K S R INSTITUTE FOR ENGINEERING AND TECHNOLOGY

TIRUCHENGODE -637 215

Computer Science and Engineering

NAANMUDHALVAN

SB8024- Blockchain Development

by Naan Mudhalvan Scheme – 2023

TEAM ID: NM2023TMID11732

PROJECT DOMAIN: BLOCKCHAIN TECHNOLOGY

PROJECT TITLE: BLOCKCHAIN POWERED LIBRARY MANAGEMENT

TEAM MEMBERS

REGISTER NUMBER	**NAME**
731620104016	**GANESH KUMAR V**
731620104021	**IBRAHIM SHERIFF T.K**
731620104026	**KAMALESH T**
731620104027	**KAVINRAJ D**

1. INTRODUCTION

Project Overview

The Block chain-Powered Library Management System project aims to enhance library operations by leveraging block chain technology to improve efficiency, security, and transparency. It offers features such as block chain-based user registration and authentication, catalog management with an immutable ledger, streamlined borrowing and returning through smart contracts, transparent transaction records, secure interlibrary loans, automated fine management, reservation systems, user ratings and reviews, secure document delivery, robust data privacy, data-driven analytics, and decentralized governance. The system benefits from enhanced security, transparency, and a streamlined user experience, leading to operational efficiency, reliable record-keeping, and data-driven decision-making, ultimately revolutionizing library services for both administrators and users.

Purpose

The purpose of a block chain-powered library management system (LMS) is to make libraries more secure, transparent, and efficient, while also reducing costs. Block chain is a distributed ledger technology that allows for secure, transparent, and tamper-proof data storage. This makes it ideal for use in a variety of applications, including library management. A block chain-powered LMS could be used to store and manage all aspects of library data, including book records, user records, and circulation records. This data would be stored on the block chain in a secure and tamper-proof manner. Users could access library data through a web or mobile application. The application would authenticate users using block chain-based authentication. All data transactions on a block chain are recorded and publicly visible. This would ensure that all library activity is transparent and auditable. Block chain could help to streamline library operations by automating tasks such as book checkouts and returns.

2. EXISTINGPROBLEM

Existing Problem

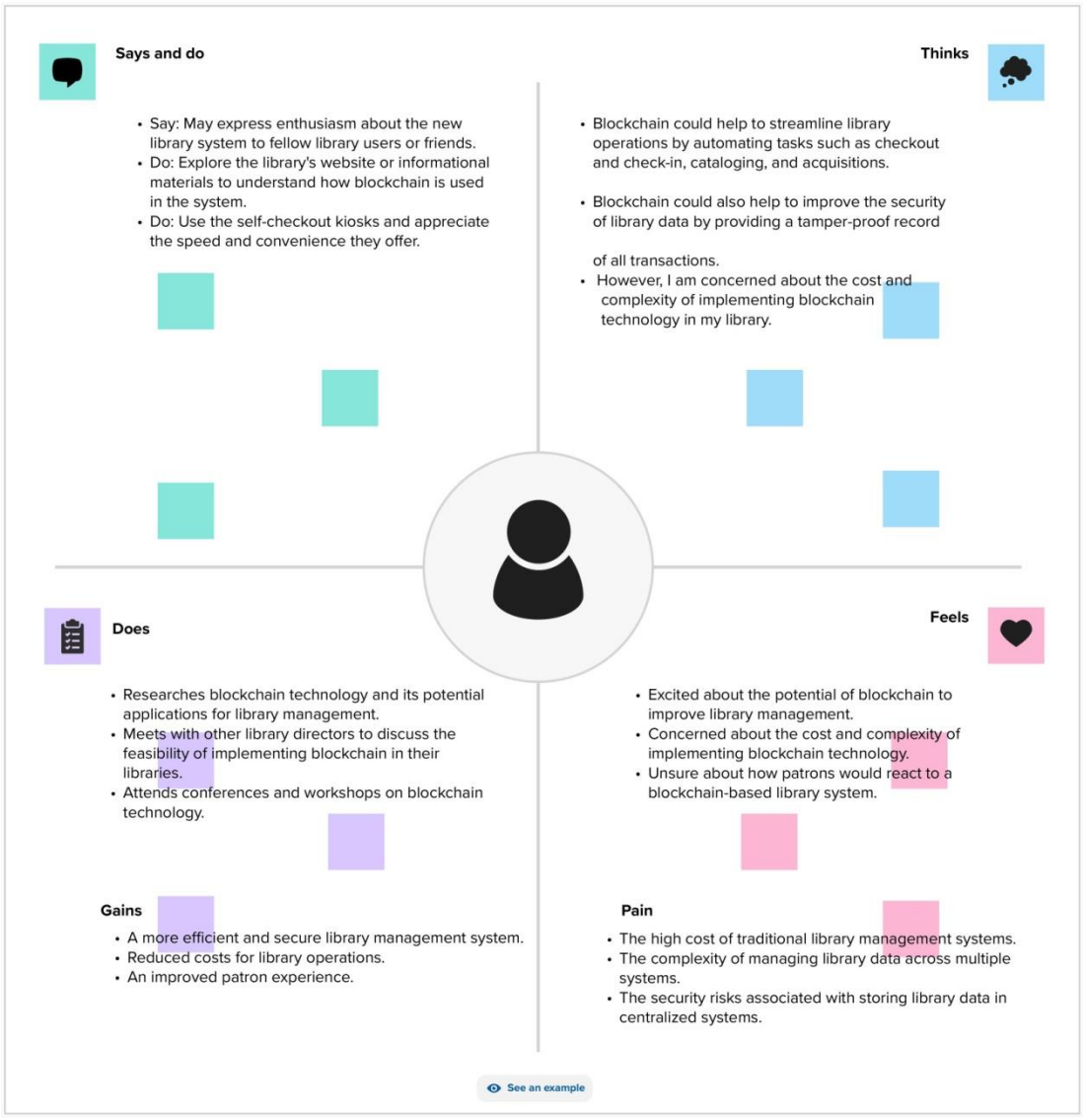
Library Management Systems face numerous challenges, including limited accessibility and the reliance on manual cataloging and updates, resulting in data inaccuracies and inefficiencies. Data security concerns arise from centralized data storage, inefficient resource tracking, and inadequate user engagement features. Manual check-out and return processes lead to delays and errors, while interlibrary connectivity issues hinder resource sharing. Operational complexities and inconsistent management of overdues and fines further compound the problems.Additionally, the lack of robust analytics and reporting capabilities limits data-driven decision-making. These issues underscore the need for more advanced Library Management Systems that can enhance accessibility, security, efficiency, and user satisfaction.

Problem Statement Definition

Libraries today face critical challenges in terms of efficiency, security, and transparency in their operations. Traditional Library Management Systems often struggleto provide real-time data updates, protect userdata fromsecurity breaches, and ensuretransparent and tamper-prooftransaction records. These limitations hinder the overall library experience and fail to meet the expectations of users in a digital age.The problem at hand involves the inefficiencies and security vulnerabilities inherent in conventional Library Management Systems. These systems often rely on manual processes for cataloging and updating data, resulting in inaccuracies and operational inefficiencies. Centralized data storage raises concerns about data security and privacy, leaving user information vulnerable to breaches. Furthermore, transaction records may lack transparency and immutability, impacting the integrity of library operations. As a result, there is a pressing need for a Blockchain-Powered Library Management System that can address theseissues by leveraging blockchain technology to enhance efficiency, security,and transparency in library operations, ultimately delivering a modern anduser-centric library experience.

3. IDEATION AND PROPOSED SOLUTION

EmpathyMapCanvas



IdeationandBrainstorming

1

define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm,

10 minutes

Design and implement a blockchain-based library management system to address the challenges faced by traditional library management systems, including data security, accessibility, and transparency, while also ensuring efficient and cost-effective operations for libraries of varying sizes and resources.Libraries play a vital role in knowledge dissemination, research, and education. However, traditional library management systems often face challenges such as data security, transparency, and efficient resource allocation. In the digital age, there is a growing need to leverage emerging technologies like blockchain to address these issues and create a more secure, transparent, and efficient library management system.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Kamalesh T

Decentralized Campaign | Smart contracts for book rentals | Transparent donation tracking

Ganesh Kumar

Immutable records of book ownership history | copyright and royalty management | Digital asset management

Kavinraj D

Community voting for book purchase | Supply chain management for physical books | Digital identity for users

Ibrahim sheriff T.K

Environmental impact tracking | Decentralized funding | Library event management

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Kamalesh T

Track library donation transparently using smart contracts | Manage copyright information and royalties blockchain

Ganesh Kumar

Trace the supply chain physical books to ensure authenticity | Implement blockchain for accessible library services

Ibrahim sheriff T.K

Develop a decentralized networks on blockchain for academics | Use blockchain to track and reduce the library's en

Kavinraj D

Supply chain management for physical books | Community voting for book purchase

3

GROUP IDEAS

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

15 minutes

SECURITY

ensures that resources are easily traceable, reducing the chances of loss and enabling quick inventory updates.

Blockchain can be configured to maintain the privacy of library patrons. Personal data can be anonymized or encrypted, ensuring that only authorized personnel have access to specific information.

Blockchain allows for fine-grained access control. Libraries can manage who can view, modify, or add data to the blockchain, enhancing security and privacy.

FEATURES

Enhanced library efficiency and transparency.

Improved patron experience and resource access.

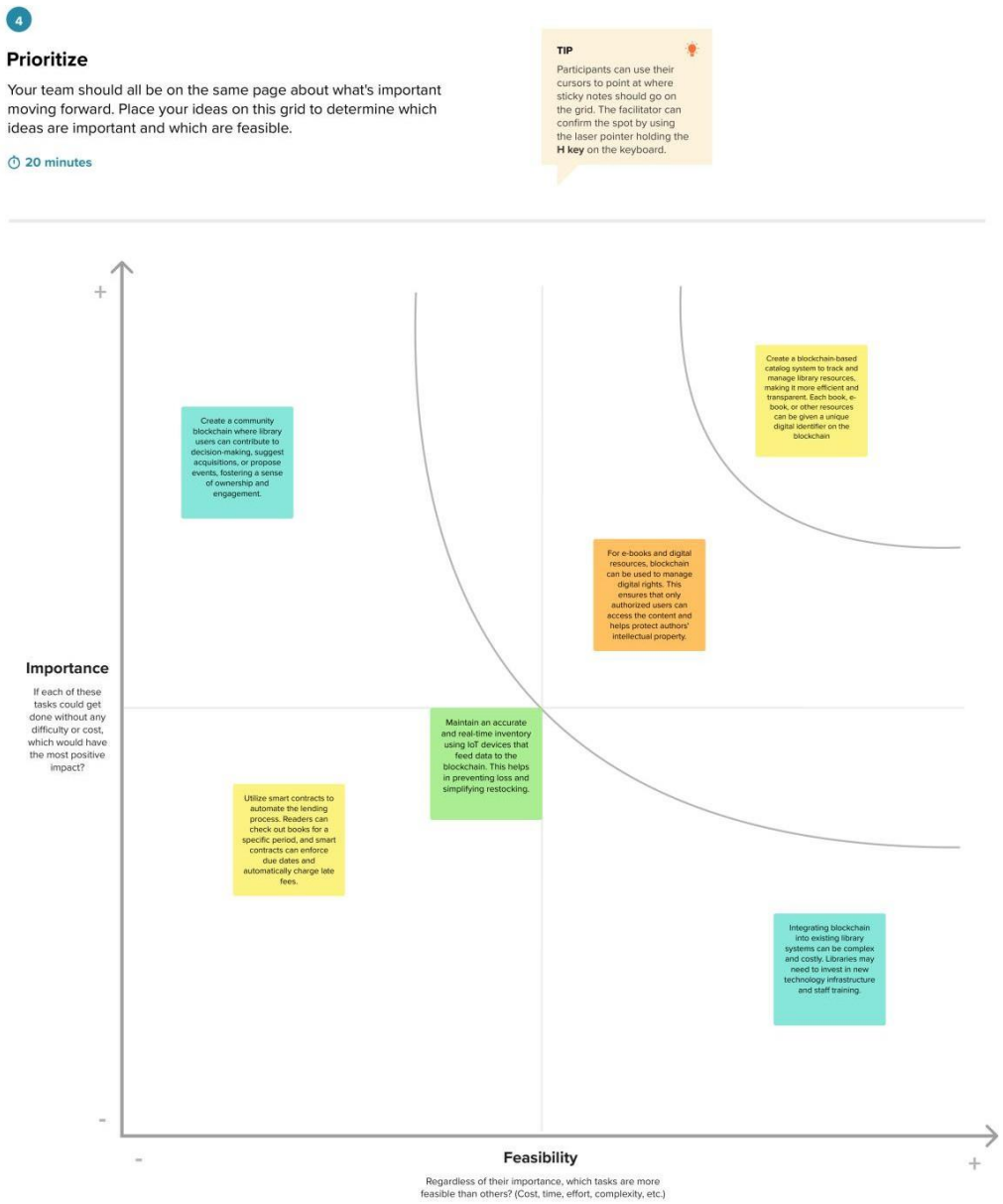
Minimized resource loss and overdue fines

CHALLENGES

Integrating blockchain into existing library systems can be complex and costly. Libraries may need to invest in new technology infrastructure and staff training.

Developing and maintaining a blockchain-based system can be expensive. This includes the cost of blockchain technology, smart contract development, and ongoing maintenance.

Developing a user-friendly interface for the platform



4 .REQUIREMENTANALYSIS

Functional Requirements

User Management:

i. User Registration: Users should be able to create block chain-based accounts with unique IDs and passwords.
ii. User Authentication: Implement secure authentication mechanisms using block chain keys.
iii. User Roles: Define different user roles, such as librarians, administrators, and patrons, with varying permissions.

Catalog Management:

i. ResourceUpload:Allow library staff upload, update, and manage the catalog of books, e-books, journals, and other resources.

ii. Immutability: Ensure that catalog data is stored on the blockchain, making ittamper-proof and auditable.
iii. SearchandDiscovery:Enableuserstosearchforresources,viewdetailed information, and check availability.

Borrowingand Returning:

i. Check-OutandReturn:Enableuserstocheckoutandreturnlibraryitems through the system.
ii. Smart Contracts: Implement smart contracts to enforce borrowingrules, duedates, and automated renewals

TransparentTransactionRecords:

i. Record Keeping: Record all library transactions on the blockchain, including borrowing, returning, and reservations.
ii. Real-timeUpdates:Ensurethattransactionrecordsareupdatedinreal-timefor users to view.

InterlibraryLoans:

i. InterlibraryLoanRequests:Facilitatesecureinterlibraryloansandresource sharing between different libraries.
ii. ResourceTracking:Allowlibrariestotrackthelocationandstatusof resources on loan.

FinesandOverdues:

i. Automatic Fine Calculation: Automatically calculate fines for overdue items using smart contracts.
ii. NotificationSystem:Notifyusersaboutfinesandduedatesthroughautomated alerts.

ReservationSystem:

i. ResourceReservations:Allowuserstoreservebooksandresources,with priority based on a transparent reservation queue.
ii. ReservationManagement:Enablelibrarystafftomanageandallocatereserved resources.

UserRatingsandReviews:

i. User Feedback: Enable users to rate and review library resources, contributing to a user-driven rating system.
ii. ReviewModeration:Implementamoderationsystemforuser-generated content.

DataPrivacy andAccessControl:

i. AccessControl:Implementrole-basedaccesscontroltosafeguarduserdata and comply with privacy regulations.
ii. UserDataProtection:Encryptandprotect userdata storedonthe blockchain.

AnalyticsandReporting:

i. Data Analytics: Generate reports and insights from blockchain data to inform decision-making, resource acquisition, and library management.
ii. CustomReports:Allow library staff to createcustom reports based on specific criteria.

.

These functional requirements are essential for the successful developmentand implementation of a Blockchain-Powered Library Management System, ensuring that it delivers improved efficiency, security, and transparency in library operations.

Non-FunctionalRequirements

Security:

i. Data Security: Ensure the highest level of data security by implementing robust encryption and access control mechanisms to protect user data and transaction records stored on the blockchain.
ii. Privacy Compliance: Adhere to data privacy regulations (e.g., GDPR) to safeguard user privacy and ensure compliance with relevant legalrequirements.
iii. BlockchainSecurity: Chooseasecureblockchainplatformandimplementbest practices to protect against hacking, fraud, and unauthorized access.

Performance:

i. Responsiveness: Ensure that the system is responsive and capable of handling concurrent user requests without significant delays or downtimes.
ii. Scalability:Designthesystemtohandleagrowinguserbaseandanexpanding library catalog efficiently by scaling resources as needed.
iii. TransactionSpeed:Optimizeblockchaintransactionprocessingforquickborrowin g, returning, and reservations.

Reliability:

i. SystemUptime:Maintainahighlevelofsystemavailabilitytominimize disruptions to library services.
ii. DataIntegrity:Guaranteetheintegrityofblockchain-storeddatatoprevent data corruption or loss.
iii. FaultTolerance:Implementmechanismstorecoverfromsystemfailuresor data discrepancies without affecting user experience.

Usability:

i. User-Friendly Interface: Design an intuitive and user-friendly interface that caters to both experienced and novice users, ensuring easy navigation and resource discovery.
ii. Accessibility: Ensure that the system is accessible to individuals with disabilities in compliance with accessibility standards (e.g., WCAG).

Interoperability:

i. Integration: Support integration with external systems, databases, and library networks to enable seamless resource sharing and interlibrary loans.
ii. Standard Compliance: Adhere to industry standards (e.g., MARC, Z39.50) for resource description and data exchange.

Compliance:

i. Regulatory Compliance: Ensure compliance with all relevant library and data protection regulations and standards.
ii. BlockchainGovernance:Implementmechanismsforadheringtoblockchain governance protocols and community guidelines.

PerformanceTesting:

i. LoadTesting:Performloadtestingtoensurethesystemcanhandlepeak usage without degradation in performance.
ii. StressTesting:Assessthesystem'sbehaviorunderextremeconditionsto identify potential weaknesses.

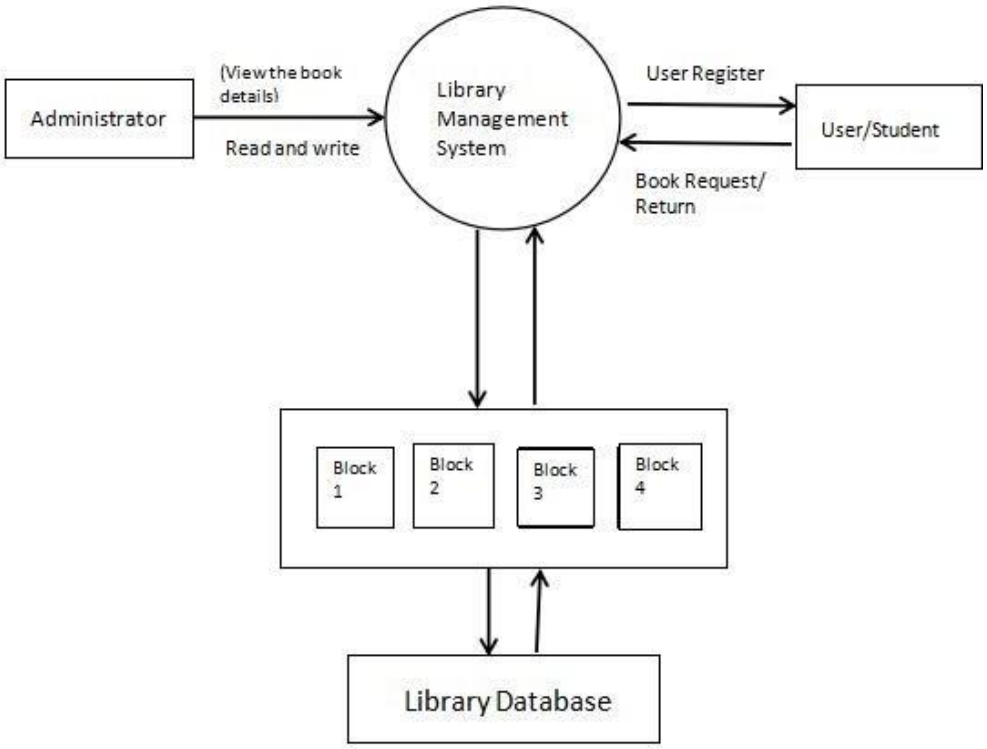
DataBackupandRecovery:

i. Data Backup:Implement regular,automateddata backupsto protectagainst data loss.
ii. DisasterRecovery:Establishadisasterrecoveryplantoquicklyrestorethe system in case of unexpected events.

These non-functional requirements are crucial for the successful deployment and operation of a Blockchain-Powered Library Management System, guaranteeing a secure, reliable, and user-friendly experience while adhering to legal and regulatory standards.

5. PROJECTDESIGN

DataFlowDiagram&User Stories



LIBRARYMANAGEMENTINBLOCKCHAIN

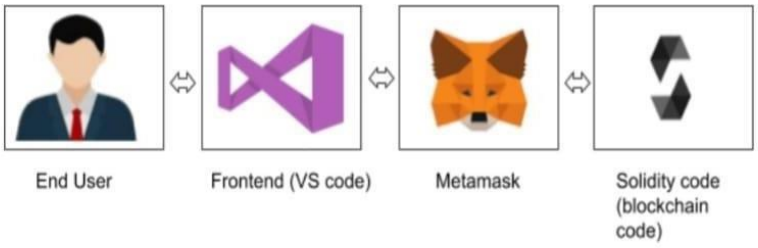
Story1

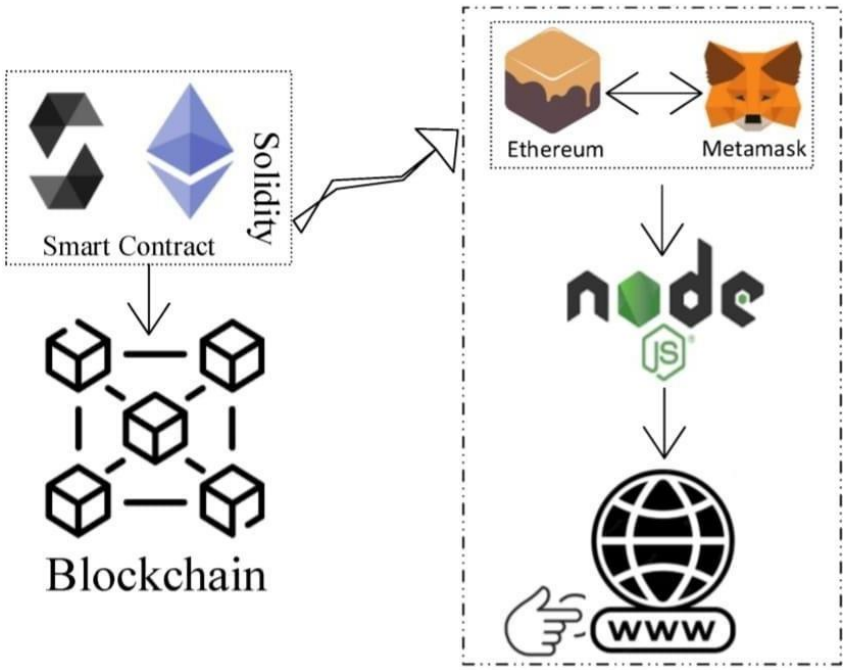
John, a university student, logs in using secure blockchain credentials and searches for a book. He initiates a borrowing request, and a smart contract on the blockchain confirms it. The system updates the catalog and notifies John of the due date. He retrieves the book from the library using location information, and upon return, another smart contract verifies it. Transaction data is recorded for analytics, allowing library staff to monitor usage. John's experience showcases the Blockchain-Powered Library Management System's efficiency and security, benefiting both users and administrators.

Story2

Sarah, a graduate student, logs into the library system with her secure blockchain credentials. She reserves a high-demand book, and the system records her reservation on the blockchain. The transparent reservation queue allows her to monitor her position. Sarah also provides feedback on her library experiences, securely stored on the blockchain. Notifications keep her updated on her reservation's status. When the book becomes available, she retrieves it. Library staff can analyze user feedback to improve services and resources, enhancing the overall library experience.

SolutionArchitecture

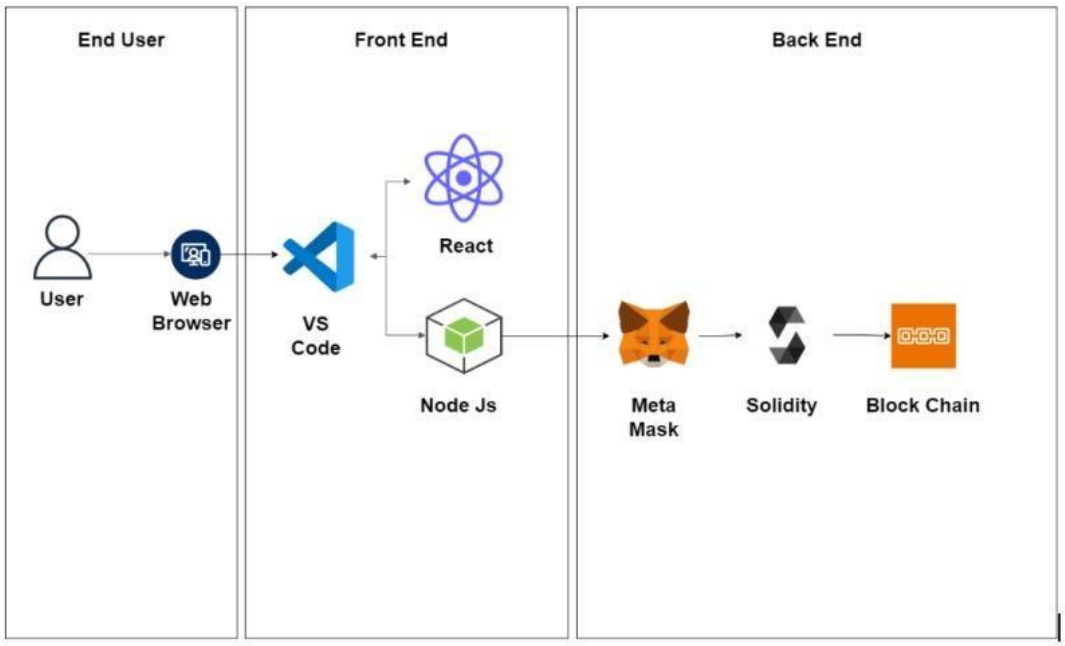




Interactionbetweenwebandthe Contract

6. PROJECTPLANNING&SCHEDULING

TechnicalArchitecture



GENERALARCHITECTURE

SprintPlanningandEstimation

Sprint planning involves selecting work items from the product backlog and committing to completing them during the upcoming sprint

Reviewing Product Backlog: Our project team, consisting of the product owner, scrum master, and development team, regularly reviews the items inthe product backlog.We evaluate user stories and technical tasks, taking into account the project's evolving needs and priorities.

Setting Sprint Goals: Based on the product backlog, our team establishes clear sprint goals. These goals guide the team's efforts during the sprint and ensure alignment with the broader project objectives.

Breaking Down User Stories: User stories and tasks are further decomposed into smaller, actionable sub-tasks. This detailed breakdown helps create a comprehensive plan for the sprint.

Estimating Work: Our development team employs agile estimation techniques,suchasstorypointsandt-shirtsizes,toestimatetheeffortrequired for each task. These estimates guide the team in understanding the scope and complexity of work for the sprint.

SprintBacklog:Theselected userstoriesand tasks, along withtheirestimates, constitute thesprint backlog. This forms the basis for what the team will work on during the sprint.

EstimationTechniques

Story Points: Story points serve as a relative measure of the complexity and effort needed to complete a task. Tasks are assigned story point values based on their complexity compared to reference tasks.

T-Shirt Sizes: To provide a quick and high-level estimate of effort, tasks are categorized into t-shirt sizes, such as small, medium, and large. This approach simplifies the estimation process for less complex tasks.

SprintDelivering Schedule

Week1:EstablishtheCore

• Setupthe basicblockchaininfrastructure.

• Implementaminimaluserregistration andauthentication system.

• Developarudimentary transactionrecording feature.

• Focusonfundamentalsecurity measures.

Week2:Expandand Enhance

• Extendtransactionrecordingtosupportmoretransactiontypes.

• Addbasicreal-timeupdates fortransactionstatus.

• Enhanceuserauthenticationwithmulti-factorauthentication.

• Begindevelopingauser dashboard.

Week3:Finalizeand Prepare

• Completetheuserdashboard withadditionalfeatures.

• Performminimalcompliancechecks.

• Conductbasictestingandissueresolution.

• Createessentialusersupportresources.

7. CODINGANDSOLUTIONING

Feature 1

Smartcontract(Solidity)

```
//SPDX-License-Identifier:MIT pragma
solidity ^0.8.0;

contractBookRegistry{
   addresspublicowner;

   constructor(){
      owner=msg.sender;
   }

   modifieronlyOwner(){
      require(msg.sender==owner, "Onlytheownercanperformthisaction");
      _;
   }

   struct Book {
      string title;
      stringauthor;
      addresscurrentOwner;
   }

   mapping(uint256=>Book)publicbooks;
   uint256 public bookCount;

   eventBookAdded(uint256indexedbookId,stringtitle,stringauthor,address indexed
owner);
   eventOwnershipTransferred(uint256indexedbookId,addressindexed
previousOwner, address indexed newOwner);

   functionaddBook(uint256registration,stringmemory _title,string memory
_author)externalonlyOwner{
```

```
        books[registration]=Book(_title,_author,owner);
        bookCount++;
        emitBookAdded(registration,_title,_author, owner);
    }

    functiontransferOwnership(uint256registrationId,address_newOwner)external{
        require(_newOwner != address(0), "Invalid address");
        require(_newOwner!=books[registrationId].currentOwner,"Thenewowneris the
same as the current owner");
        require(msg.sender==books[registrationId].currentOwner,"Onlythecurrent
owner can transfer ownership");

        addresspreviousOwner=books[registrationId].currentOwner;
        books[registrationId].currentOwner = _newOwner;

        emitOwnershipTransferred(registrationId,previousOwner,_newOwner);
    }

    functiongetBookDetails(uint256registrationId)externalviewreturns(string
memory, string memory, address) {

        Book memorybook=books[registrationId];
        return(book.title,book.author, book.currentOwner);
    }
}
```

The provided Solidity code defines a BookRegistry smart contract for managing a decentralized book registry. It allows the owner to add books, transfer ownership, and retrieve book details. Books are represented by a Book struct, and events are emitted for book additions and ownership transfers. The code ensures data immutability and transparency on the blockchain.

Feature2

```
importReact,{useState}from"react";
import{Button,Container,Row,Col}from'react-bootstrap'; import
'bootstrap/dist/css/bootstrap.min.css';
import{contract }from"./connector";

functionHome(){
  const [Id, setId] = useState("");
  const[Title,setTitle]=useState("");
  const [Author, setAuthor] = useState("");
  const [TranId, setTranId] = useState("");
  const [Owner, setOwner] = useState("");
  const [BookId, setBookId] = useState("");
  const[BookDet,setBookDet]=useState("");
  const [Wallet, setWallet] = useState("");




  consthandleId=(e)=>{
    setId(e.target.value)
  }

  consthandleTitle=(e)=>{
    setTitle(e.target.value)
  }

  consthandleAuthor=(e)=>{
    setAuthor(e.target.value)
  }

  consthandleAddBook=async()=>{ try {
      lettx=awaitcontract.addBook(Id.toString(),Title,Author) let
      wait = await tx.wait()
      alert(wait.transactionHash)
    }catch(error){
      alert(error)
    }
  }

  consthandleTid=(e)=>{
    setTranId(e.target.value)
  }

  consthandleNewOwner=(e)=>{
    setOwner(e.target.value)
  }
```

```
  consthandleTransfer=async()=>{ try
    {
      lettx=awaitcontract.transferOwnership(TranId.toString(),Owner) let
      wait = await tx.wait()
      console.log(wait);
      alert(wait.transactionHash)
    }catch(error){
      alert(error)
    }
  }

  consthandleBookId=(e)=>{
    setBookId(e.target.value)
  }

  consthandleBookDetails=async()=>{ try {
      lettx=awaitcontract.getBookDetails(BookId.toString()) let
      arr = []
      tx.map(e=>arr.push(e))

      setBookDet(arr)
    } catch (error) {
      alert(error)
      console.log(error);
    }
  }

  consthandleWallet=async()=>{ if
    (!window.ethereum) {
      returnalert('pleaseinstall metamask');
    }

    constaddr=awaitwindow.ethereum.request({ method:
      'eth_requestAccounts',
    });

    setWallet(addr[0])

  }

 return(
  <div>
   <h1style={{marginTop:"30px",marginBottom:"80px"}}>BookDetailsOn
Blockchain</h1>
      {!Wallet?

        <ButtononClick={handleWallet}style={{marginTop:"30px",marginBottom: "50px"
}}>Connect Wallet </Button>
        :
```

```
        <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom:
"50px", border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)} .. {Wallet.slice(-6)}</p>
      }
   <Container>
    <Row>
     <Colstyle={{marginRight:"100px"}}>
      <div>


       <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleId}type="number"placeholder="EnterRegistrationnumber"
value={Id} /><br />

      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleTitle}type="string"placeholder="EnterBookTitle"value={Title}
/><br/>

                <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleAuthor}type="string"placeholder="EnterBookAuthor" value={Author}
/><br />

       <ButtononClick={handleAddBook}style={{marginTop:"10px"}}
variant="primary">Add book</Button>
      </div>
     </Col>
      <Col>
        <div>
                <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleTid}type="number"placeholder="EnterRegistrationId" value={TranId}
/><br />

                <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleNewOwner}type="string"placeholder="EnterNewowner
metamask address" value={Owner} /><br />

          <ButtononClick={handleTransfer}style={{marginTop:"10px"}}
variant="primary">Transfer book ownership</Button>

        </div>
      </Col>
   </Row>
   <Rowstyle={{marginTop:"100px"}}>
     <Col>
       <divstyle={{margin:"auto"}}>
        <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleBookId}type="number"placeholder="EnterbookId"
value={BookId} /><br />

       <ButtononClick={handleBookDetails}style={{marginTop:"10px"}}
variant="primary">Get Identity Details</Button>
```

```
    {BookDet?BookDet?.map(e=>{
       return <p>{e.toString()}</p>
     }): <p></p>}
    </div>
   </Col>
  </Row>


  </Container>

 </div>
)
}
```

exportdefault Home;

ContractABI(ApplicationBinaryInterface):

TheabivariableholdstheABIofanEthereumsmartcontract.ABIsareessentialfor encoding and decoding function calls and data when interacting with the Ethereum blockchain.

MetaMask Check:

The code first checks whether the MetaMask wallet extensionis installedin the user's browser. If MetaMask is not detected, it displays an alert notifying the user that MetaMask is not found and provides a link to download it.

Ethers.jsConfiguration:

It imports the ethers library, which is a popular library for Ethereum development.It createsaproviderusingWeb3Provider,whichconnectstotheuser'sMetaMask wallet and provides access to Ethereum. It creates a signer to interact with The Ethereum blockchain onbehalfoftheuser.ItdefinesanEthereumcontractaddress and sets up the contractobject usingethers.Contract, allowingtheJavaScript codetointeract withthe contract's functions.In summary, this code is used for interacting with an Ethereum smart contract through MetaMask and ethers.js. It configures the necessary Ethereum provider and signer for communication with the blockchain and sets up a contract object for executing functions and fetching data from the specified contract address using the providedABI.

8. PERFORMANCETESTING

PerformanceMetrics

Smart Contract Libraries: Evaluate the performance of smart contract libraries available for your blockchain platform. Consider factors like ease of use, security, and efficiency when creating and deploying smart contracts.

Blockchain Protocol: Assess the performance of the underlying blockchain protocol, whether it's Ethereum, Hyperledger, or another platform. Consider factors like transaction speed, scalability, and consensus mechanism.

Security: Measure the security of the blockchain platform, including its resistance to attacks and vulnerabilities. Assess the availability of security libraries and practices for developing secure smart contracts.

Interoperability: Evaluate how well the blockchain platform integrates with other systems and networks. A strong library management system should support interoperability with various APIs and tools.

Documentation and Resources: Consider the availability and quality of documentation and educational resources for developers. Comprehensive documentation and tutorials can enhance performance.

Community and Support: A thriving community and support system are crucial for addressing issues and optimizing performance. Assess the availability of forums, developer communities, and support channels.

Tooling: Evaluate the performance of development tools and frameworks for blockchain. These tools can significantly impact the speed and efficiency of development.

Testing and Simulation: Implement testing and simulation libraries to measure and improve the performance of smart contracts and the blockchain network.

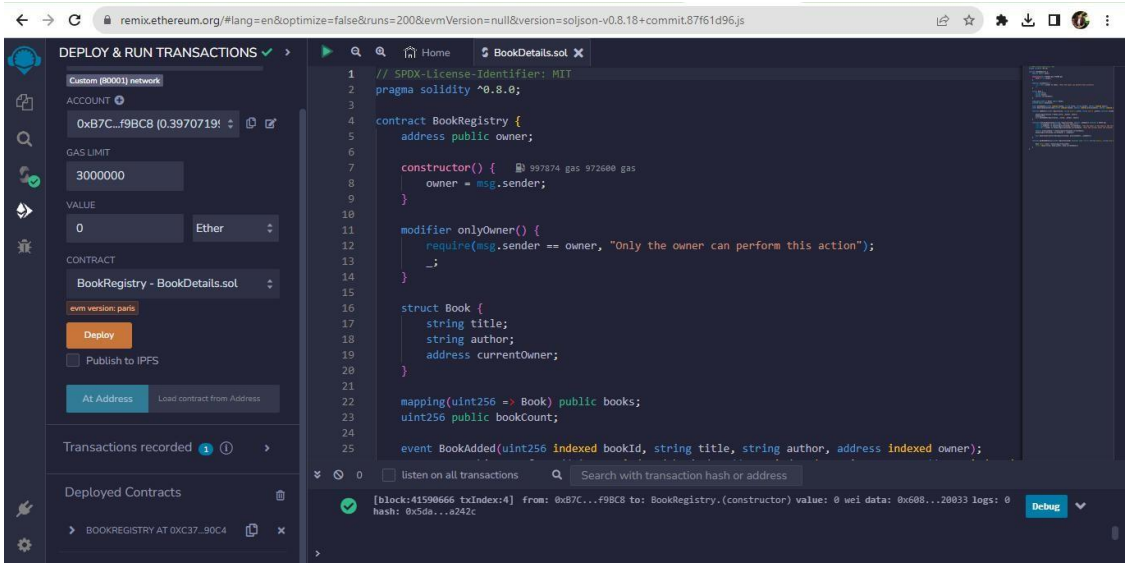
ScalabilitySolutions: Assesstheavailabilityoflibrariesandsolutionsfor scaling the blockchain network, as scalability is a critical performance factor. **Monitoring and Analytics:** Implement libraries for real-time monitoring and performance analytics to identify and resolve issues promptly.

RegulatoryCompliance: Considerlibrariesortoolsthathelpensurecompliance with legal and regulatory requirements in your jurisdiction.

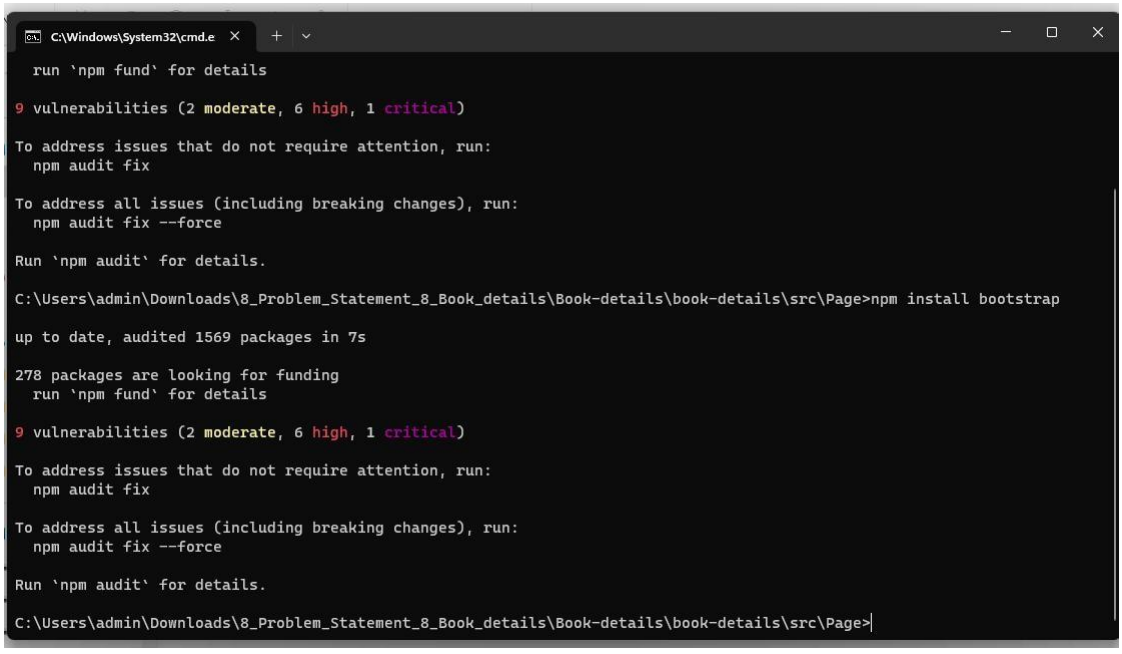
IntegrationwithDatabases: Evaluatelibrariesforintegratingblockchain with traditional databases to handle data efficiently

9. RESULTS

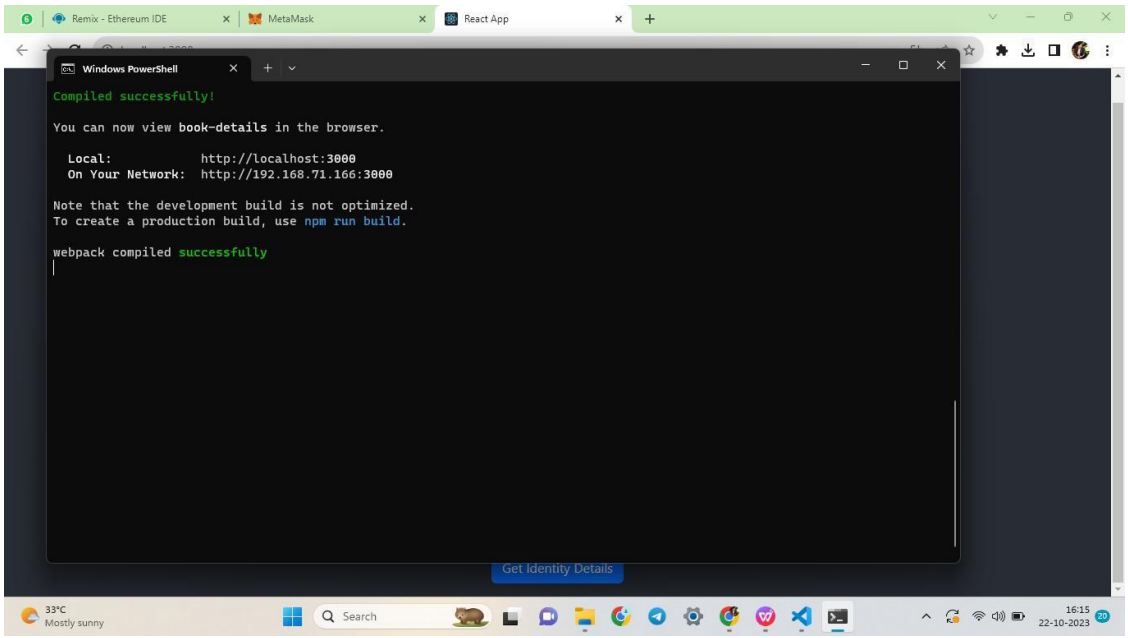
OutputScreenshots



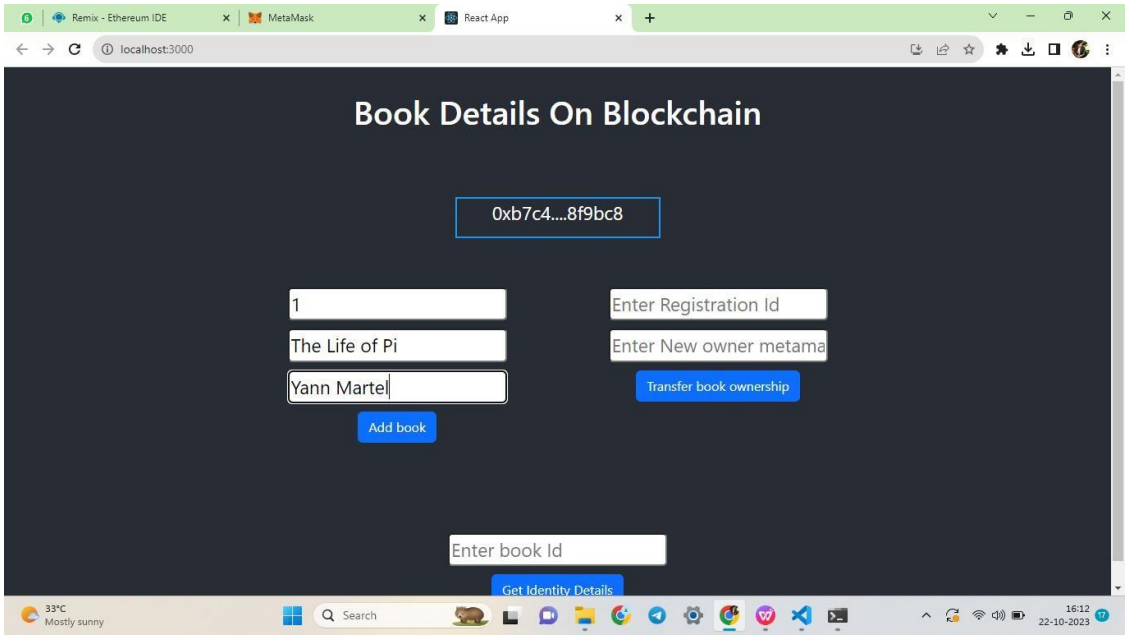
CREATINGASMARTCONTRACT



INSTALLINGDEPENDENCIES



HOSTINGTHESITE LOCALLY



OUTPUTSCREEN

10. ADVANTAGESANDDISADVANTAGES

Advantages

EnhancedSecurity: Blockchain'sdecentralizedandimmutableledgerensures that library records are secure and tamper-proof, reducing the risk of data breaches and fraud.

Digital Identity and Access Control: Blockchain can be used for managing user identities and access to library resources. It allows for more robust authentication and authorization mechanisms.

Smart Contracts for Transactions: Smart contracts can automate and streamline processes like borrowing and returning books, fines, and reservations, reducing administrative overhead.

Interlibrary Loans: Blockchain can facilitate interlibrary loans by providinga transparent and secure platform for tracking borrowed materials across multiple libraries.

SupplyChainforLibraryMaterials: Librariescouldуseblockchaintotrack the procurement and distribution of materials, ensuring a streamlined supply chain.

Disadvantages

Scalability:Blockchain'sscalabilitylimitationsmaymakeitlesssuitablefor managing large and complex code libraries.

Complexity:Implementingblockchainforlibrarymanagementcanbetechnically complex, requiring expertise in blockchain development.

Cost: Runningablockchainnetworkcanbecostlyintermsofbothdevelopment and ongoing maintenance.

Speed: Blockchain transactions can be slower compared to traditionaldatabasesystems,whichmightnotbeidealforrapiddevelopmentandupda tes. **Adoption:** Blockchain technology is still not widespread, and finding developers with expertise in blockchain development can be challenging.

11. CONCLUSION

Blockchain technology holds immense potential for revolutionizing library management in multiple ways. Its decentralized and immutable ledger ensures the security and integrity of library records, minimizing the risk of data breaches andfraud.Blockchain'sroleinmanagingdigitalidentitiesandaccess

control offers robust authentication and authorization mechanisms for users. The automation capabilities of smart contracts simplify processes like book borrowing, returns, fines, and reservations, reducing administrative burdens. Interlibrary loans benefit from blockchain's transparency and security,enabling efficient tracking of borrowed materials across various libraries. Additionally, blockchain can be instrumental in managing copyright and licensingagreements,guaranteeingfaircompensationforcontentcreators.The technology also brings transparency to donation tracking, helping libraries monitor the use of funds. For rare and historical materials, blockchain aids in establishing provenance and authenticity. Furthermore, it streamlines the supply chain for library materials, enhancing overall efficiency. However, successful implementation requires thoughtful planning to address the unique needs and challenges of library operations.

In conclusion, implementing blockchain in library management can improve security, transparency, and efficiency. However, it's important to carefullyplan and design the system to meet the specific needs and challenges of the library environment.

12. FUTURESCOPE

Data Security: Blockchain can ensure the security of sensitive library data, such as patron records and digital resources. In the future, as cybersecurity threats evolve, blockchain can play a crucial role in protecting this data.

Digital Rights Management (DRM): With the increasing prevalence of digital books and media in libraries, blockchain can be used to manage digital rights and access control. It can ensure that only authorized users have access to specific digital content.

Interlibrary Loans: Blockchain can streamline interlibrary loans, making it easier for libraries to share resources securely and efficiently. Future developments may include the use of smart contracts to automate the borrowing and lending process.

Provenance and Authentication: Blockchain can be used to verify the authenticity and provenance of rare and historical items in a library's collection. This can be crucial for preserving cultural heritage.

Decentralized Cataloging: Blockchain can facilitate decentralized cataloging systems where libraries worldwide can collaboratively maintain a single, authoritative catalog. This could make it easier for users to find resources.

Digital Preservation: Blockchain can play a role in ensuring the long-term preservation of digital assets in libraries. This includes ensuring the integrityof digital collections for future generations.

Smart Contracts for Acquisitions: Libraries could use smart contracts to automate the acquisition of new materials, ensuring that copyright and licensing agreements are adhered to automatically.

Public Record Transparency: In government and public libraries, Blockchain can enhance transparency and accountability in record-keeping, ensuring that records are tamper-proof and publicly accessible.

Tokenization and Microtransactions: In the future, libraries may explore tokenization to enable microtransactions for accessing content or services, allowing users to pay for what they use with digital tokens.

UserPrivacy: As concernsabout user privacy continuetogrow, libraries may useblockchain togivepatronsmorecontrolover theirpersonal data, including what information is shared for library services

To implement a blockchain project in library management effectively, it's crucial to consider the specific needs of the library, its user base, and the evolving technological landscape. Additionally, collaborating with blockchain experts and staying updated on blockchain developments is vital for a successful project.

12. APPENDIX

Source code

BookDetails.Sol(SmartContract)

```
//SPDX-License-Identifier:MIT pragma
solidity ^0.8.0;

contractBookRegistry{
   addresspublicowner;

   constructor(){
      owner=msg.sender;
```

```
    }

    modifieronlyOwner(){
        require(msg.sender==owner, "Onlytheownercanperformthisaction");
        _;
    }

    struct Book {
        string title;
        stringauthor;
        addresscurrentOwner;
    }

    mapping(uint256=>Book)publicbooks;
    uint256 public bookCount;

    eventBookAdded(uint256indexedbookId,stringtitle,stringauthor,address indexed
owner);
    eventOwnershipTransferred(uint256indexedbookId,addressindexed
previousOwner, address indexed newOwner);

    functionaddBook(uint256registration,stringmemory _title,string memory
_author)externalonlyOwner{

        books[registration]=Book(_title,_author,owner);
        bookCount++;
        emitBookAdded(registration,_title,_author, owner);
    }

    functiontransferOwnership(uint256registrationId,address_newOwner)external{
        require(_newOwner != address(0), "Invalid address");
        require(_newOwner!=books[registrationId].currentOwner,"Thenewowneris the
same as the current owner");
```

```
        require(msg.sender==books[registrationId].currentOwner,"Onlythecurrent
owner can transfer ownership");

        addresspreviousOwner=books[registrationId].currentOwner;
        books[registrationId].currentOwner = _newOwner;

        emitOwnershipTransferred(registrationId,previousOwner,_newOwner);
    }

    functiongetBookDetails(uint256registrationId)externalviewreturns(string
memory, string memory, address) {

        Book memorybook=books[registrationId];
        return(book.title,book.author, book.currentOwner);
    }
}
```

Connector.js

```
const{ethers}=require("ethers");

constabi=[
 {
  "inputs":[],
  "stateMutability":"nonpayable",
  "type": "constructor"
 },
 {
  "anonymous":false,
  "inputs": [
   {
    "indexed": true,
    "internalType":"uint256",
    "name": "bookId",
    "type":"uint256"
   },
```

```
 {
  "indexed": false,
  "internalType":"string",
  "name": "title",
  "type":"string"
 },
 {
  "indexed": false,
  "internalType":"string",
  "name": "author",
  "type":"string"
 },
 {
  "indexed": true,
  "internalType":"address",
  "name": "owner",
  "type":"address"
 }
],
"name":"BookAdded",
"type":"event"
},
{
"anonymous":false,
"inputs": [
 {
  "indexed": true,
  "internalType":"uint256",
  "name": "bookId",
  "type":"uint256"
 },
 {
  "indexed": true,
  "internalType":"address",
```

```
  "name":"previousOwner",
  "type": "address"
 },
 {
  "indexed": true,
  "internalType":"address",
  "name": "newOwner",
  "type":"address"
 }
],
"name":"OwnershipTransferred",
"type": "event"
},
{
"inputs":[
 {
  "internalType":"uint256",
  "name": "registration",
  "type": "uint256"
 },
 {
  "internalType":"string",
  "name": "_title",
  "type":"string"
 },
 {
  "internalType":"string",
  "name": "_author",
  "type":"string"
 }
],
"name":"addBook",
"outputs":[],
"stateMutability":"nonpayable",
```

```
 "type":"function"
},
{
 "inputs":[],
 "name":"bookCount",
 "outputs": [
  {
   "internalType":"uint256",
   "name": "",
   "type":"uint256"
  }
 ],
 "stateMutability":"view",
 "type": "function"
},
{
 "inputs":[
  {
   "internalType":"uint256",
   "name": "",
   "type":"uint256"
  }
 ],
 "name":"books",
 "outputs": [
  {
   "internalType":"string",
   "name": "title",
   "type":"string"
  },
  {
   "internalType":"string",
   "name": "author",
   "type":"string"
```

```
  },
  {
   "internalType":"address",
   "name": "currentOwner",
   "type": "address"
  }
 ],
 "stateMutability":"view",
 "type": "function"
},
{
 "inputs":[
  {
   "internalType":"uint256",
   "name": "registrationId",
   "type": "uint256"
  }
 ],
 "name":"getBookDetails",
 "outputs": [
  {
   "internalType":"string",
   "name": "",
   "type":"string"
  },
  {
   "internalType":"string",
   "name": "",
   "type":"string"
  },
  {
   "internalType":"address",
   "name": "",
   "type":"address"
```

```
 }
],
"stateMutability":"view",
"type": "function"
},
{
"inputs":[],
"name":"owner",
"outputs": [
 {
  "internalType":"address",
  "name": "",
  "type":"address"
 }
],
"stateMutability":"view",
"type": "function"
},
{
"inputs":[
 {
  "internalType":"uint256",
  "name": "registrationId",
  "type": "uint256"
 },
 {
  "internalType":"address",
  "name": "_newOwner",
  "type":"address"
 }
],
"name":"transferOwnership",
"outputs": [],
"stateMutability":"nonpayable",
```

```
  "type":"function"
 }
]

if (!window.ethereum) {
 alert('MetaMaskNotFound')
 window.open("https://metamask.io/download/")
}

exportconstprovider=newethers.providers.Web3Provider(window.ethereum); export
const signer = provider.getSigner();
exportconstaddress="0xd9145CCE52D386f254917e481eB44e9943F39138" export
const contract = new ethers.Contract(address, abi, signer)
```

Home.js

```
importReact,{useState}from"react";
import{Button,Container,Row,Col}from'react-bootstrap'; import
'bootstrap/dist/css/bootstrap.min.css';
import{contract }from"./connector";

functionHome(){
   const [Id, setId] = useState("");
   const[Title,setTitle]=useState("");
   const [Author, setAuthor] = useState("");
   const [TranId, setTranId] = useState("");
   const [Owner, setOwner] = useState("");
   const [BookId, setBookId] = useState("");
   const[BookDet,setBookDet]=useState("");
   const [Wallet, setWallet] = useState("");




   consthandleId=(e)=>{
```

```
  setId(e.target.value)
}

consthandleTitle=(e)=>{
  setTitle(e.target.value)
}

consthandleAuthor=(e)=>{
  setAuthor(e.target.value)
}

consthandleAddBook=async()=>{ try {
    lettx=awaitcontract.addBook(Id.toString(),Title,Author) let
    wait = await tx.wait()
    alert(wait.transactionHash)
  }catch(error){
    alert(error)
  }
}

consthandleTid=(e)=>{
  setTranId(e.target.value)
}

consthandleNewOwner=(e)=>{
  setOwner(e.target.value)
}

consthandleTransfer=async()=>{ try
  {
    lettx=awaitcontract.transferOwnership(TranId.toString(),Owner) let
    wait = await tx.wait()
    console.log(wait);
```

```
    alert(wait.transactionHash)
  }catch(error){
    alert(error)
  }
}

consthandleBookId=(e)=>{
  setBookId(e.target.value)
}

consthandleBookDetails=async()=>{ try {
    lettx=awaitcontract.getBookDetails(BookId.toString()) let
    arr = []
    tx.map(e=>arr.push(e))

    setBookDet(arr)
  } catch (error) {
    alert(error)
    console.log(error);
  }
}

consthandleWallet=async()=>{ if
  (!window.ethereum) {
    returnalert('pleaseinstall metamask');
  }

  constaddr=awaitwindow.ethereum.request({ method:
    'eth_requestAccounts',
  });

  setWallet(addr[0])
```

```
  }

 return(
  <div>
   <h1style={{marginTop:"30px",marginBottom:"80px"}}>BookDetailsOn
Blockchain</h1>
      {!Wallet?

        <ButtononClick={handleWallet}style={{marginTop:"30px",marginBottom: "50px"
}}>Connect Wallet </Button>
        :
        <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom:
"50px", border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)} .. {Wallet.slice(-6)}</p>
      }
   <Container>
    <Row>
     <Colstyle={{marginRight:"100px"}}>
      <div>


      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleId}type="number"placeholder="EnterRegistrationnumber"
value={Id} /><br />

     <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleTitle}type="string"placeholder="EnterBookTitle"value={Title}
/><br/>

                <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleAuthor}type="string"placeholder="EnterBookAuthor" value={Author}
/><br />

      <ButtononClick={handleAddBook}style={{marginTop:"10px"}}
variant="primary">Add book</Button>
```

```
      </div>
     </Col>
      <Col>
        <div>
              <inputstyle={{marginTop:"10px",borderRadius:"5px"}}
onChange={handleTid}type="number"placeholder="EnterRegistrationId" value={TranId}
/><br />

              <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleNewOwner}type="string"placeholder="EnterNewowner
metamask address" value={Owner} /><br />

          <ButtononClick={handleTransfer}style={{marginTop:"10px"}}
variant="primary">Transfer book ownership</Button>

        </div>
      </Col>
    </Row>
    <Rowstyle={{marginTop:"100px"}}>
      <Col>
       <divstyle={{margin:"auto"}}>
        <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleBookId}type="number"placeholder="EnterbookId"
value={BookId} /><br />

       <ButtononClick={handleBookDetails}style={{marginTop:"10px"}}
variant="primary">Get Identity Details</Button>
      {BookDet?BookDet?.map(e=>{
         return <p>{e.toString()}</p>
       }): <p></p>}
      </div>
     </Col>
    </Row>
```

```
  </Container>

 </div>
)
}

exportdefault Home;
```

App.js

```
import'./App.css';
importHome from'./Page/Home'

functionApp(){
 return (
   <divclassName="App">
     <headerclassName="App-header">
       <Home/>
     </header>
   </div>
 );
}

exportdefault App;
```

App.css

```
.App{
 text-align:center;
}

.App-logo {
 height:40vmin;
 pointer-events:none;
}

@media(prefers-reduced-motion:no-preference){
```

```
  .App-logo{
    animation:App-logo-spin infinite20slinear;
  }
}

.App-header{
  background-color:#282c34;
  min-height: 100vh;
  display:flex;
  flex-direction:column;
  align-items: center;
  justify-content: center;
  font-size:calc(10px+2vmin); color:
  white;
}

.App-link {
  color:#61dafb;
}

@keyframesApp-logo-spin{
  from {
    transform:rotate(0deg);
  }
  to{
    transform:rotate(360deg);
  }
}
```

Index.js

```
importReact from'react';
importReactDOMfrom'react-dom/client';
import './index.css';
importAppfrom'./App';
```

```
importreportWebVitalsfrom'./reportWebVitals';

constroot=ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
);

//Ifyou wanttostartmeasuring performanceinyourapp, passa function
//tologresults(for example: reportWebVitals(console.log))
//orsendtoananalyticsendpoint.Learnmore:https://bit.ly/CRA-vitals
reportWebVitals();
```

Index.css

```
body {
margin: 0;
  font-family:-apple-system,BlinkMacSystemFont,'SegoeUI','Roboto','Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing:antialiased;
  -moz-osx-font-smoothing:grayscale;
}

code {
  font-family:source-code-pro,Menlo,Monaco,Consolas,'CourierNew', monospace;
}
```

GIT HUB LINK - https://github.com/Sistance255/NM--BLOCKCHAIN-POWERED-LIBRARY-MANAGEMENT.git

PROJECT DEMO LINK - https://drive.google.com/file/d/1IgCd2vYQ4Lk9ty_QvprZ4_f9_vSu8sDT/view?usp=sharing