, відповідність усіх даних;

– невдалий: повторне створення акаунту неможливе.

2) створення репортера:

– невдалий: невірний акаунт асоціації;

– невдалий: підписант не є створювачем асоціації;

– успішний: коректно створений репортер, відповідність усіх даних;

– невдалий: повторне створення акаунту неможливе.

3) оновлення статусу репортера:

– невдалий: невірний акаунт асоціації;

– невдалий: підписант не є створювачем асоціації;

– невдалий: не вірний акаунт репортера;

– успішний: інструкція виконана, дані репортера змінено.

4) створення події:

– невдалий: невірний акаунт асоціації;

– невдалий: не вірний акаунт репортера;

– невдалий: підписант не є власником репортера;

– невдалий: тип репортера не Expert;

– невдалий: репортер має більше штрафних очок, ніж це допустимо;

– невдалий: заблокований репортер;

– невдалий: неактивний репортер;

– успішний: створення події, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

5) редагування події:

– 3 невдалі: невірні акаунти репортера та асоціації та події;

– невдалий: репортер не є створювачем події;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– невдалий: подія вже має зв’язки;

– успішний: інструкція виконана, дані події змінено.

6) створення факту:

– 2 невдалі: невірні акаунти репортера та асоціації;

– невдалий: тип репортера не Expert;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– успішний: створення факту, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

7) редагування факту:

– 3 невдалі: невірні акаунти репортера та асоціації та факту;

– невдалий: репортер не є створювачем факту;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– невдалий: факт вже має докази;

– успішний: інструкція виконана, дані факту змінено.

8) створення зв'язку:

– 4 невдалі: невірні акаунти репортера та асоціації, події та факту;

– невдалий: тип репортера не Expert;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– успішний: створення зв’язку, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

9) створення доказу:

– 3 невдалі: невірні акаунти репортера, асоціації та факту;

– невдалий: тип репортера не Expert та не Connesseur;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– успішний: створення доказу, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

10) створення оцінки доказу:

– 4 невдалі: невірні акаунти репортера, асоціації, факту та доказу;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– успішний: створення оцінки, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

11) створення оцінки з'єднання:

– 4 невдалі: невірні акаунти репортера, асоціації, факту та з'єднання;

– 3 невдалі: заблокований та неактивний статус, штрафний ліміт;

– успішний: створення оцінки, відповідність даних;

– невдалий: повторне створення акаунту неможливе.

Наступний етап тестування — інтеграційні тести. Інтеграційний тест має проходити за таким сценарієм: Створення асоціації => створення усіх видів репортерів => створення подій, фактів та зв’язків => створення доказу => створення оцінок (і заперечень і підтверджень) => перевірка penalty points => зміна статусів репортера => перевірка активності репортерів.

За описаними кейсами були прописані тести, деякі були скомпоновані разом. На рисунку 3.4 показаний результат виконання тестів.

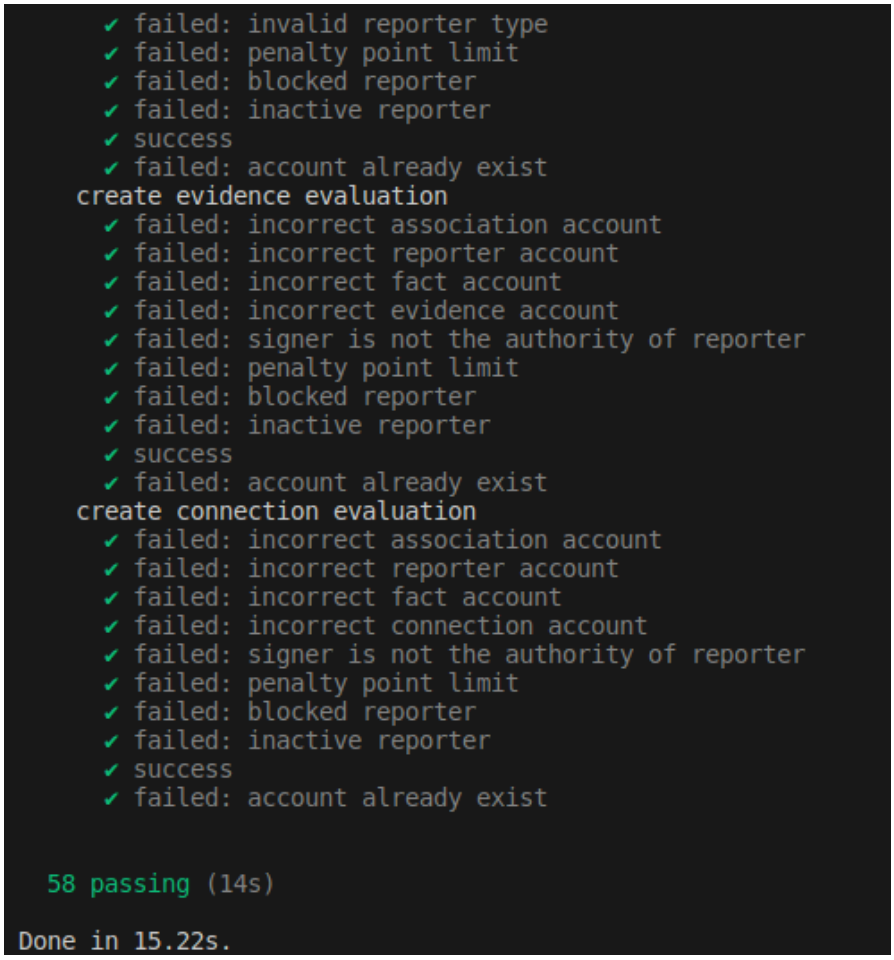

Рисунок 3.4 — Результати проходження тестів

3.2 Розробка off-chain частини

В даній частині буде спроектована та розроблена клієнтська частина застосунку та проведене мануальне тестування усієї системи.

3.2.1 Розробка бібліотеки для комунікації з контрактом

Для більш зручної роботи з Blockchain, було вирішено створити клієнтську бібліотеку для комунікації з контрактом. Основна ціль — інкапсуляція методів, що взаємодіють з Blockchain. При ініціалізації бібліотеки клієнт зможе виконувати основні дії, які описані в таблиці 3.1.

Таблиця 3.1 — Методи бібліотеки для взаємодії з контрактом

Назва методу	Функціонал	Результат
createConnection()	Створення з’єднання з Blockchain	Повернення SolanaConnection
connectProgram()	Підключення до контракту	HistoryValidator Program
findAssociationAddress(), findReporterAddress(), findEventAddress(), findFactAddress(), findEvidenceAddress(), findConnectionAddress()	Генерація ключів для PDA (для усіх сутностей)	[accountPubkey, bump]
getAccountData()	Стягнення усіх даних PDA акаунтів	Десеріалізовані дані акаунту
initAssociationAddress(), initReporterAddress(), initEventAddress(), initFactAddress(), initEvidenceAddress(), initConnectionAddress()	Запит до RPC API зі створенням транзакції для виклику метода ініціалізації заданого акаунту	Результат виконання транзакції — помилка або хеш транзакції
updateReporterStatus(), updateEvent(), updateFact()	Запит до RPC API зі створенням транзакції для виклику метода модифікації заданого акаунту	Результат виконання транзакції — помилка або хеш транзакції

3.2.2 Розробка контексту та валідація помилок

Для внутрішньої взаємодії між компонентами буде використаний Context — компонент фреймворку React, в котрому будуть задані основні компоненти системи. Його буде ініціалізовано один раз і при наступних візуальних генераціях буде використано вже створену версію контексту.

При запуску застосунку відбуватиметься ініціалізація контексту базовими параметрами, а під час подальшої роботи з клієнтом, значення будуть задаватися за допомогою setter методів. Для взаємоузгодженої роботи компонентів контекст повинен включати параметри, котрі описані в таблиці 3.2.

Таблиця 3.2 — Компоненти контексту

Назва та тип	Опис
program: HistoryValidatorProgram	З’єднання із контрактом
wallet: PhantomWallet	З’єднання із гаманцем Phantom
associationName: string	І’мя асоціації
userType: UserType	Тип користувача (Admin/Reporter/User)
reporter?: Reporter	Інформація по репортеру (якщо тип користувача — Reporter)

При початку роботи користувача з системою необхідно пройти 3 основні етапи ініціалізації, котрі будуть заповнювати контекст для подальшої роботи додатку:

1) підключення гаманця Phantom — це необхідна процедура, адже при відправці транзакцій на RPC ноду, необхідно поставити підпис, для цього необхідні ключі підписанта. Для цього підходить віртуальний гаманець Phantom, він має функціонал під’єднання до різних мереж, включаючи Solana;

2) під’єднання до асоціації: користувача буде надано вибір, підключитися до вже існуючої асоціації (для цього необхідно буде ввести ID асоціації), або ж створити нову (для цього необхідно буде заповнити дані для створення асоціації);

3) перевірка типу користувача: є три основні користувацькі типи — Admin, Reporter, User.

Якщо користувач у минулій частині створив асоціацію — йому автоматично буде назначений тип користувача Admin, якщо він обрав вже існуючу асоціацію буде проведена перевірка в декілька етапів:

– Перевірка Admin: з Blockchain стягуються та десериалізовані дані акаунту обраної асоціації, в котрому перевіряється поле authority (тобто ПК створювача), якщо воно спідпадає з ПК з гаманця Phantom — користувачу надається статус Admin;

– Якщо попередня перевірка не пройшла, проводиться перевірка на Reporter: з ПК користувача та ПК PDA акаунту користувача створюються seed та дістається PDA акаунт репортера, з Solana дістаються десереалізовані дані акаунту. Якщо такий акаунт існує та дані не пусті — відповідно репортер існує, тоді користувачу надається статус Reporter;

– В інакшому випадку користувачу надається статус User.

Після заповнення контексту, відповідно до типу користувача, буде наданий певний спектр дій, котрий описаний в минулому розділі. Кожному із користувачів буде надана можливість відправляти виклики до RPC, тож необхідно обробляти помилки, котрі можуть виникати в процесі роботи застосунку. Помилки системи можна розділити на декілька типів (представлені тільки ті типи, котрі можуть з’явитися в застосунку):

– код 0 – 19: системні помилки Solana;

– код 100 – 103: помилки при опрацюванні транзакцій;

– код 1000-1001: помилки опрацювання IDL;

– код 2000 – 2019: помилки при проходженні перевірок;

– код 3000 – 3014: помилки обробки акаунтів;

– код 6000+: авторські помилки, визначені в контракті.

Усі помилки будуть оброблюватись як developerError, окрім 2 останніх пунктів, де кожна з помилок буде індивідуально оброблятись на виводитись на інтерфейс користувачу. Для цього був створений мапінг помилок, котрі будуть показані користувача, в таблиці 3.3 приведений список можливих помилок, а на рисунку 3.5 показане відображення помилки.

Таблиця 3.3 — Помилки системи

Код	Помилка	Опис
5999	developerError	Внутрішня проблема розробки
6000	AuthorityMismatch	Невірний підписант
6001	AssociationMismatch	Невірно обрана асоціація
6002	InsufficientQualifications	Репортер має занизьку кваліфікацію
6003	BlockedReporter	Репортер заблокований
6004	InactiveReporter	Репортер не активний
6005	InvalidReporter	Невірно вказаний репортер
6006	LateUpdate	Час модифікації сплив
6007	NonSequentialId	Не послідовний ідентифікатор
6008	InvalidProgramData	Невірний дата акаунт контракту
6009	InvalidProgramAccount	Невірні дані програмного акаунту

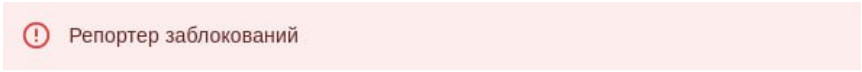

Рисунок 3.5 — Відображення помилки

На діаграмі з кресленику ІАз91.010БАК.004 Д4 послідовності розписаний процес ініціалізації та створення зв’язку між подією та факту. Можна побачити, що першими етапами є під’єднання до гаманця Phantom та вибір асоціації, усі дії користувача проходять через контекст та переадресовуються до бібліотеки, котра взаємодіє із контрактом, також всі процеси перевірки акаунтів звертаються до контракту — перевірка асоціації, репортера, факту та подію, вона полягає в

стягуванні даних акаунту та перевірки існування цього акаунту. Після того, як контракт повертає валідні дані акаунтів, виконується транзакція створення зв’язку, результат виконання транзакції валідується та повертається користувачу у вигляді вже обробленої помилки або ж успішного результату.

3.2.3 Розробка UI частини

Для розробки UI частини активно використовувалися бібліотека React та фреймворк MUI.

Було розроблено декілька екранів інтерфейсу:

– головний екран (відображення під’єднання до Phantom та вибір асоціації, відображення даних користувача);

– екран подій (відображення усіх подій + меню дій над подіями);

– екран фактів (відображення усіх фактів+ меню дій над фактами);

– екран репортерів (відображення усіх репортерів + меню дій над реопртерами).

Для розробки користувацького інтерфейсу були використані дефолтні компоненти MUI:

– List Item;

– Button;

– Icon Button;

– Thumb up/down item;

Також були розроблені власні компоненти:

– CreateReporterDialog (діалогове вікно для створення репортера);

– CreateEventDialog (діалогове вікно для створення подій);

– CreateFactDialog (діалогове вікно для створення факту);

– CreateEvidenceDialog (діалогове вікно для створення доказу);

– ExpandableListItem (компонент для відображення додаткової інформації);

– ItemMenu (компонент відображення додаткових опцій);

– ResponciveAppBar (компонент для відображення шапки застосунку);

– EvidenceListItem (похідний від ExpandableListItem, але з іншими параметрами);

Також був розроблений власний hook useDataContext для зручного отримання доступу до параметрів контексту. За допомогою нього можна було швидко звернутися з будь якого компонента застосунку до контексту в моменті виконання.

3.2.4 Тестування

Тестування off-chain частини буде проведене мануально. Для початку роботи необхідно виконати послідовно такі дії:

1) запустити локальний валідатор Solana;

2) задеплоїти контракт на валідатор;

3) створити ключі Solana та під'єднати їх до Phantom;

4) запустити проект командою npm run dev.

Сценарії тестування:

– створити асоціацію => створити репортера => модифікувати репортера => перевірити оновлені дані;

– створити асоціацію => спробувати створити подію/доказ/факт => отримати помилку "Невірний тип користувача";

– зайти в якості Connosseur/Validator спробувати створити подію/факт/ зв'язок => отримати помилку "Репортер має занизьку кваліфікацію";

– зайти в ролі Admin => створити репортера і деактивувати його => зайти у вже існуючу подію, із репортера з деактивованим статусом => виконати будь яку дію репортера => отримати помилку "Репортер не активний";

– зайти в ролі Admin => створити репортера і деактивувати його => зайти у вже існуючу подію, із заблокованим статусом => виконати будь яку дію репортера => отримати помилку "Репортер заблокований";

– зайти в ролі Admin => створити декілька репортерів => зайти з одного із цих акаунтів, створити факт та доказ факту => зайти з інших репортерів і поставити оцінку із запереченням доказу, доки перший репортер не отримає штрафне очко => зайти з репортера із штрафними очками (необхідно, щоб к-ть штрафних акаунтів перебільшувала або дорівнювала ліміту штрафних очок) => спробувати виконати будь-яку дію, що доступна репортеру => отримати помилку “Репортер заблокований”;

– зайти з репортера Expert => створити подію => створити докази події => спробувати модифікувати подію => отримати помилку “Час модифікації сплив” (аналогічно із фактом та доказами).

3.3 Демонстрація роботи інформаційної системи

В ході роботи над дипломним проектом було розроблено двокомпонентну інформаційну систему для верифікації історичних подій. На основі поставлених задач був спроектований користувацький інтерфейс з можливістю інтеграції гаманця Phantom. Код проекту можна знайти за посиланням або за QR-code з додатку А.

На рисунку 3.6 зображений початковий інтерфейс (етап під’єднання гаманця був пропущений), на котрому виведені параметри користувача, а також показані опції вибору або створення асоціації. Після створення під’єднання до асоціації з’являється основне вікго із відображенням даних користувача.

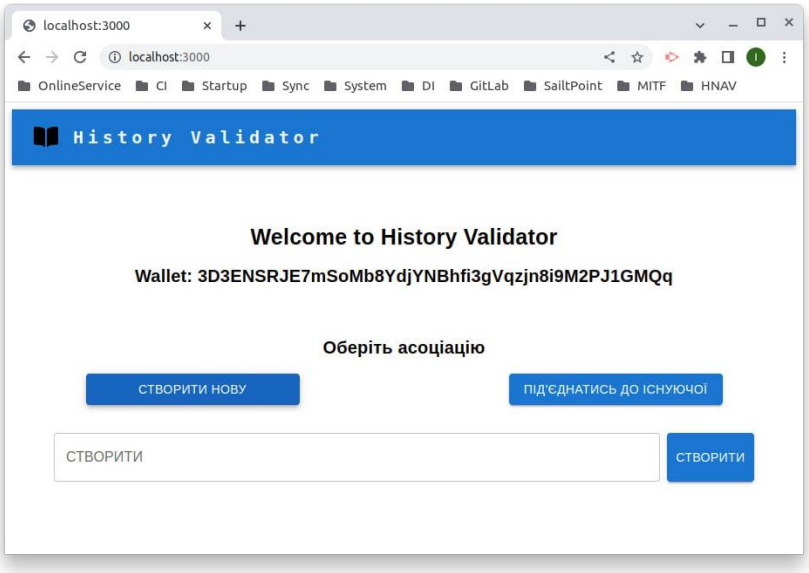

Рисунок 3.6 — Початковий інтерфейс

Якщо користувач зайшов у ролі адміна, у вкладці з репортерами буде доступна кнопка “Створити репортера”. На рисунку 3.7 зображене вікно створення репортера, на котрому можна побачити три поля для введення даних: ім'я репортера, адреса гаманця на Solana та тип репортера.

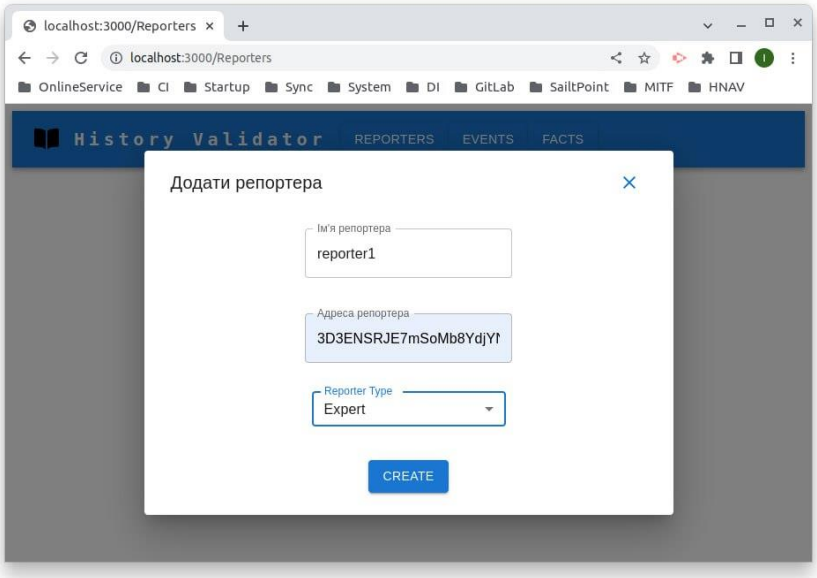

Рисунок 3.7 — Вікно створення репортера

На рисунку 3.8 зображена панель відображення усіх репортерів, на вкладці кожного репортера є меню редагування, в котрому можна змінити статус, при натисканні цієї кнопки відкривається панель редарування репортера.

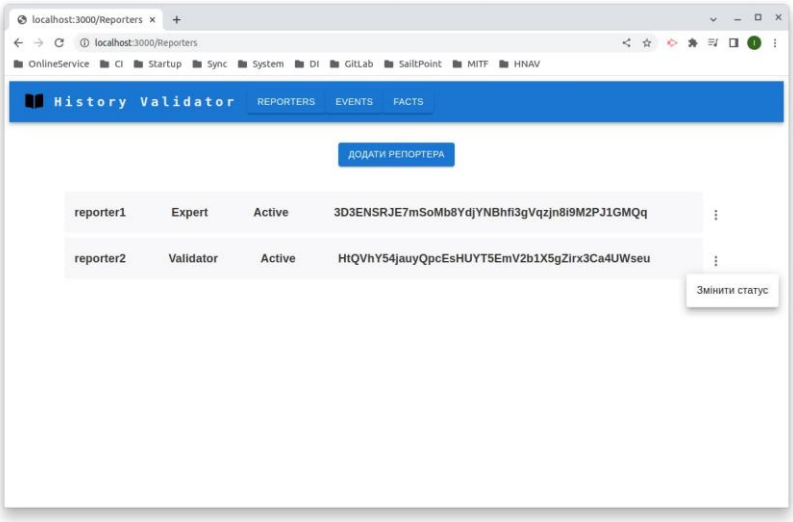

Рисунок 3.8 — Панель репортерів

Якщо користувач зайшов у ролі репортера типу Expert, у вкладці з подіями буде доступна кнопка “Створити подію”, аналогічно у вкладці із фактами буде кнопка “Створити факт”. На рисунку 3.9 зображене вікно створення події, котре містить п’ять полів внесення даних: назва, місце, опис, дати початку та кінця.

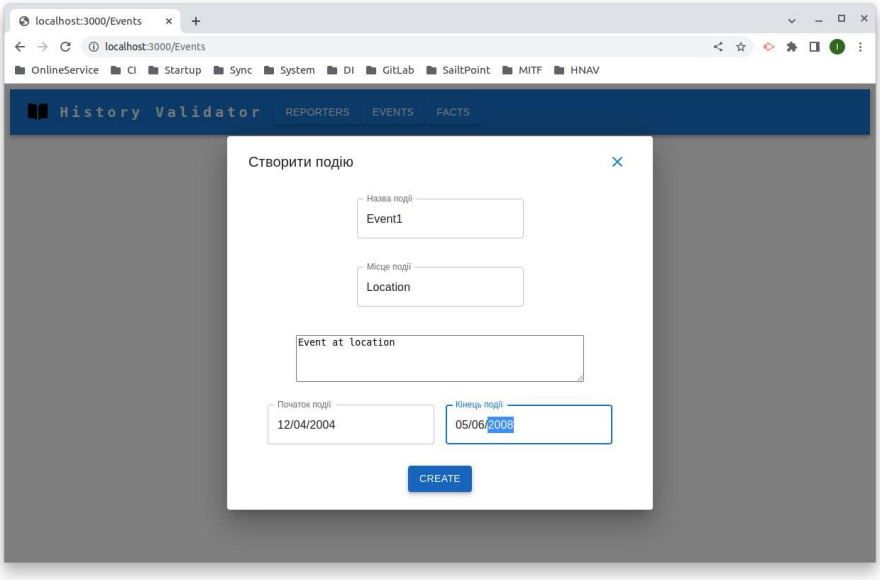

Рисунок 3.9 — Вікно створення події

На рисунку 3.10 зображена панель відображення усіх подій, в параметрах події можна оновити її, створити з'єднання, а також відобразити усі факти події. В

параметрах факту доступі опції редагування факту, створення доказу, а також демонстрація усіх доказів факту.

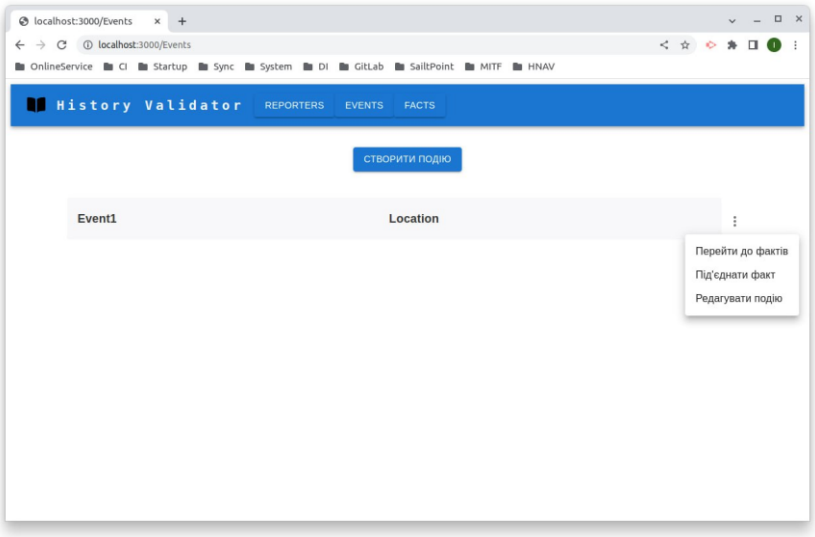

Рисунок 3.10 — Панель подій

Якщо користувач зайшов у ролі репортера типу Connosseur або Expert, йому буде доступна опція створення доказу, котра зображена на рисунку 3.11.

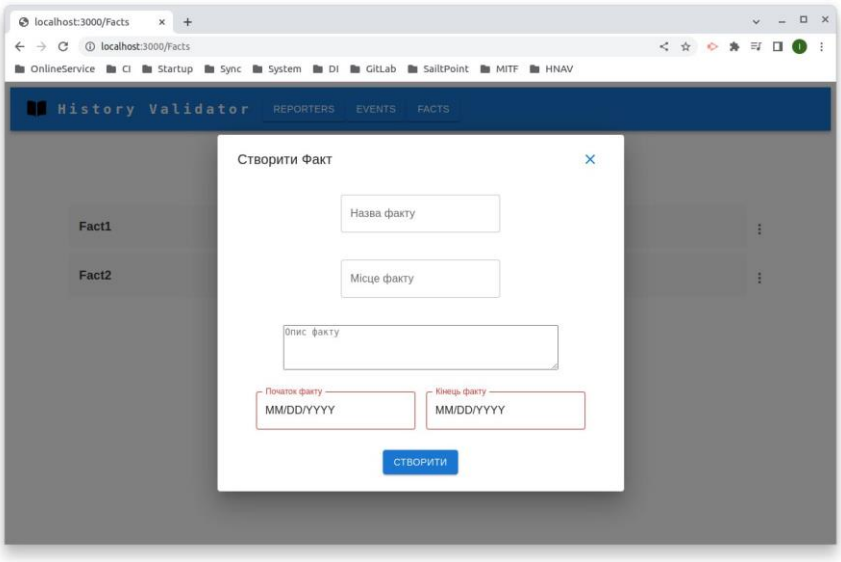

Рисунок 3.11 — Вікно створення факту

Усі типи репортерів можуть залишати оцінку доказам та з’єднанням – це відбувається натисканням кнопки лайк або дизлайк. На рисунку 3.12 зображена панель відображення усіх факів та їх доказів із оцінками.

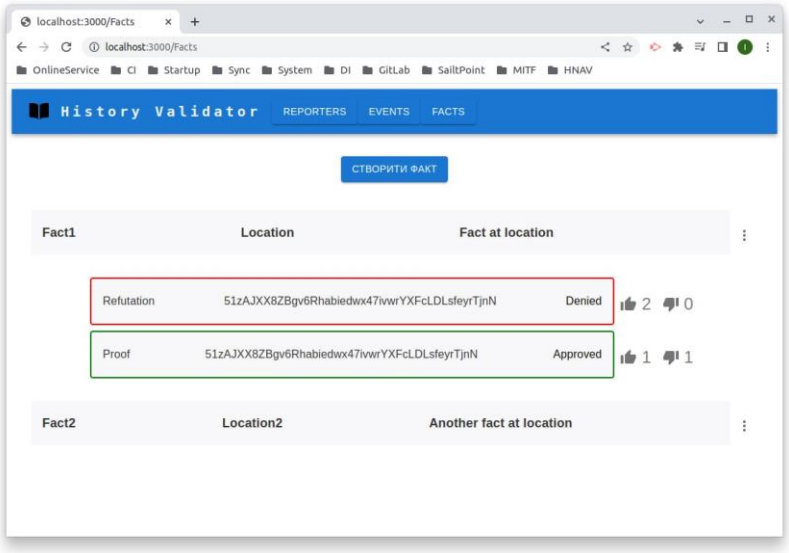

Рисунок 3.12 — Панель фактів із доказами та їх оцінками

На основі проведеної роботи був створений зручний користувацький інтерфейс для комунікації із контрактом. Дана робота може містити ряд покращень, але вже такому вигляді є досить перспективним та дієздатним продуктом.

4 ПРИКЛАДНІ АСПЕКТИ ЗАСТОСУВАННЯ СИСТЕМИ

4.1 Області застосування системи

Створена система верифікації історичних подій може бути використана для створення екосистеми верифікації наукових праць дослідників, вона може бути корисною для оцінки кваліфікації фахівців та створення інструменту презентації наукового авторитету вчених. Це може бути використано як в глобальних дослідницьких проектах, так і для локальних ініціатив, наприклад в едукаційних цілях для студентів шкіл та університетів для створення інтерактиву навчального процесу (будь який студент може і сам стати репортером в новоствореній асоціації).

Також система буде корисна в сфері просвітницької діяльності, як інформаційний архів. Це корисний інструмент, котрий може домогти більш якісно фільтрувати інформацію. За рахунок використання Blockchain частини, надана можливість відкритого доступу до інформації для будь якого користувача. На основі вже розробленої системи можна проектувати рішення, котрі будуть зручні та корисні для більш вузьких сфер, наприклад інтеграція AI з інформаційною системою верифікації історичних подій надасть можливість швиткого сортування та категоризації даних, а також пошуку найбільш правдивого переказу певного проміжку історії. Подібні інтеграції дадуть можливість екосистемі розширюватися та залучати нових користувачів.

4.2 Система заохочення репортерів

До вибору установ та організацій, що будуть мати статус репортера варто поставится з належною увагою та відповідальністю, покладаючись не тільки на їх авторитет в науковій спільноті, але й на рецензії інших науковців щодо них, на вклад в дослідження в сфері, в котрій вони працюють, а також, що дуже важливо, на їх неупередженість та незалежність від впливу третіх сторін.

Тому важливо мати інструменти, щоб зацікавити науковців приймати активну участь у розвитку екосистеми. Одним із рішень, з котрих репортери будуть отримувати вигоду, може стати токенізація активності в системі. Можна ввести систему заохочень у вигляді роздачі токенів системи за підтвердження правдивості наданих доказів.

Токеноміка даної системи може бути прописана таким чином: при внесенні даних до системи (події/факту/доказу) репортер буде стейкати певну суму токенів на токен акаунт асоціації. При отриманні певної кількості підтверджень правдивості він зможе заанстейкати цю суму назад на свій акаунт. Якщо внесена ним інформація буде набирати необхідну кількість оцінок правдивості, репортеру будуть нараховуватися токени системи. Такий підхід має 2 переваги: репортери будуть зацікавлені вносити тільки правдиві дані в систему, адже в інакшому випадку вони не зможуть повернути свій стейк назад, а також вони будуть зацікавлені отримати винагороду за підтвердження своєї інформації.

При початковому встановленні активності може бути важливим заливати отримані токени від стейкінга та токени системи в пули ліквідності для задання початкової ціни в доступні лаунчпади. Такий підхід допоможе стимулювати активність не тільки учасників системи, а і звичайних людей, котрі будуть зацікавлені в покупці токена.

ВИСНОВОК

В ході виконання дипломного проєкту було спроєктовано та розроблено інформаційну систему для архівації та валідації історичних подій.

На початку роботи був проведений аналіз предметної області та сформульована проблема, а саме відсутність єдиної архівної системи історичних подій із гарантією правдивості даних. Далі був проведений огляд вже існуючих рішень та аналіз їх переваг і недоліків, на основі отриманих даних були сформульовані вимоги до інформаційної системи.

Проведене проектування архітектури системи, прописані основні сутності на основі аналізу предметної області та сформована логіка взаємодії компонентів системи, були спроектовані основні механіки взаємодії учасників системи. Також наданий алгоритм взаємодії репортерів із системою, логіка валідації подій та алгоритми контролю правдивості інформації.

Був проведений огляд існуючих Blockchain систем, проаналізовані переваги та недоліки, на основі проведеного аналізу була обрана мережа Solana. Відштовхуючись від обраного Blockchain був проведений підбір списку технологій, котрі будуть використовуватися в розробці.

Розробка системи була розділена на дві частини — on-chain та off-chain частину. Відповідно до висновків минулих розділів контракти on-chain частини були написані на мові Rust із використанням фреймворку Anchor. Усі описані сутності в розділі із проектуванням були реалізовані у вигляді PDA акаунтів, була прописана логіка процесінгу акаунтів в контексті та внутрішня логіка контракту. Тестування контракту проводилось на мові TypeScript, для цього був розвернутий локальний валідатор, котрий був інтегрований в тести Anchor. Тестування контракту було дворівневе — юніт та інтеграційні тести.

Наступним етапом була розробка off-chain частини. Розробка йшла мовою TypeScript із використанням фреймворків React, Material-UI та Next.js. Спочатку була створена бібліотека для взаємодії із контрактом, потім був сформований контекст застосунку та прописаний мапінг помилок, далі йшла розробка

користувацького інтерфейсу. Дана частина роботи проходила мануальне тестування за декількома сценаріями на відповідність поставленим вимогам та функціям. Загалом система пройшла трирівневе тестування та показала високу швидкість обробки користувацьких команд та коректність даних.

В результаті виконання дипломного проєкту було розроблено інформаційну систему, котра зберігає історичні події в Blockchain, це гарантує їх незнищенність, а також має систему валідації фактів науковою спільнотою, тож користувач може легко оцінити об’єктивність представленої інформації. Ціль дипломного проєкту було досягнуто успішно, всі поставлені завдання були виконані в повному обсязі.

Розроблена система має велику значущість в теперішній час, адже в умовах профіциту інформації виникає проблема фільтрації інформації та аналізу її правдивості. Система має великий потенціал в різних галузях за рахунок наданої можливості інтеграції інших програмних рішень, що дозволить розширити екосистему та розвивати більш вузькі рішення на основі наявних даних інформаційної системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Evolution of Data to Life-Critical: Don't Focus on Big Data; Focus on the Data That's Big. URL: https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

2. Словник української мови: в 11 томах. — Том 6, 1975. — с. 749.

3. Словник української мови: в 11 томах. — Том 4, 1973. — с. 51.

4. Кулешов С. Документальні джерела наукової інформації: поняття, типологія, історія типологічної схеми / Укр. акад. інформатики. — К.: УкрІНТЕІ, 1995. — 191 с.;

5. Лаппо-Данилевский А. С. Методология истории. — 2. — 1913. —С. 186. — 321 с.

6. Truth and History: Historical Truth and Historical Narrative. URL: https://historynewsnetwork.org/article/174493.

7. Суперечка про геноцид вірмен у запитаннях і відповідях. URL: https://www.bbc.com/ukrainian/politics/2015/04/150423_armenia_massacre_ko

8. Арабо-ізраїльський конфлікт.

URL: https://vue.gov.ua/%D0%90%D1%80%D0%B0%D0%B1%D0%BE-%D1%96%D0%B7%D1%80%D0%B0%D1%97%D0%BB%D1%8C%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%84%D0%BB%D1%96%D0%BA%D1%82

9. Historical Truth.

URL: https://www.encyclopedia.com/psychology/dictionaries-thesauruses-pictures-and-press-releases/historical-truth

10. A guide to Historic Method" by Garraghan, Gilbert J. Garraghan, 1946. – с. 146.

11. Source criticism. URL: https://en.wikipedia.org/wiki/Source_criticism

12. Historical Reasoning: Towards a Framework for Analyzing Students' Reasoning about the Past — Educational Psychology Review. URL: https://link.springer.com/article/10.1007/s10648-007-9056-1

13. Guidelines for the Preparation, Evaluation, and Selection of History Textbooks (2018). URL: https://www.historians.org/jobs-and-professional-development/statements-standards-and-guidelines-of-the-discipline/guidelines-for-the-preparation-evaluation-and-selection-of-history-textbooks

14. Truth and History: Historical Truth and Historical Narrative. URL: https://historynewsnetwork.org/article/174493

15. Ways to verify online news in countries worldwide 2020. URL: https://www.statista.com/statistics/1228781/verify-news-online-worldwide/

16. Identifying false information online in countries worldwide 2020. URL: https://www.statista.com/statistics/1227193/identifying-misinformation-difficulty-worldwide/

17. Що таке Ethereum. URL: https://ethereum.org/uk/what-is-ethereum/

18. Що таке протокол NEAR. URL: https://www.kraken.com/uk-ua/learn/what-is-near-protocol

19. Solana . There are no bad questions about... blockchain basics. URL: https://solana.com/ru/learn/blockchain-basics

20. NEAR Explorer, валідатори. URL: https://explorer.near.org/nodes/validators

21. Solana Explorer, валідатори. URL: https://solanabeach.io/validators

22. BeaconScan Ethereum Explorer, валідатори. URL: https://beaconscan.com/validators

23. Біловезька угода. URL: — https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%BB%D0%BE%D0%B2%D0%B5%D0%B7%D1%8C%D0%BA%D0%B0_%D1%83%D0%B3%D0%BE%D0%B4%D0%B0

24. Historical Analysis and Interpretation. URL: https://phi.history.ucla.edu/nchs/historical-thinking-standards/3-historical-analysis-interpretation/

25. Solana account model. URL: https://www.alchemy.com/overviews/solana-account-model

26. PDA accounts. URL: https://solanacookbook.com/core-concepts/pdas.html#generating-pdas

27. Solana Network. URL: https://www.soldev.app/course/intro-to-reading-data

28. Серіалізація даних в контракті. URL: https://solanacookbook.com/guides/serialization.html#setting-up-for-borsh-serialization

29. Кулешов С. Документальні джерела наукової інформації як об'єкт дослідження інформатики: Студії з архів. справи та документознавства / УДНДІАСД. — 1996. — Т.1. — С. 72-76.п

ДОДАТОК А

Посилання на репозиторій: https://github.com/hlgltvnnk/history_validator

QR-code на репозиторій: