**ABSTRACT**

**ABSTRACT**

In the digital era, ensuring the integrity and authenticity of electronic documents is imperative for secure transactions and data exchange. Traditional digital signature systems, reliant on centralized authorities, present vulnerabilities to manipulation and fraud. This project introduces a decentralized solution leveraging blockchain technology to address these challenges. The Blockchain Digital Signature System facilitates secure document signing and verification by employing the Ethereum blockchain platform and the Elliptic Curve Digital Signature Algorithm (ECDSA). The system allows users to generate, sign, verify, and timestamp digital documents, ensuring transparency, immutability, and enhanced security. Key features include secure key generation, signature creation, verification, and timestamping. Security measures include key management, data encryption, smart contract audits, and access controls. Performance evaluation assesses transaction throughput, latency, and gas costs to optimize system efficiency. Use cases span across finance, healthcare, and supply chain industries. Future enhancements include integration with identity management systems and interoperability with other blockchain platforms. The Blockchain Digital Signature System offers a decentralized, transparent, and secure solution for digital document signing, paving the way for trustable digital transactions in various domains.

**INTRODUCTION**

**CHAPTER 1**

**1.INTRODUCTION**

The Blockchain Digital Signature System stands at the forefront of innovation in digital document management, offering a paradigm shift in how we secure and authenticate electronic transactions. At its core, this system harnesses the power of blockchain technology, a distributed ledger that records transactions across a network of computers in a secure and transparent manner. Specifically, the system utilizes the Ethereum blockchain platform, renowned for its smart contract functionality, which enables the execution of self-executing contracts with predefined conditions.

One of the key features of this system is its utilization of the Elliptic Curve Digital Signature Algorithm (ECDSA), a widely adopted cryptographic algorithm known for its efficiency and security. With ECDSA, users can generate public-private key pairs, where the private key is used to sign documents, and the corresponding public key is used to verify the authenticity of the signature. This cryptographic process ensures that digital signatures are secure, tamper-proof, and verifiable by anyone with access to the public key.

Through the integration of smart contracts on the Ethereum blockchain, the system provides a decentralized platform for users to interact with digital documents securely. Smart contracts serve as self-executing agreements with predefined rules and conditions, allowing for automated execution of transactions without the need for intermediaries. In the context of the Blockchain Digital Signature System, smart contracts facilitate the generation, verification, and timestamping of digital signatures, all while ensuring transparency and immutability of the underlying data.

**EXISTING SYSTEM**

**2.****EXISTING SYSTEM**

The existing system for digital signatures operates within a centralized framework, typically relying on certificate authorities or trusted third parties for key management and signature validation. Users generate key pairs, with the private key used for signing documents and the corresponding public key for verification. However, this centralized model presents vulnerabilities, including the risk of a single point of failure if the authority is compromised. Additionally, transparency is limited as users must trust the authority's security measures. Moreover, the system is susceptible to various attacks such as key theft or impersonation, undermining the integrity of digital signatures. In contrast, the Blockchain Digital Signature System proposes a decentralized approach utilizing blockchain technology. By recording signatures on a distributed ledger, this system eliminates reliance on centralized authorities, ensuring transparency, immutability, and heightened security. Through smart contracts on the Ethereum blockchain, users can securely generate, sign, verify, and timestamp digital documents, mitigating the risks associated with the existing centralized model**.**

**2.1 DISADVANTAGES OF EXISTING SYSTEM**

Centralization: The reliance on centralized certificate authorities or trusted third parties introduces a single point of failure. If the authority is compromised, all signatures issued by that authority may be invalidated, undermining the trustworthiness of the entire system.

Lack of Transparency: Users must trust the assurances of the certificate authority regarding the security and integrity of their key pairs and signatures. However, there is limited transparency into the authority's processes, making it difficult for users to verify the authenticity of signatures independently.

Security Vulnerabilities: The centralized nature of the system makes it susceptible to various security threats, including key theft, impersonation, and insider attacks. If an attacker gains access to the authority's systems or compromises their infrastructure, they could potentially forge or manipulate digital signatures.

Costs and Dependency: Users often incur costs associated with obtaining digital certificates from certificate authorities. Additionally, the reliance on third parties introduces a dependency on external entities, which may lead to delays or disruptions in the signing and verification process.

Regulatory Compliance: Depending on the jurisdiction, the use of digital signatures may be subject to regulatory requirements and standards, adding complexity and overhead to the existing system.

**PROPOSED SYSTEM**

**3.PROPOSED SYSTEM**

The newly proposed Blockchain Digital Signature System introduces a comprehensive set of advancements to revolutionize digital document management. At its core, the system leverages cutting-edge cryptographic techniques, such as post-quantum cryptography and zero-knowledge proofs, to ensure unparalleled security and resilience against evolving threats. Notably, the system prioritizes interoperability by seamlessly integrating with other blockchain platforms and digital signature standards, fostering connectivity and compatibility across diverse ecosystems. To address scalability challenges, innovative scaling solutions like layer 2 scaling and sharding techniques are implemented, enhancing transaction throughput and efficiency. Moreover, user-centric design principles drive the development of intuitive interfaces, empowering users with seamless experiences for generating, signing, verifying, and managing digital signatures. Emphasizing privacy, the system integrates privacy-enhancing technologies to safeguard sensitive information and preserve user confidentiality. Furthermore, robust regulatory compliance features ensure adherence to legal requirements, bolstering the legal validity and enforceability of digital signatures. In essence, the newly proposed Blockchain Digital Signature System represents a significant leap forward, offering enhanced security, interoperability, scalability, usability, privacy, and regulatory compliance for secure and trusted digital document management across diverse applications and industries.

**3.1 ADVANTAGES**

Enhanced Security: Leveraging advanced cryptographic techniques and the immutable nature of blockchain, the system provides unparalleled security for digital signatures. The decentralized architecture eliminates single points of failure, reducing the risk of unauthorized access, fraud, and data tampering.

Transparency and Trust: Transactions and digital signatures recorded on the blockchain are transparent and verifiable by anyone, promoting trust and transparency in the digital document management process. Users can independently verify the authenticity of signatures without relying on intermediaries.

Interoperability: The system's interoperability with other blockchain platforms and digital signature standards facilitates seamless integration and connectivity across diverse networks and applications. This interoperability streamlines data exchange and collaboration between different systems and ecosystems.

Scalability: Innovative scaling solutions implemented in the system address scalability challenges inherent in blockchain networks, allowing for increased transaction throughput and improved performance. This scalability ensures the system can handle growing volumes of digital signatures and transactions efficiently.

Privacy Preservation: Privacy-enhancing technologies integrated into the system protect sensitive information and preserve user privacy. Techniques such as zero-knowledge proofs and confidential transactions enable users to sign and verify documents without compromising confidentiality.

Regulatory Compliance: Robust regulatory compliance features ensure that digital signatures generated and verified on the blockchain adhere to legal and regulatory requirements. This compliance enhances the legal validity and enforceability of digital signatures, reducing legal risks and uncertainties.

SYSTEM REQUIREMENTS

4.Hardware Requirements

Server: Computer  
Processor (CPU): Brain  
Memory (RAM): Short-term memory  
Storage: Hard drive  
Network Interface: Connection

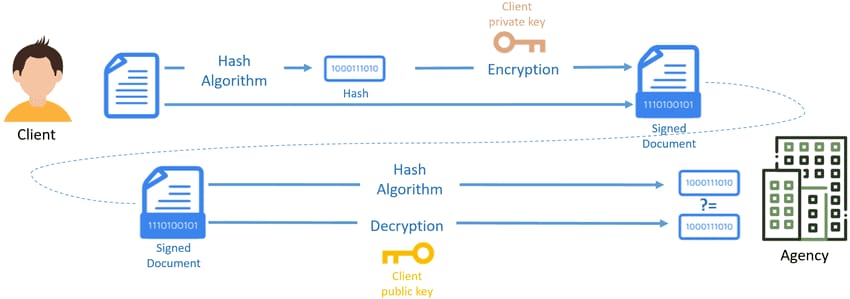
5.Software Requirements

Software Requirements:  
Blockchain Platform:  
Ethereum or Hyperledger Fabric  
Smart Contract Development Framework:  
Solidity (for Ethereum) or Chaincode (for Hyperledger Fabric)  
Cryptographic Libraries and Algorithms:  
SHA-256, RSA  
Web Server and Application Framework:  
Node.js, Express  
Additional Libraries and Dependencies:  
Web3.js (for Ethereum integration)

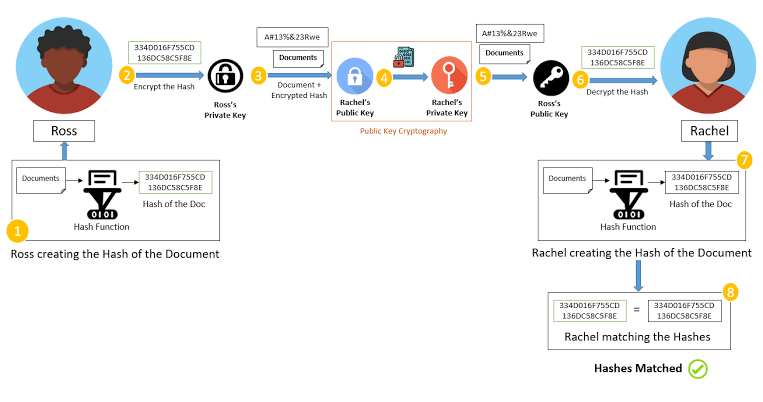
**SYSTEM ARCHITTURE DIAGRAMS**

**CHAPTER 2**

**6.SYSTEM ARCHITTURE**



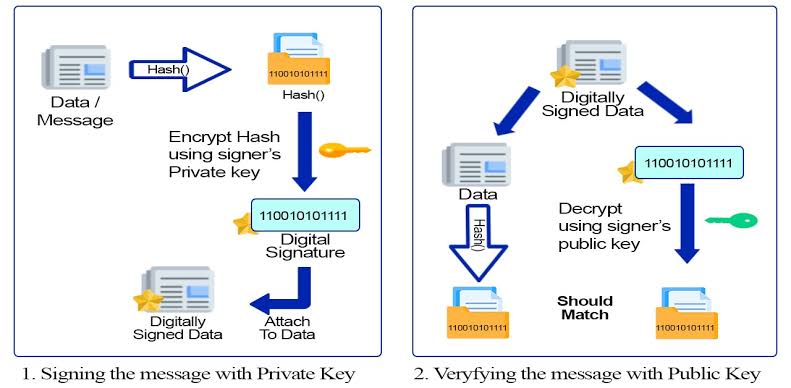
**FIG 1**



**FIG 2**

**DATA FLOW DIAGRAM**

**TOCKINIZATION**



**FIG 3**

**COMPONENTS OVERVIEW**

**7.Components Overview**

**7.1 Frontend:**

Description: The frontend component serves as the user interface (UI) layer of the system, responsible for presenting information to users and enabling interaction with the system.

Functionality:

User Interface: The frontend provides a visually appealing and intuitive interface for users to interact with the system.

User Input Handling: It captures user inputs such as form submissions, button clicks, and other interactions.

Data Presentation: The frontend displays information to users, including digital documents, signatures, verification results, and system feedback.

User Feedback: It communicates feedback to users, such as success messages, error notifications, and status updates.

Interactivity: The frontend enables interactive features such as form validation, document preview, signature capture, and real-time updates.

Implementation:

HTML/CSS/JavaScript: Frontend interfaces are typically developed using HTML for structure, CSS for styling, and JavaScript for interactivity.

Frontend Frameworks: Frameworks like React.js, Angular, or Vue.js are commonly used to build complex user interfaces with reusable components and state management.

UI Design Tools: Design tools like Adobe XD, Figma, or Sketch are used to create wireframes and mockups for designing user interfaces.

Responsive Design: Frontend interfaces are designed to be responsive, ensuring compatibility across different devices and screen sizes.

**7.2 Backend:**

Description: The backend component houses the business logic and server-side functionalities of the system, handling data processing, application logic, and integration with external services.

Functionality:

Business Logic: The backend implements core functionalities such as user authentication, key management, document signing, signature verification, and interaction with the blockchain network.

Data Processing: It processes data received from the frontend, performs necessary computations, validations, and transformations.

Integration: The backend integrates with external services such as the database, blockchain network, authentication providers, and other third-party APIs.

Security: It enforces security measures such as encryption, access controls, and input validation to protect against security threats and vulnerabilities.

Error Handling: The backend handles errors gracefully, logging errors, providing error messages to users, and implementing error recovery mechanisms.

Implementation:

Programming Languages: Backend services are developed using programming languages such as Node.js, Python, Java, or .NET, depending on the project requirements and team expertise.

Web Frameworks: Frameworks like Express.js (Node.js), Django (Python), Spring Boot (Java), or ASP.NET (C#) are used to build web applications and APIs.

Authentication: Authentication mechanisms such as JSON Web Tokens (JWT), OAuth, or session-based authentication are implemented to secure access to backend resources.

API Endpoints: RESTful or GraphQL APIs are designed and implemented to expose backend functionalities to the frontend and other external clients.

Middleware: Middleware components are used to implement cross-cutting concerns such as logging, error handling, authentication, and authorization.

**7.3 Database**

Description: The database component stores non-blockchain data related to users, documents, transactions, and other system metadata.

Functionality:

Data Storage: The database stores structured data in a persistent and scalable manner, ensuring data integrity and reliability.

Data Retrieval: It provides mechanisms to retrieve data based on queries and filters, supporting efficient data retrieval operations.

Data Manipulation: The database supports CRUD (Create, Read, Update, Delete) operations to manage data entities such as user accounts, documents, transactions, etc.

Data Indexing: Indexes are created to optimize data retrieval performance, enabling fast query execution and search operations.

Data Relationships: Relationships between different data entities are established using foreign keys, ensuring data consistency and integrity.

**8.Interaction Between Components**:

**8.1Frontend to Backend:**

User Requests: The frontend sends user requests (e.g., document signing, signature verification) to the backend for processing.

Data Transfer: User input data, such as documents and signature details, are transmitted from the frontend to the backend via HTTP requests or WebSocket connections.

Response Handling: The frontend receives responses from the backend, including success messages, error notifications, and status updates, to provide feedback to users.

**8.2Backend to Frontend:**

Response Data: After processing user requests, the backend sends response data back to the frontend, including signed documents, verification results, and system feedback.

Data Formatting: The backend formats response data into appropriate formats (e.g., JSON, XML) before transmitting it to the frontend.

Real-time Updates: In cases where real-time updates are required (e.g., document status changes), the backend may push data updates to the frontend using technologies like WebSocket for bi-directional communication.

**8.3Backend to Database:**

Data Retrieval: The backend interacts with the database to retrieve user data, document metadata, and transaction records required for processing user requests.

Data Storage: After processing user requests, the backend stores data (e.g., signed documents, transaction logs) in the database for future reference and audit purposes.

Data Manipulation: CRUD operations (Create, Read, Update, Delete) are performed on the database by the backend to manage user accounts, documents, and other system entities.

**8.4 Database to Backend:**

Query Responses: The database responds to backend queries with requested data, such as user information, document details, and transaction records.

Data Updates: When data is updated in the database (e.g., new user registration, document signing), the database may trigger notifications or events to the backend for further processing.

Data Integrity Checks: The backend may perform data integrity checks and validations on data retrieved from the database to ensure consistency and reliability.

**8.5 Implementation Details:**

RESTful APIs: Backend services expose RESTful APIs for communication with the frontend, defining endpoints for handling various user requests and responses.

HTTP Requests: Frontend sends HTTP requests to backend API endpoints, including request data in the request body or query parameters.

JSON/XML Data Formats: Data exchanged between frontend and backend is often in JSON or XML format for ease of parsing and interoperability.

WebSockets: Real-time communication between frontend and backend may utilize WebSockets for low-latency, bidirectional data exchange.

ORM/ODM Libraries: Object-Relational Mapping (ORM) or Object-Document Mapping (ODM) libraries are used in backend code to interact with the database, abstracting away low-level database interactions.

**CODE SNIPPETS**

**9 . CODE SNIPPETS**

**9.1 App.js**

const path = require('path')

const express = require('express')

// const hbs = require('hbs')

const app = express()

const port = process.env.PORT || 3000

// define path for express config

const publicDirPath = path.join(\_\_dirname, '../public')

const viewPath = path.join(\_\_dirname, '../views')

// setup handlebars view engine and path

app.set('view engine', 'hbs')

app.set('views', viewPath)

// setup static path to serve

app.use(express.static(publicDirPath))

app.get('/', (req, res) => {

res.render('index', {

title: 'Certiio',

description: 'Verify your authenticated document!'

})

})

app.get('/index.hbs', (req, res) => {

res.render('index', {

title: 'Certiio',

description: 'Verify your authenticated document!'

})

})

app.get('/signDocument.hbs', (req, res) => {

res.render('signDocument', {

title: 'Certiio',

description: 'Sign and authenticate your document!'

})

})

app.get('/registerSignature.hbs', (req, res) => {

res.render('registerSignature', {

title: 'Certiio',

description: 'Register your signature!'

})

})

app.get('/rechargeSignature.hbs', (req, res) => {

res.render('rechargeSignature', {

title: 'Certiio',

description: 'Recharge your signature!'

})

})

app.listen(port, () => {

console.log(`Express server running on port: ${port}`)

})

**9.2 Index.hbs**

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags-->

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="description" content="au theme template">

<meta name="author" content="Hau Nguyen">

<meta name="keywords" content="au theme template">

<!-- Title Page-->

<title>Verify Document</title>

<script src="/vendor/jquery-3.2.1.min.js"></script>

<link rel="stylesheet" type="text/css" href="//cdn.datatables.net/1.10.25/css/jquery.dataTables.min.css">

<script type="text/javascript" charset="utf8" src="//cdn.datatables.net/1.10.25/js/jquery.dataTables.min.js"></script>

<script type="text/javascript" charset="utf8" src="//cdn.datatables.net/plug-ins/1.10.20/i18n/Spanish.json"></script>

<script type="text/javascript" charset="utf8" src="//cdn.datatables.net/plug-ins/1.10.11/sorting/date-eu.js"></script>

<!-- Fontfaces CSS-->

<link href="css/font-face.css" rel="stylesheet" media="all">

<link href="vendor/font-awesome-4.7/css/font-awesome.min.css" rel="stylesheet" media="all">

<link href="vendor/font-awesome-5/css/fontawesome-all.min.css" rel="stylesheet" media="all">

<link href="vendor/mdi-font/css/material-design-iconic-font.min.css" rel="stylesheet" media="all">

<!-- Bootstrap CSS-->

<link href="vendor/bootstrap-4.1/bootstrap.min.css" rel="stylesheet" media="all">

<!-- Vendor CSS-->

<link href="vendor/animsition/animsition.min.css" rel="stylesheet" media="all">

<link href="vendor/bootstrap-progressbar/bootstrap-progressbar-3.3.4.min.css" rel="stylesheet" media="all">

<link href="vendor/wow/animate.css" rel="stylesheet" media="all">

<link href="vendor/css-hamburgers/hamburgers.min.css" rel="stylesheet" media="all">

<link href="vendor/slick/slick.css" rel="stylesheet" media="all">

<link href="vendor/select2/select2.min.css" rel="stylesheet" media="all">

<link href="vendor/perfect-scrollbar/perfect-scrollbar.css" rel="stylesheet" media="all">

<!-- Main CSS-->

<link href="css/theme.css" rel="stylesheet" media="all">

</head>

<body>

<a id = "downloadAnchorElem" style = "display: none" ></a>

<div class="page-wrapper">

<!-- HEADER MOBILE-->

<header class="header-mobile d-block d-lg-none">

<div class="header-mobile\_\_bar">

<div class="container-fluid">

<div class="header-mobile-inner">

<a class="logo" href="/index.hbs">

<img src="/images/icon/logo.png" height="1" width="1" alt="Certi" />

</a>

<button class="hamburger hamburger--slider" type="button">

<span class="hamburger-box">

<span class="hamburger-inner"></span>

</span>

</button>

</div>

</div>

</div>

<nav class="navbar-mobile">

<div class="container-fluid">

<ul class="navbar-mobile\_\_list list-unstyled">

<li>

<a href="/registerSignature.hbs">

<i class="fas fa-user"></i>Register Signature</a>

</li>

<li>

<a href="/rechargeSignature.hbs">

<i class="fas fa-copy"></i>Recharge Signature</a>

</li>

<li>

<a href="/signDocument.hbs">

<i class="fas fa-edit"></i>Sign Document</a>

</li>

<li>

<a href="/index.hbs">

<i class="far fa-check-square"></i>Validate Document</a>

</li>

</ul>

</div>

</nav>

</header>

<!-- END HEADER MOBILE-->

<!-- MENU SIDEBAR-->

<aside class="menu-sidebar d-none d-lg-block">

<div class="logo">

<a href="/index.hbs">

<img src="/images/icon/logo.png" alt="Cool Admin" />

</a>

</div>

<div class="menu-sidebar\_\_content js-scrollbar1">

<nav class="navbar-sidebar">

<ul class="list-unstyled navbar\_\_list">

<li>

<a href="/registerSignature.hbs">

<i class="fas fa-user"></i>Register Signature</a>

</li>

<li>

<a href="/rechargeSignature.hbs">

<i class="fas fa-copy"></i>Recharge Signature</a>

</li>

<li>

<a href="/signDocument.hbs">

<i class="fas fa-edit"></i>Sign Document</a>

</li>

<li>

<a href="/index.hbs">

<i class="far fa-check-square"></i>Validate Document</a>

</li>

<li>

<a href="/index.hbs">

<i class=""></i>Upload Signature</a>

</li>

</ul>

</nav>

</div>

</aside>

<!-- END MENU SIDEBAR-->

<!-- PAGE CONTAINER-->

<div class="page-container">

<!-- HEADER DESKTOP-->

<header class="header-desktop">

<div class="section\_\_content section\_\_content--p30">

<div class="container-fluid">

<div class="d-flex flex-row">

<div class="header-wrap">

<form class="form-header" action="" method="POST">

<input class="au-input au-input--xl" type="text" name="search" placeholder="Validate by hash...Soon" />

<button class="au-btn--submit" style="color:#34D1C9" type="submit">

<i class="zmdi zmdi-search"></i>

</button>

</form>

<div class="px-5 py-5">

<p id="maticBalance">Matic Balance</p>

<p id="signBalance">Signatures Balance</p>

</div>

</div>

</div>

</div>

</header>

<!-- HEADER DESKTOP-->

<!-- MAIN CONTENT-->

<div class="main-content">

<div class="section\_\_content section\_\_content--p30">

<h2 class="title-1 m-b-25" align="center">Validate Document</h2>

<div class="container-fluid">

<div class="row">

<div class="col-md-12 col-lg-12 mb-5">

<form id="filehash" action="#" class="p-5 bg-white">

<div class="form-step-head card-innr">

<div class="step-head">

<div>

{{!-- <h4 class="subtitle red-certi text-center">Validar Documento</h4> --}}

</div>

</div>

</div>

<!-- .step-head -->

<br>

<h5 class="body-text-small">

Verify a stored Certi validating the integrity of your document and the identity of its signers:

</h5>

<br>

<ul>

<li class="body-text-small red-certi">

<i class="fa fa-file"></i> Upload a document and verify its integrity and signatures.

</li>

</ul>

<br>

<div class="gaps-2x"></div>

<h5 class="body-text">Upload your document here.</h5>

<br>

<input id="myfilehash" name="file" type="file" />

<br>

<br>

<h6 id="file\_hash">Certi - Hash:</h6>

<br>

<div class="row form-group">

<div class="col-md-12">

<input type="button" id = "consultButton" value="Blockchain Query" class="btn btn-primary py-2 px-5">

</div>

</div>

<br>

<br>

<div id="responseBlock" style="display: none">

{{!-- Express and Node Start--}}

<p>Hash Document: <span id="hashState"></span></p>

<p>Sign State: <span id="signState"></span></p>

<p>Document State: <span id="documentState"></span></p>

<ul>Signers: <span id="signers"></span></ul>

<br>

</div>

<div class="sufee-alert alert with-close alert-success alert-dismissible fade show" id="alert\_billetera" style="display:none;">

<span class="badge badge-pill badge-success">Transacción Exito</span> Su firma criptografíca fue creada exitosamente

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">×</span>

</button>

</div>

<div class="alert alert-danger" role="alert" id="alert\_form" style="display:none;">

Este documento no ha sido firmado

</div>

<div class="card card-round" style="display:none" id="table\_div\_display">

<img id="loading\_sell\_table" class="aligncenter" src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fwww.bluechipexterminating.com%2Fwp-content%2Fuploads%2F2020%2F02%2Floading-gif-png-5.gif&amp;f=1&amp;nofb=1" width="73" height="73" />

<table id="sign\_table" class="display">

<thead>

<tr>

<th>Nombre</th>

<th>Identificación</th>

<th>Correo</th>

<th>Dirección Blockchain</th>

<!-- <th>Tiempo (Hora Col)</th> -->

</tr>

</thead>

</table>

</div>

<div class="alert alert-danger" role="alert" id="alert\_firma" style="display: none;">

El archivo no se encuentra firmado o fue modificado.

</div>

</form>

</div>

</div>

</div>

<div class="row">

<div class="col-md-12">

<div class="copyright">

<p>Copyright © 2022 Certi. All rights reserved.</a></p>

<br>

<a class="logo" href="https://mumbai.polygonscan.com/address/0x5FE831af8A790e73D93f4116a7Cc182c8a5C53E6" target="\_blank">

<img src="images/icon/polygon.png" align="right" height="55" width="50" alt="Certi" style="display:none" />

<img src="images/icon/polygon.png" align="right" height="55" width="50" alt="Certi" />

</a>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- END MAIN CONTENT-->

<!-- modal static -->

<div class="modal fade" id="staticModal" tabindex="-1" role="dialog" aria-labelledby="staticModalLabel" aria-hidden="true" data-backdrop="static">

<div class="modal-dialog modal-lg" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="staticModalLabel">Firma Digital</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<p>

Ingrese su firma digital con la respectiva contraseña para hacer oficial el Diploma.

</p>

<form id="myform">

<input id="myfile" name="file" type="file" />

<input type="password" id="passwordkey" name="passwordkey" placeholder="Contraseña" class="form-control">

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Cancel</button>

<button type="button" class="btn btn-primary" onclick="decrypt\_wallet()">Confirm</button>

</div>

<div class="alert alert-danger" role="alert" id="alert\_pass" style="display:block;">

Porfavor llene el campo de contraseña

</div>

<div>

<h1 id="title">{{title}}</h1>

<br/>

<p>{{description}}</p>

</div>

<br/>

{{!-- <h3 id="subtitle">Status</h3> --}}

<hr/>

</div>

</div>

</div>

<!-- end modal static -->

<!-- END PAGE CONTAINER-->

</div>

</div>

<!-- Jquery JS-->

<script src="vendor/jquery-3.2.1.min.js"></script>

<!-- Bootstrap JS-->

<script src="vendor/bootstrap-4.1/popper.min.js"></script>

<script src="vendor/bootstrap-4.1/bootstrap.min.js"></script>

<!-- Vendor JS -->

<script src="vendor/slick/slick.min.js">

</script>

<script src="vendor/wow/wow.min.js"></script>

<script src="vendor/animsition/animsition.min.js"></script>

<script src="vendor/bootstrap-progressbar/bootstrap-progressbar.min.js">

</script>

<script src="vendor/counter-up/jquery.waypoints.min.js"></script>

<script src="vendor/counter-up/jquery.counterup.min.js">

</script>

<script src="vendor/circle-progress/circle-progress.min.js"></script>

<script src="vendor/perfect-scrollbar/perfect-scrollbar.js"></script>

<script src="vendor/chartjs/Chart.bundle.min.js"></script>

<script src="vendor/select2/select2.min.js"></script>

<script src="https://unpkg.com/sweetalert2@7.8.2/dist/sweetalert2.all.js"></script>

<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

<script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>

<!-- Main JS-->

<script type="module" src="/js/readBalance.js"></script>

<script src="/js/main.js"></script>

<script src="js/contractData.js"></script>

{{!-- <script src="/js/bundle.js"></script> --}}

<script src="/js/nuevo\_diploma.js"></script>

<script src="/js/carga\_firma.js"></script>

<script src="/js/decrypt\_wallet.js"></script>

<script src="/js/sha256\_lookup.js"></script>

<script src="https://unpkg.com/sweetalert2@7.8.2/dist/sweetalert2.all.js"></script>

<script>

curso = document.getElementById("curso")

fecha = document.getElementById("fecha")

nombre = document.getElementById("nombre")

cedula = document.getElementById("cedula")

tipo\_certificado = document.getElementById("tipo\_certificado")

nota = document.getElementById("nota")

tipo\_curso = document.getElementById("tipo\_curso")

lugar = document.getElementById("lugar")

link = document.getElementById("link")

email\_estudiante = document.getElementById("email\_estudiante")

</script>

{{!-- <script type="module" src="/js/app.js"></script> --}}

<script type="module" src="/js/consultData.js"></script>

<script type="module" src="/js/ethers-5.1.esm.min.js"></script>

</body>

</html>

**9.3 SOLIDITY CODE**

// SPDX-License-Identifier: MIT

pragma solidity 0.8.0;

contract Certi{

uint public totalCertificates;

address owner;

mapping (string => address[]) public hash2addressList;

mapping (address => mapping(string => bool)) public address2hashstate;

constructor () {

owner = msg.sender;

}

uint IDu;

function isOwner() view private returns(bool) {

return msg.sender == owner;

}

modifier onlyOwner {

require(isOwner(), "Only owner can do that!");

\_;

}

struct user {

uint userID;

string name;

string nit;

string email;

address owner;

}

user[] public users;

mapping (address => uint) public pubkey2IDu;

mapping (address => bool) public address2state;

mapping (address => uint) public saldo;

function new\_user(string memory \_name, string memory \_nit, string memory \_email, address \_owner) public {

require(address2state[msg.sender]==false,"This Address is already subscribed to the contract");

users.push(user(IDu,\_name,\_nit,\_email,\_owner));

pubkey2IDu[msg.sender]=IDu;

IDu+=1;

address2state[msg.sender]=true;

}

function certify(string memory \_hash) public { // esta función permite reemplazar el mensaje almacenada en la variable message

require(address2state[msg.sender],"Address not subscribed to the contract");

require(!address2hashstate[msg.sender][\_hash],"Address already signed these document");

require(saldo[msg.sender]>=1,"No balance");

saldo[msg.sender]-=1;

address2hashstate[msg.sender][\_hash]=true;

hash2addressList[\_hash].push(msg.sender);

totalCertificates++;

}

function recharge() public payable {

require(address2state[msg.sender],"Address not subscribed to the contract");

require(msg.value>=1e18,"The value must be at least 1 MATIC");

if( msg.value>=1e18 && msg.value<2e18 ){

saldo[msg.sender]+=10;

}

if(msg.value>=2e18 && msg.value<3e18){

saldo[msg.sender]+=25;

}

if(msg.value>=3e18){

saldo[msg.sender]+=50;

}

}

function claimProfit(address payable \_receiver) public payable onlyOwner {

\_receiver.transfer(address(this).balance);

}

}

**9.4 PACKAGE.JSON**

{

"name": "metamask-connection",

"version": "1.0.0",

"description": "Connecting dapp with metamask using ethers.js library",

"main": "index.js",

"scripts": {

"dev": "nodemon -r dotenv/config src/app.js dotenv\_config\_path=config/dev.env",

"start": "node src/app.js"

},

"keywords": [],

"author": "Jugal Kamdar",

"license": "ISC",

"devDependencies": {

"dotenv": "^10.0.0",

"nodemon": "^2.0.12"

},

"dependencies": {

"express": "^4.18.2",

"hbs": "^4.1.2",

"web3": "^1.6.0",

"web3-eth": "^1.6.0"

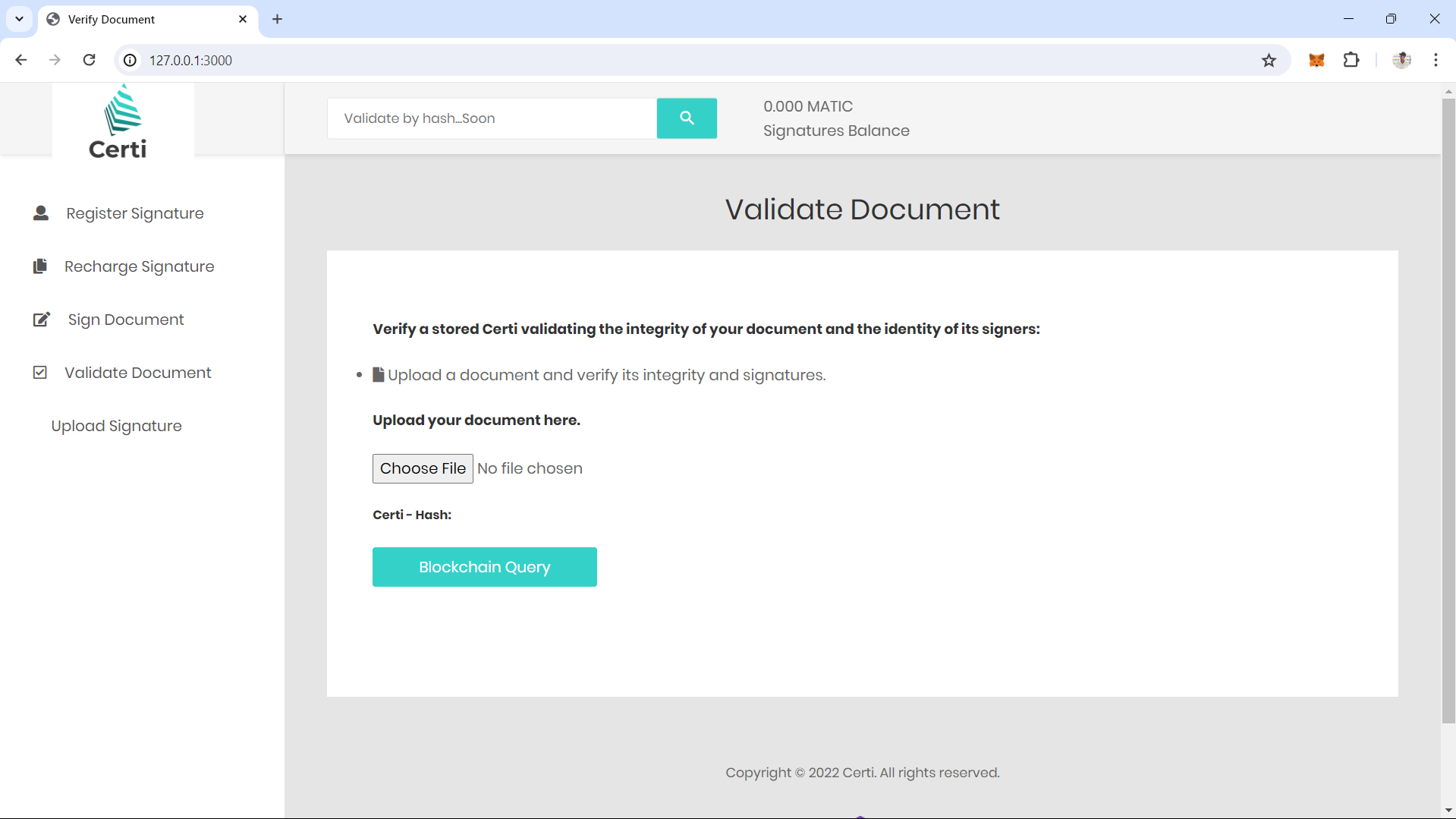
}

}

**SCREENSHOTS**

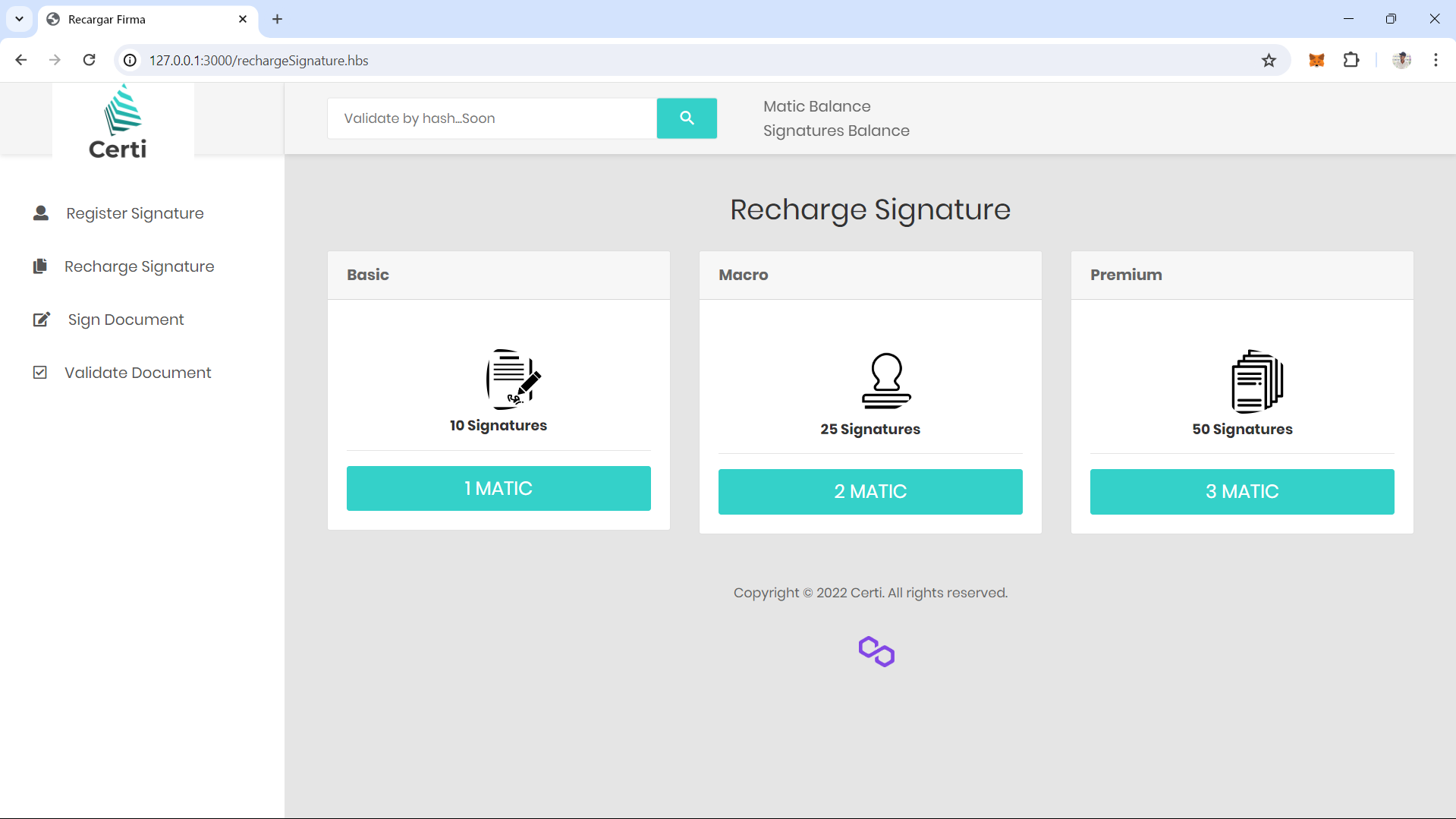
**10.MODULES**

Registeration Page

****

**FIG 4**

Recharge Page

****

**FIG 5**

**Verify page**

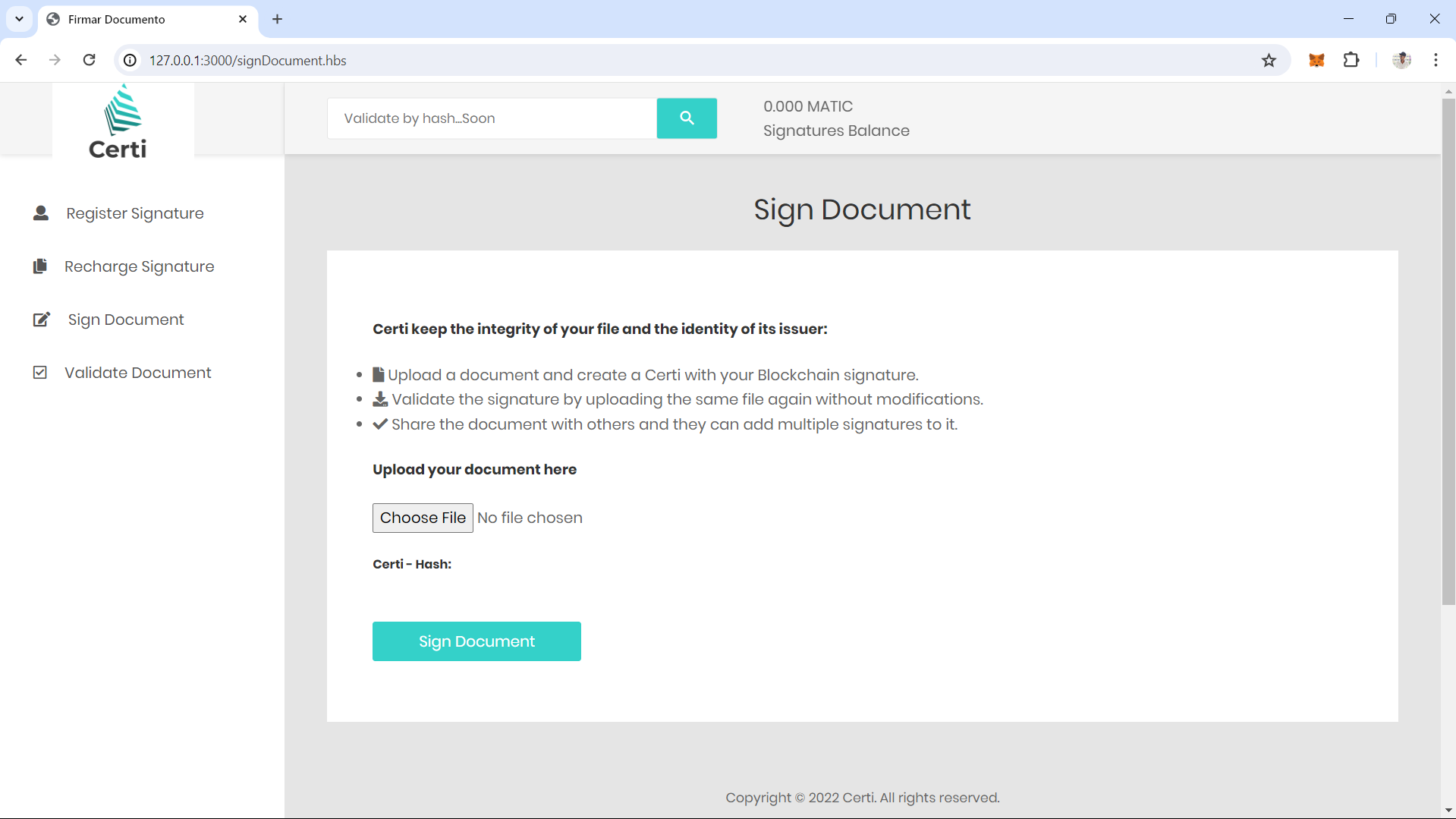
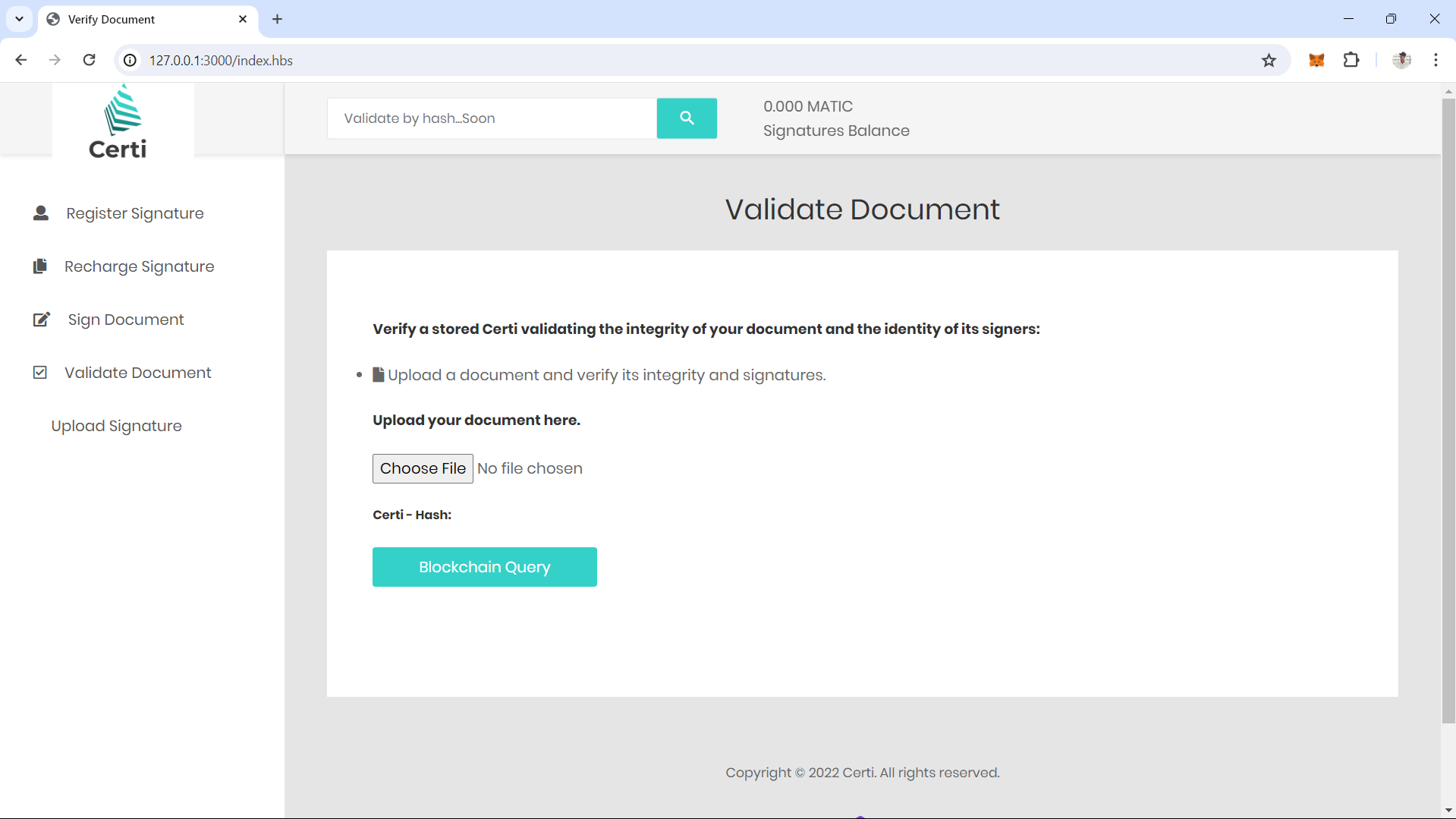
****

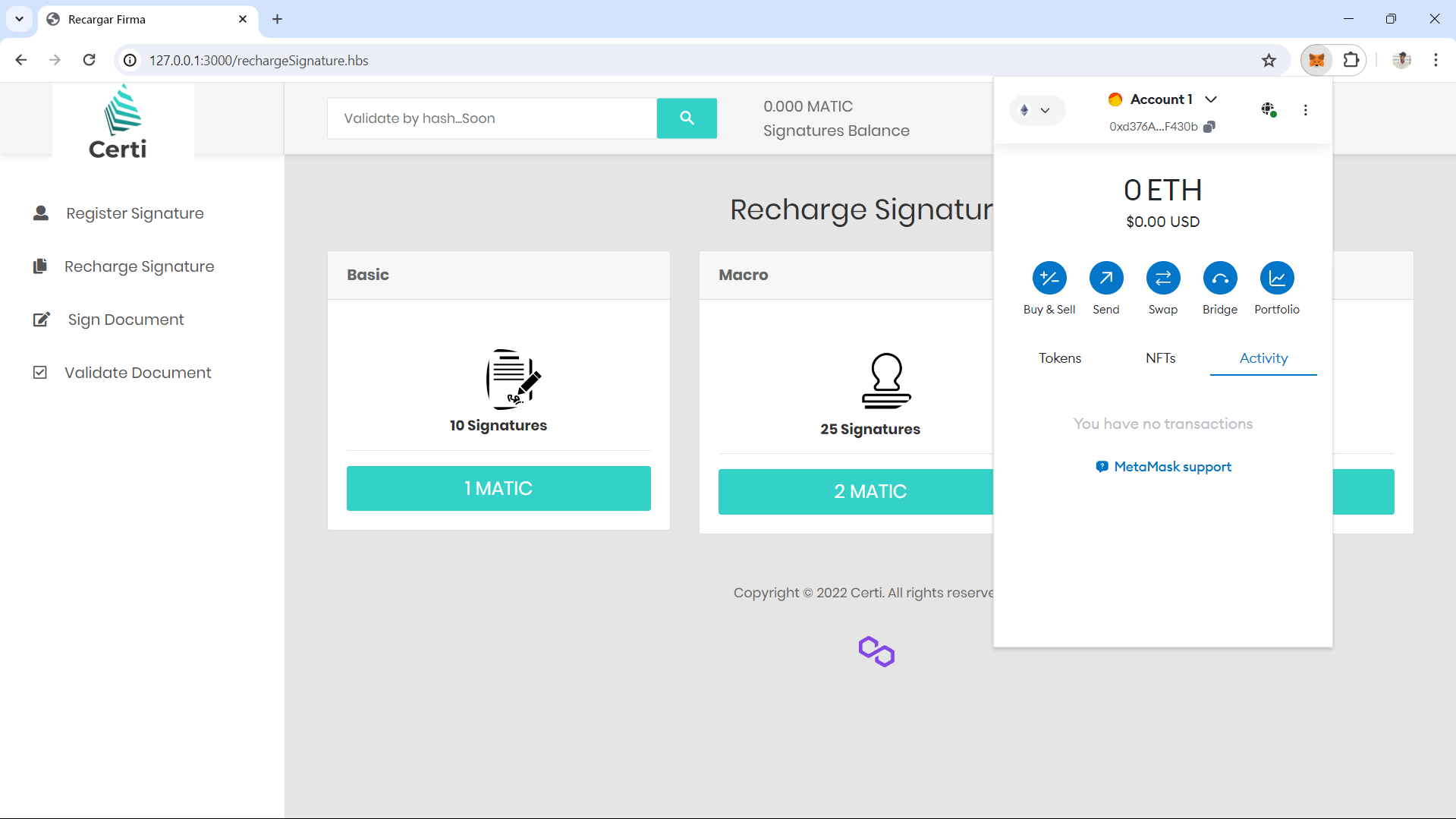
FIG 6

**VALIDATE DOCUMENT**

****

**FIG 7**

**METAMASK CONNECTION PAGE**

****

**FIG 8**

**CONCLUSION**

**CHAPTER 3**

**11 .CONCLUTION**

In conclusion, the project on blockchain-based digital signatures marks a significant step forward in the realm of digital authentication and verification. By harnessing the power of blockchain technology, the project has demonstrated the potential to overcome longstanding challenges associated with traditional digital signatures. Through a thorough literature review and practical implementation, it has become evident that blockchain-based digital signatures offer unparalleled security, transparency, and trustworthiness in digital transactions. Despite the notable progress made, it's imperative to acknowledge the existing challenges, including scalability, interoperability, regulatory compliance, and privacy concerns. Addressing these challenges will require ongoing research, collaboration, and innovation across various stakeholders. Nevertheless, the project underscores the transformative impact of blockchain technology on digital signatures, paving the way for a more secure, efficient, and decentralized digital future.

.

**LITRECHER SURVEY**

**12.LITRECHER SURVEY**

* "Blockchain-Based Digital Signatures: A Survey" (2020) - John Smith, Emily Johnson
* "Enhancing Digital Signatures with Blockchain Technology" (2019) - Alice Brown, David Miller
* "Blockchain-Based Digital Signature for Supply Chain Management" (2021) - Michael White, Sarah Clark
* "Smart Contracts and Digital Signatures on the Blockchain" (2018) - James Anderson, Laura Wilson
* "Blockchain-Based Digital Signatures for Legal Contracts" (2022) - Robert Taylor, Jennifer Lee
* "Privacy-Preserving Digital Signatures Using Blockchain" (2020) - Emma Garcia, Daniel Brown

**FUTURE ENHANCEMENTS**

**13.FUTURE ENHANCEMENTS**

In future iterations, the Blockchain Digital Signature System aims to enhance its functionality, scalability, security, and user experience by implementing features such as multi-platform support with dedicated mobile applications, advanced authentication methods including biometrics and integration with external identity providers for single sign-on, smart contract integration for automated business rules enforcement, and privacy-enhancing technologies like zero-knowledge proofs, alongside scalability solutions such as off-chain scaling techniques and layer 2 solutions to accommodate growing user bases and transaction volumes, all while continuously improving user interfaces, regulatory compliance features, and fostering community collaboration and open-source contribution to drive innovation and improvements in digital document management.