

DEPARTMENT OF APPLIED INFORMATICS
UNIVERSITY OF MACEDONIA

**Development of a Web Application with Artificial Intelligence
Chatbot and Decentralized Database:**

NexusChat

ANASTASIADIS SOTIRIOS

THESIS

Supervising Professor : Christos K. Georgiadis

Abstract

NexusChat represents a new era in the category of ChatBot applications and overall, enabling interaction with an artificial intelligence model. It offers functionalities such as the storage of messages and notes for future use, while implementing an innovative approach for the data storage, which is the most important point of the application. NexusChat's pioneering data storage approach is achieved through the decentralization of data on the Sui Blockchain network. The primary objective of NexusChat is to apply data decentralization in an environment that typically relies on centralized servers. This method ensures the user enhanced security, confidentiality, and data portability, without the need to depend on third parties or the risk of privacy breaches.

Content

1 Introduction.....	6
1.1 Importance of the Topic.....	6
1.2 Purpose.....	6
1.3 Basic Terminology.....	7
1.3.1 Blockchain.....	7
1.3.2 Sui Blockchain Network.....	7
1.3.3 Smart Contract.....	8
1.3.4 Sui Move.....	8
1.3.5 Digital Object.....	9
1.3.6 Cryptographic Wallet.....	9
1.3.7 Decentralized Application (dApp).....	10
2 NexusChat.....	11
2.1 Introduction to NexusChat.....	11
2.2 Core Features and Functionalities.....	12
2.2.1 Wallet-Based Authentication & Accountless Access.....	12
2.2.2 AI Interaction with Data Control (ChatGPT Integration).....	12
2.2.3 Decentralized & User-Owned Chat History.....	14
2.2.4 Decentralized Note System.....	15
2.2.5 Data Encryption.....	16
3 Conceptual Architecture & System Overview.....	17
3.1 Introduction.....	17
3.2 Development Methodology and Philosophy.....	17
3.3 High Level System Architecture.....	18
3.3.1 Frontend (Client-Side): Reactjs (Typescript) and Libraries.....	18
3.3.2 Backend (Server-Side): .NET (ASP.NET Web API Core).....	21
3.3.3 Blockchain Layer: Sui Move Smart Contract.....	23
4 System Demonstration and User Workflow.....	26
4.1 Introduction.....	26
4.2 Wallet-Based Authentication and Session Initiation.....	27
4.3 Core AI Interaction and On-Chain Data Persistence.....	33
4.4 Archiving Information: The Decentralized Note System.....	34
5 Conclusion and Future Work.....	39
5.1 Introduction.....	40
5.2 Future Work and Potential Enhancements.....	40
6 Acknowledgements.....	41

Table of Images

Image 3.1: User Provider Code.....	19
Image 3.2: Transaction Services Code.....	20
Image 3.3: TailwindCss Example.....	20
Image 3.4: JSON-RPC Interaction Example.....	21
Image 3.5: AES Decryption Function.....	22
Image 3.6: Message Struct Type.....	23
Image 3.7: Chat Struct Type.....	24
Image 3.8: Folder Struct Type.....	24
Image 3.9: Note Struct Type.....	24
Image 3.10: On-Chain Function.....	25
Image 4.1: Home Page.....	27
Image 4.2: Side Navigation Menu with the “Connect” Action Button.....	28
Image 4.3: list of all detected Sui-compatible wallet extensions.....	29
Image 4.4: Wallet Authorization for application access.....	30
Image 4.5: Authenticated State with application Features.....	31
Image 4.6: Chat Page.....	32
Image 4.7: AI Interaction and Wallet authorization.....	33
Image 4.8: Notepad Modal.....	34
Image 4.9: Create a Folder.....	35
Image 4.10: Folder Creation.....	36
Image 4.11: Folders Display in Menu.....	37
Image 4.12: Notepad Page with Folders.....	38
Image 4.13: Note Display.....	39

Prologue

The most important aspect of being human is the ability to dream. Dreams are the power that drives us forward, urging us to transcend our limits and discover new possibilities. Without dreams, progress ceases, and challenges appear like a mountain. This is why every major innovation always begins with a vision—a dream that empowers us to create something pioneering, something new, something better!

1 Introduction

1.1 Importance of the Topic

In the contemporary digital world we live in, while the AI models and chatbot applications are increasingly prevalent, a significant majority still rely on the traditional centralized data centers (Tatipatri et al, 2024). This conventional approach, while common, often presents vulnerabilities related to data breaches and users' privacy. The NexusChat web application emerges as the result of these challenges, introducing a new era in how personal information is managed and secured.

The most significant and different features of NexusChat lie in its unique data storage mechanism: the decentralization of user data into the Sui Blockchain Network. Unlike common centralized data centers, where user data (such as chat messages and personal notes) is aggregated on company-controlled servers, NexusChat atomizes this data into discrete, individually owned digital assets. When the user interacts with the application, their data is broken into encrypted objects, which are transferred into the Sui Blockchain. The ownership and the control of these data objects being identified by unique identifiers, are directly tied to the user's cryptographic wallet. This means the user has full ownership and control over their individual data "pieces".

Concluding on its foundational importance, a particularly significant advantage of NexusChat's utilization of blockchain technology is the immutability of stored data. Once user data, such as notes or chat messages, is committed to the blockchain, it becomes unalterable. This characteristic reinforces the security but also offers a distinct advantage to users who wish to retain important information generated during the interaction of the user and the AI engine. NexusChat empowers users to preserve valuable knowledge over time, secure in the understanding that their data remains both protected and verifiably tamper-proof, a crucial feature often lacking in systems where data can be lost or modified during transport.

1.2 Purpose

The innovative architecture of NexusChat not only addresses long-standing concerns of centralized vulnerabilities but also redefines how AI and data privacy can coexist. By activating true data sovereignty, the platform removes the dependency on commonly used servers and eliminates single points of failure. Especially, the AI model within NexusChat is

designed to function within strict privacy boundaries: it temporarily adapts to the user's inputs during an active session for improvement, but upon logout of the user, its access to the user's encrypted data is revoked. This ensures that no more data is retained without explicit consent. By combining AI with decentralized data storage, NexusChat sets a new privacy standard — one where personalization does not come at the cost of privacy.

1.3 Basic Terminology

1.3.1 Blockchain

Blockchain is a decentralized and distributed system, sometimes a public digital ledger consisting of a continuously growing list of records, called "blocks," where they are securely linked together using cryptography (Zheng et al., 2018). Each block typically contains a cryptographic hash of the previous block, a timestamp, and transaction data. In the context of applications like NexusChat, this data component can also change into other forms, such as messages and notes. This chained structure inherently makes blockchains resistant to modification, as altering any recorded data would require re-calculating all subsequent blocks, a computationally prohibitive task without network consensus.

1.3.2 Sui Blockchain Network

The Sui Blockchain network is a Layer 1 blockchain designed and implemented for high throughput, low latency transaction processing and asset-centric applications. Sui is developed by a team called Mysten Labs, which previously led the Novi Research team for Meta's blockchain organization and was the architects of the Diem blockchain and the Move programming language. Sui aims to address common challenges, such as scalability and usability, that earlier blockchain generations are facing (Blackshear et al., 2024).

A key feature that distinguishes Sui from others is its object-centric data model. Unlike account-based models prevalent in many other blockchains (where state is stored within smart contract accounts), Sui treats most data, including digital assets and even smart contracts themselves, as distinct objects. Each object has a unique ID, well-defined ownership, and can have its own rules for mutability and interaction. This approach, combined with Sui's consensus mechanism, which can process many transactions in parallel if they involve distinct objects, is designed to enable significant scalability.

Sui utilizes a version of the Move programming language, known as Sui Move, for writing smart contracts (Antonios Giatzis et al., 2023). This language is engineered for safety and expressiveness, particularly in managing and manipulating digital assets or objects. For applications like NexusChat, Sui Network provides a foundation to build decentralized systems where user data can be represented as individual, user-owned objects, leveraging Sui's architecture for secure, efficient, and scalable on-chain storage.

1.3.3 Smart Contract

Smart contract is a self-executing computer program or a transaction protocol stored on a blockchain. Its core consists of code that defines a precise set of logic, rules, and callable functions (Zou et al., 2019). These elements are designed to automatically execute predefined actions or enforce the terms of an agreement, either when their functions are directly invoked by an incoming transaction (often initiated by a user or another contract) or when other conditions embedded within their logic are satisfied. Once it's been deployed, the execution of a smart contract is deterministic, transparent, and tamper-resistant, with all operations and resulting state changes being validated and immutably recorded by the blockchain's consensus mechanism.

1.3.4 Sui Move

Sui Move is a specialized programming language for developing smart contracts on the Sui Network, adapted from the original Move language, designed with a focus on safety and secure asset management (Welc & Blackshear, 2023). A characteristic of Sui Move is its resource-oriented programming model, where digital assets (represented as “objects” in Sui) are treated as first-class “resources”. This model is engineered to provide guarantees about how resources are handled, ensuring precise control over their creation, transfer, and storage.

Sui Move's strict rules are significant for on-chain data management. The language dictates that resources, which can represent valuable data such as messages or notes, cannot be implicitly copied or accidentally lost; instead, they must be explicitly “moved”. This design directly helps prevent unintentional loss or duplication of such data. For this specific reason, this design is vital for applications like NexusChat that prioritize the integrity of the user information on-chain.

1.3.5 Digital Object

In the context of blockchain technology, particularly on platforms like the Sui blockchain, a Digital Object represents a fundamental unit of data and state. Sui's object-centric architecture means that most on-chain data, ranging from simple pieces of information to complex digital assets and even smart contracts themselves, are structured and managed as distinct individual objects.

Each Sui Object possesses several key characteristics such as:

- **Unique Object ID:** A globally unique identifier that distinguishes it from other objects.
- **Ownership:** Explicitly defined ownership, which can be an external account (controlled by a user's crypto wallet), another object, or shared. This is critical for establishing control and access.
- **Type and Behavior:** Defined by the Sui Move module (smart contract code) that created and governs it, dictating how the object can be interacted with, modified, or transferred. Objects can be mutable or immutable.
- **Data Payload:** The actual data or state that the object encapsulates.

NexusChat takes full advantage of the object-centric model. Each chat message or personal note can be an individual, user-owned Sui object. This approach allows for granular control and direct, verifiable ownership of each piece of data by the user. When a user interacts with the AI engine, NexusChat creates a new Sui Object containing useful data, with the ownership attributed to the user's wallet address. This is the mechanism through which NexusChat achieves decentralized, user-sovereign data storage.

1.3.6 Cryptographic Wallet

A cryptographic wallet, often referred to as a crypto wallet, is a software application or hardware device that serves as the primary interface for users to interact with blockchain networks (Mohamed, 2017). Its primary purpose is to securely store and manage a user's cryptographic key and, most importantly, the private key, which is a secret piece of data granting access to and control over the user's blockchain address and any associated digital assets or data. Wallets do not typically "store" cryptocurrency or digital assets themselves; they hold the keys that prove the ownership and authorize transactions.

When a user performs an action on the blockchain, such as sending funds, deploying a smart contract, or just interacting with a Decentralized Application (dApp), the wallet uses the private key to cryptographically sign the transaction. This digital signature serves as

verifiable proof that the transaction was authorized by the owner of the specific blockchain address.

1.3.7 Decentralized Application (dApp)

A Decentralized Application (DApp) is a software application whose backend logic, primarily implemented through smart contracts, runs on a decentralized peer-to-peer network, such as blockchain, rather than on a traditional centralized server. While the frontend user interface of a dApp can be developed using standard web or mobile technologies (e.g., React or Angular), its core operations, state management, and data interactions are executed and recorded on the distributed ledger (Antal et al., 2021).

NexusChat itself is an example of a dApp. Its core functionality related to the secure storage of user messages and notes, the management of ownership of these data objects, and the enforcement of access rules is handled by the smart contract deployed on the blockchain and specifically in Sui Network. This architectural choice is what enables NexusChat to offer its users enhanced security and portability without dependence on a central authority.

2 NexusChat

2.1 Introduction to NexusChat

Imagine engaging with a sophisticated AI , like Chat-GPT, for insightful conversations, brainstorming, or capturing fleeting ideas to create your own decentralized application (Mao et al., 2023), all while processing an unwavering assurance: every word exchanged and every note taken is not only secure but entirely under your direct control, shielded from third-party access or unintended use. This vision is empowered, private digital interaction is the force and the core feature behind NexusChat. NexusChat is a Decentralized Application that seamlessly integrates the advanced conversational capabilities of AI with an innovative, user-centric data storage built upon the Sui Blockchain Network.

In the modern digital world, users of conventional chatbot and note-taking applications frequently encounter a compromise. While benefiting from powerful AI assistance and the convenience of cloud-based data storage, their sensitive personal data is typically entrusted to centralized servers. This prevalent model, however, exposes users to numerous risks and limitations. Concerns range from the potential for service providers to analyze, monetize or share private data without explicit user consent to the vulnerability of centralized databases as prime targets for security breaches, potentially exposing personal and sensitive data. Furthermore, users often lack true ownership and control over their information. They are granted access by a service provider who can alter terms or restrict access. This often results in data silos, making portability impossible.

NexusChat was conceived and developed to directly confront these challenges. It aims to shift the balance of power back to the user by fundamentally rethinking how interaction data with AI and personal notes are managed and stored. The primary objectives of the NexusChat project are to empower Users with data sovereignty and provide them verifiable ownership and absolute control over their conversational data and notes, and this is achieved by storing data as individual, user-owned objects on-chain, directly linked and managed via their crypto-wallet. Ensure granular user control and consent by implementing a system where no data is persisted without the user's explicit, per-interaction authorization via their wallet, granting them full control over what is saved. NexusChat promotes true data portability; this design system is developed to create portable user data, rather than being locked into a proprietary platform, fostering greater freedom and control. Finally, NexusChat leverages the inherent security features of blockchain technology, robust cryptography, and a wallet-based authentication system, offering a significantly more secure environment for the user.

2.2 Core Features and Functionalities

2.2.1 Wallet-Based Authentication & Accountless Access

A fundamental aspect of NexusChat's user-centric architecture is its approach to authentication and access. Departing from traditional applications that require users to create and manage separate accounts with usernames and passwords, NexusChat offers a streamlined and inherently secure login mechanism:

- **Accountless Operation:** Users do not need to register for a new account specific to NexusChat. This eliminates the burden of remembering yet another set of credentials and reduces the attack surface associated with application-specific account databases.
- **Direct Wallet Integration with Sui wallet:** Authentication is handled entirely through the user's existing Sui-compatible wallet. This integration is leveraging the official Sui's Software Development Kit (SDK), which provides functionalities for the NexusChat application to securely interface with these wallets. NexusChat supports a range of popular Sui wallets, such as the official Sui wallet, Suite, Slush Wallet, and potentially any other wallet that adheres to the Sui network. This includes wallets that themselves might offer social login integrations (e.g. Gmail) for user convenience in managing their wallet access.
- **User-Authorized Connection:** To access NexusChat, the user simply initiates a "login" or "Connect" action from the application's interface. This triggers a prompt directly within the user's chosen wallet, requesting authorization to connect to the NexusChat application. Only upon the user's explicit approval is the connection established for the session.
- **Session-Based Access:** The authorization granted is typically for the current session; logging out or periods of inactivity will sever the connection, requiring re-authorization for subsequent access.

The benefits of this wallet-based approach are manifold. It significantly enhances user security by leveraging the cryptographic blockchain wallets, effectively making the wallet the single point of secure access. It also improves user convenience by allowing them to use a familiar and trusted method for authentication across numerous applications, including NexusChat.

2.2.2 AI Interaction with Data Control (ChatGPT Integration)

NexusChat integrates advanced conversational AI capabilities, by leveraging powerful Large Language Models, prominently featuring OpenAI's ChatGPT, including

state-of-the-art models such as GPT-4. The architecture is intentionally designed to be adaptable, allowing for the integration of future leading AI models as they emerge, ensuring users can consistently benefit from the latest technologies. What fundamentally sets NexusChat apart is its core design philosophy: empowering users to harness these sophisticated AI capabilities without ever compromising their data privacy or relinquishing ownership. This is achieved by strategically utilizing the distinct advantages of the Sui Blockchain to decentralize and secure every piece of user-generated and AI-generated information as individually owned digital assets.

The flow of the interaction with the AI and the subsequent data persistence onto the Sui Blockchain is meticulously structured to ensure granular user control and explicit consent at each step:

- **User Input and AI Response:**

- The user writes a query and submits it through the NexusChat interface. This input is securely relayed to the integrated AI model on the backend server.
- The AI model processes the input and generates a corresponding response, which is then returned to the NexusChat application and displayed to the user.

- **Saving the User's Message (User-Authorized Transaction):**

- Immediately after the AI's response is received, the process for potentially storing the user's preceding message begins.
- NexusChat will prompt the user, asking through the crypto-wallet if they wish to save their message to the Blockchain. The wallet interface will display the details of the transaction, including information about the new Sui Object that will be created to represent their message.
- Only upon the user's explicit approval of this transaction via their wallet is the user's message encrypted server-side and minted as a distinct, individually-owned Sui Object, with ownership directly tied to their wallet address.

- **Saving the AI's Response (User-Authorized Transaction):**

- Following the successful (or failed) saving of the user's message, a similar procedure is initiated for the AI's generated response.

- NexusChat will again prompt the user, asking if they wish to save the AI's response to the Sui Blockchain and the user's wallet once more invoked, presenting the details for the creation of another Sui Object (this time representing the AI's response) and the associated transaction.
- Upon explicit user approval, the AI's response is encrypted server-side and minted as a separate, individually-owned Sui Object under the user's control.

- **AI Contextual Learning and Data Access Limitations:**

- During an active session, the AI model learns and adapts “through the dialogues,” meaning it maintains a temporary, session-specific contextual understanding based on the immediate, unencrypted conversational flow. This allows for coherent and relevant follow-up interactions.
- Crucially, the AI never has full access to the user's historical data stored as encrypted Sui Objects on the blockchain. Its “learning” is confined to the transient context of the current active “session” only.
- When the user logs out, the “session” is terminated. This action is designed to sever the AI's contextual link to that session, ensuring that any temporary understanding it had is cleared and not persistently retained, thereby upholding the strict privacy boundaries of NexusChat.

This granular, dual-authorization approach enables saving both user and AI messages, while ensuring maximum transparency and control. Each piece of information becomes a distinct, user-owned asset, reinforcing NexusChat's commitment to data sovereignty while allowing users to leverage the power of cutting-edge AI models fully at its fullest.

2.2.3 Decentralized & User-Owned Chat History

A foundation of NexusChat's privacy design is its innovative approach to managing and storing conversational records. The Chat History in NexusChat is not just a common log file or a database entry on a traditional server. Instead, it represents a collection of individual, encrypted data objects, each corresponding to a message, all securely residing on the Sui Network and directly owned by the user.

The type and structure of this collection are explicitly defined by a custom **struct** type, named **Chat**, which is an integral part of the NexusChat smart contract. This struct essentially serves as the primary container for a user's entire saved conversation history within NexusChat. The initialization of this container is handled during a user's initial interaction with the application. Upon the user's first login via the crypto-wallet, NexusChat's server queries the Sui Network to detect the presence of this object within the user's wallet assets. If no such object is found, indicating a first-time user, the server triggers a function within the NexusChat smart contract. This function mints a new instance of the Chat and transfers its ownership directly to the user's wallet. Once this foundational Chat Object resides in the user's assets, it becomes the designated structure for organizing and linking all subsequent individual message objects that the user chooses to save from their AI interactions. This mechanism ensures that from their first engagement, users are provided with their own decentralized, sovereign container for their chat data, reinforcing the principles of data ownership and control from the outset.

2.2.4 Decentralized Note System

Extending to its principles of user sovereignty and the decentralization of the data storage, NexusChat incorporates a sophisticated Decentralized Note System. This system mirrors the robust architecture of its Chat history, allowing users to capture, organize, and securely store valuable messages from the AI interaction.

The core functionality allows users, after the interaction with the AI, to select any message they deem important and save it as a "Note". To facilitate optimal organization and "best management", users can create personalized "Folders" within their note system. These folders, much like the notes themselves, are also represented as decentralized objects under the user's control. A user can then choose to save specific messages or newly created notes into one or more of these designated folders.

The mechanics operate with the same logic as the Decentralized Chat history. The NexusChat smart contract defines a primary struct type called Notepad, which serves as the root container for a user's entire note system. Upon a user's first interaction with the application, if a Notepad asset is not already present in their wallet, one is minted by the smart contract and transferred to their ownership. Organizing notes involves creating Folder Objects; the on-chain creation and saving of each Folder requires explicit user approval via a wallet transaction, and these Folder Objects are then included within the user's Notepad asset. Similarly, when a user chooses to save a message as a Note within a specific Folder, this action also necessitates a separate user-approved transaction to mint and store the encrypted Note Object. This hierarchical structure-with Note objects residing in Folder Objects, which are themselves contained within the main Notepad asset, ensures that every component of the note system is explicitly authorized, encrypted, and directly owned by the user.

2.2.5 Data Encryption

While blockchain technology inherently provides cryptographic security for transactions and data integrity through mechanisms such as digital signatures, which are the tool of the user authorization, NexusChat implements an additional layer of application-level encryption to further protect the confidentiality of user's data. This critical encryption and decryption process is handled by NexusChat's backend server before any user-generated content, such as Chat messages or Notes, is transmitted to the smart contract for on-chain storage, and after it is retrieved.

3 Conceptual Architecture & System Overview

3.1 Introduction

The NexusChat application is engineered as a multi-tier system, precisely designed to deliver its core promise of AI communication with user-centric data sovereignty. This architecture integrates a modern frontend user interface, a well-structured backend service layer, and a decentralized blockchain foundation. Each component plays a distinct role in ensuring functionality, security, and the unique data ownership model that defines NexusChat as the new Modern Web3 application (Liu et al., 2023).

This chapter aims to provide an overview of the foundational structure of NexusChat. It will begin by explaining the development methodology and the core philosophy that guided its design and implementation. Following this, a high-level overview of the system architecture will be presented, illustrating the primary components and their fundamental interactions. Finally, the chapter will go through the specific technology stack and detail the core components that compose each layer of the application, from the user interface client application to the server-side logic and the underlying decentralized blockchain infrastructure. Understanding this architecture is crucial for appreciating how NexusChat achieves its objectives of privacy and security, and user empowerment in the world of AI-driven communication.

3.2 Development Methodology and Philosophy

The development of NexusChat was guided by a philosophy centered on innovation, community bridging, and leveraging robust, scalable technologies to enhance the Web3 ecosystem. A key decision in its architecture was the strategic selection of the .NET Web API Core for its backend services (Building Web APIs with ASP.NET Core, 2023). While many decentralized applications mainly utilize Javascript-centric backend frameworks, often paired with Next.js (Vercel, 2024) or similar frontend solutions for which Sui already provides comprehensive SKDs, NexusChat intentionally charts a different course.

The choice of .NET stems from a conviction in its various capabilities, extensive features and the potential for building highly performant and scalable server-side applications. This decision was also driven by a desire to explore new architectural paradigms within Web3 and to showcase the viability of .NET in this domain. A core tenet of NexusChat's development philosophy is to bridge established, mature technology stacks with the nascent and rapidly evolving Web3 space. The vision is to demonstrate the potential of .NET within decentralized application architectures, thereby encouraging the entry of the extensive .NET developer

community into Web3 exploration and innovation, an area currently blooming with opportunities.

On the frontend, Reactjs was chosen for its versatility, developer-friendly features, and its ability to create dynamic and responsive user interfaces (Meta, 2024). This flexibility was paramount for crafting an intuitive user experience. Crucially, the official Sui SDK is instrumental in managing the seamless wallet-based authentication process, prompting users via their browser-based Sui wallets to authorize login and subsequent on-chain transactions. Complementing this, TailwindCss was employed as the utility CSS Framework (Tailwind Labs, 2024), enabling the rapid development of a modern, aesthetically pleasing user interface.

Finally, the on-chain logic was implemented using Sui Move, selected for its strong emphasis on safety, its resource-oriented programming model ideal for managing digital assets like user data objects and its native interaction within the Sui ecosystem. This methodological approach, combining the robustness of .NET on the backend server, the flexibility of Reactjs for the frontend and the specific Web3 integrations provided by the Sui SDK, and the secure smart contract capabilities of Sui Move, aims to deliver a unique, secure and user friendly dApp, while also contributing to the diversification of technological approaches within the Web3 landscape.

3.3 High Level System Architecture

NexusChat is architected as a distributed system composed of three primary tiers: a dynamic client, a robust backend service layer and a decentralized blockchain layer. Each tier leverages specific tech chosen for their strengths in delivering the application's core functionalities, particularly concerning user interaction, data processing, security and decentralized ownership.

3.3.1 Frontend (Client-Side): Reactjs (Typescript) and Libraries

The user interface and client-side logic of NexusChat are built using Reactjs, a widely-adopted library for building dynamic and interactive user interfaces. The frontend is developed using Typescript, a statically typed superset of Javascript, which enhances code quality, maintainability and developer productivity through type safety and improved tooling.

React's component-based architecture, combined with Typescript, allows for the creation of robust, modular and reusable UI elements, facilitating a scalable frontend application.

- **@mysten/dapp-kit:** This is a specialized toolkit provided by mysten Labs for the Sui ecosystem, designed for Typescript environments (Mysten Labs, 2024). It offers pre-build React and Hooks specifically tailored to simplify the integration of Sui wallet functionalities into dApps. In NexusChat, it is crucial for facilitating the user's wallet connection process.
- **@mysten/sui:** This is the core Typescript SDK for interacting with the Sui Blockchain (Mysten Labs, 2024). It provides strongly-typed functions and interfaces for constructing, signing and submitting transactions.
- **React Context API and Providers:** React's built-in Context API is utilized for state management across the application, managed with Typescript for type safety and shared state. NexusChat implements custom Providers and Contexts to manage global state related to user authentication, and to pre-load data such as chat messages, folders and notes. This ensures a smoother user experience by minimizing loading times when navigating to different sections like the chat or notepad pages. Client-side routing within the single-page application is also managed using React-compatible libraries such as React Router.

```
export const UserProvider: React.FC<React.PropsWithChildren<{}>> & ({ children }) => {  
  
  const [user, setUser] = useState<User | null>(null);  
  const [token, setToken] = useState<string | null>(localStorage.getItem("token"));  
  const [chatID, setChatID] = useState<string | null>(null);  
  const [messages, setMessages] = useState<Message[]>([]);  
  const [notepad, setNotepad] = useState<Notepad | null>(null);  
  const run = useRef(true);  
  
  useEffect(() => {  
    const savedToken = localStorage.getItem("token");  
    if (savedToken && run.current) {  
      run.current = false;  
      setToken(savedToken);  
      handleTokenValidation(savedToken);  
    }  
  }, []);  
  
  const handleTokenValidation = async (token: string) => {  
    try {  
      const resultUser = await validateToken(token, "invalid");  
      if (resultUser) {  
        setUser(resultUser);  
        if (!notepad?.id) {  
          getNotes(resultUser.walletAddress, token);  
        }  
      } else {  
        logout();  
      }  
    } catch (error) {  
      logout();  
    }  
  }  
};
```

Image 3.1: User Provider Code

- **Custom Services:** Client-side services, written in Typescript, are developed to handle specific logic. For a service that manages session persistence by

sending the locally stored JWT to the backend for re-authentication; if valid, it re-initiates the wallet connection, restoring the user's session. Other services are responsible for the transaction signing flow with the user's wallet and then relaying the signed transaction block to the backend for final submission to the blockchain.

```
export const signTransaction = async (txBytesBase64: any) => {
  const keypair = Ed25519Keypair.fromSecretKey(PRIVATE_KEY);
  const txBytes = fromB64(txBytesBase64);
  return await keypair.signTransaction(txBytes);
};

export const executeTransaction = async (block: any, token: any) => {
  const response = await fetch("https://localhost:7261/api/Transaction/execute_Transaction", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: token ? `Bearer ${token}` : "",
    },
    body: JSON.stringify(block),
  });
  if (!response.ok) throw new Error("Transaction failed");
};

export const executeCreateUser = async (transactionUserData: any, walletAddress: any) => {
  try {
    const response = await fetch("https://localhost:7261/api/Transaction/execute_createUser", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ TransactionUserData: transactionUserData, WalletAddress: walletAddress }),
    });
    if (!response.ok) {
      throw new Error("Failed to log in");
    }
    return response.json();
  } catch (error) {
    console.error("Login error:", error);
    throw error;
  }
}
```

Image 3.2: Transaction Services Code

- **TailwindCss:** For styling and UI design, Tailwind CSS is utilized. It's a utility-first CSS framework that allows for rapid development of custom user interfaces by composing low-level utility classes directly in the HTML, providing flexibility and maintainability.

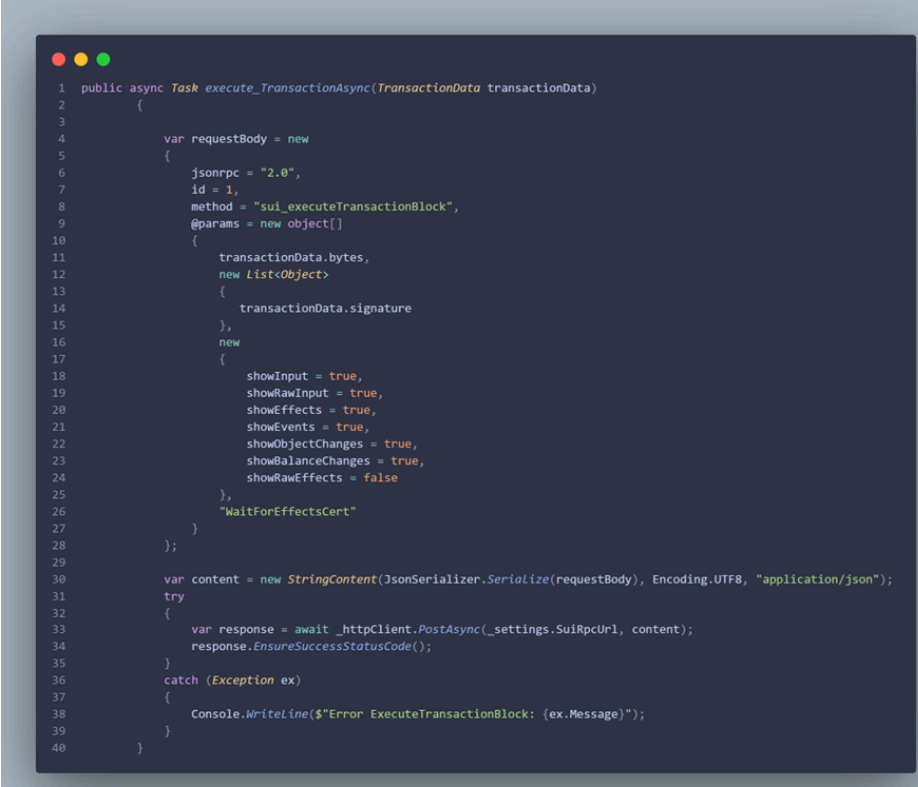
```
1 <section className='relative mb-20'>
2   <img className='w-full h-[400px] lg:h-[500px] object-cover' alt="Image that says be Creative" src={image} />
3   <div className="absolute backdrop-blur-[5px] h-full inset-0 bg-dark-900/60"></div>
4   <div className='container px-4 text-center flex flex-col items-center justify-center gap-4 absolute inset-0 mx-auto'>
5     <h1 className='text-3xl sm:text-4xl lg:text-5xl font-bold text-dark-200'>
6       Notepad
7     </h1>
8     <p className="text-lg sm:text-xl lg:text-2xl font-semibold text-dark-300 lg:w-[60%]">
9       Save your messages as notes, organize them in folders, and access everything anytime
10    </p>
11  </div>
12 </section>
```

Image 3.3: TailwindCss Example

3.3.2 Backend (Server-Side): .NET (ASP.NET Web API Core)

The backend services for NexusChat are developed using the .NET Framework, specifically ASP.NET Web API Core for building web APIs. .NET is a robust, cross-platform, open-source framework known for its performance, scalability and extensive libraries, making it suitable for complex server-side logic.

- **API Endpoints and Controller:** The backend exposes a set of RESTful API endpoints, implemented using controllers. These endpoints serve as the communication interface between the Frontend client and the server-side logic. They handle requests for AI Interactions, data persistence, and user session management.
- **Sui JSON-RPC Interaction:** To Interact with the NexusChat smart contract deployed on the Sui Network, the .NET backend acts as a client to the Sui JSON-RPC APIs. It constructs and sends RPC calls to invoke specific functions within the smart contract (e.g., to mint a new message or retrieve existing ones). The backend processes the responses from the blockchain, formatting data as needed before sending it to the frontend.



```
1 public async Task execute_TransactionAsync(TransactionData transactionData)
2 {
3
4     var requestBody = new
5     {
6         jsonrpc = "2.0",
7         id = 1,
8         method = "sui_executeTransactionBlock",
9         @params = new object[]
10         {
11             transactionData.bytes,
12             new List<Object>
13             {
14                 transactionData.signature
15             },
16             new
17             {
18                 showInput = true,
19                 showRawInput = true,
20                 showEffects = true,
21                 showEvents = true,
22                 showObjectChanges = true,
23                 showBalanceChanges = true,
24                 showRawEffects = false
25             },
26             "WaitForEffectsCert"
27         }
28     };
29
30     var content = new StringContent(JsonSerializer.Serialize(requestBody), Encoding.UTF8, "application/json");
31     try
32     {
33         var response = await _httpClient.PostAsync(_settings.SuiRpcUrl, content);
34         response.EnsureSuccessStatusCode();
35     }
36     catch (Exception ex)
37     {
38         Console.WriteLine($"Error ExecuteTransactionBlock: {ex.Message}");
39     }
40 }
```

Image 3.4: JSON-RPC Interaction Example

- **AES Cryptography:** A critical function of the backend is data encryption and decryption. Before user data (messages, notes) is sent to the smart contract for

on-chain storage, it is encrypted server-side using **Advanced Encryption Standard** (AES) (National Institute of Standards and Technology, 2001). Conversely, when data is fetched from the blockchain, the backend decrypts it before relaying it to the client. This ensures that data stored on-chain is confidential and only decipherable by authorized parties with access to the decryption keys (implicitly managed via the user's interaction flow and session).

A screenshot of a code editor with a dark background and light-colored text. The code is a C# function named `Decrypt` that takes a `string cipherText` as input and returns a `string`. The function uses `Aes` for decryption. It creates an `Aes` object, sets its `Key` and `IV` from `CryptoConfig`, creates a `ICryptoTransform` decryptor, and then uses a `MemoryStream` and `CryptoStream` to decrypt the input. Finally, it uses a `StreamReader` to read the decrypted data and return it. The code is numbered from 1 to 20.

```
1 public static string Decrypt(string cipherText)
2 {
3     using (Aes aesAlg = Aes.Create())
4     {
5         aesAlg.Key = GetValidKey(CryptoConfig.Key, 32);
6         aesAlg.IV = GetValidKey(CryptoConfig.IV, 16);
7
8         ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
9         using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(cipherText)))
10        {
11            using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
12            {
13                using (StreamReader srDecrypt = new StreamReader(csDecrypt))
14                {
15                    return srDecrypt.ReadToEnd();
16                }
17            }
18        }
19    }
20 }
```

Image 3.5: AES Decryption Function

- **OpenAI API Integration:** The backend integrates with OpenAI's services by making API calls to models like ChatGPT. It relays user queries from the frontend to the AI model and then forwards the AI's responses back to the user, managing this interaction flow.
- **JWT Token Authentication:** For session management, the backend implements JSON Web Token (JWT) authentication (Jones et al., 2015). After a successful wallet-based login, the backend issues a JWT to the frontend. This token is then included in subsequent requests from the client. The server validates this token to authenticate the user for session-based operations, allowing users to refresh the page or return after a short period and have their session seamlessly restored without needing to reconnect their wallet immediately, provided the token is still valid.

3.3.3 Blockchain Layer: Sui Move Smart Contract

The decentralized data storage and ownership logic of NexusChat are implemented as a smart contract on the Sui Blockchain Network, written in Sui Move. Sui Move is a programming language derived from Move, specifically designed for safety and expressiveness in managing digital assets on the Sui Network. Its resource-oriented architecture ensures that assets cannot be accidentally duplicated or lost.

- **Smart Contract Module:** All on-chain logic for NexusChat is encapsulated within a single Sui Move module. This module defines the custom data types and functions that govern how user data is stored and managed.
- **Custom Struct Types:** The module defines several custom struct types to represent the core data entities for NexusChat:
 - **Message:** Represents an individual chat message.

A screenshot of a code editor showing the Sui Move code for the Message struct. The code is as follows:

```
1 struct Message has key, store {  
2     id: UID,  
3     message: vector<u8>,  
4     date: u64,  
5     sender: vector<u8>,  
6 }
```

Image 3.6: Message Struct Type

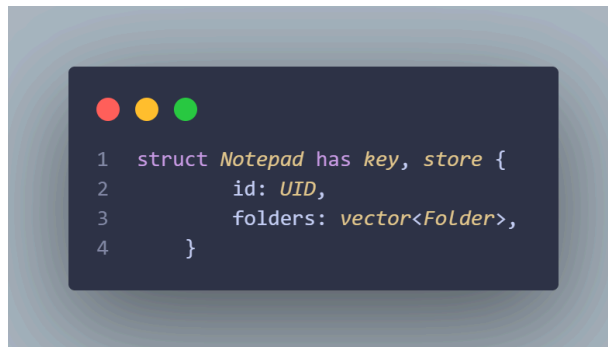
- **Chat:** Acts as a container for a user's entire collection of Message Objects.

A screenshot of a code editor showing the Sui Move code for the Chat struct. The code is as follows:

```
1 struct Chat has key, store {  
2     id: UID,  
3     messages: vector<Message>,  
4 }
```

Image 3.7: Chat Struct Type

- **Notepad:** Serves as the root container for a user's note-taking system.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following Rust code:

```
1 struct Notepad has key, store {  
2     id: UID,  
3     folders: vector<Folder>,  
4 }
```

Image 3.7: Chat Struct Type

- **Folder:** Represents a user-created folder within their Notepad for organizing notes.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following Rust code:

```
1 struct Folder has key, store {  
2     id: UID,  
3     name: vector<u8>,  
4     notes: vector<Note>,  
5 }
```

Image 3.8: Folder Struct Type

- **Note:** Represents an individual piece of information saved by the user.

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following Rust code:

```
1 struct Note has drop, store {  
2     id: vector<u8>,  
3     message: vector<u8>,  
4     date: u64,  
5     sender: vector<u8>,  
6 }
```

Image 3.9: Note Struct Type

- **On-Chain Functions:** This module includes public functions that can be called via transactions. These functions implement the core on-chain logic, such as:
 - Minting new instances of Message, Folder, Note, Chat or Notepad objects.
 - Transferring ownership of these newly created objects directly to the user's Sui wallet and stored as assets.



```
1 public entry fun add_message(  
2     chat: &mut Chat,  
3     message: vector<u8>,  
4     sender: vector<u8>,  
5     date: u64,  
6     ctx: &mut tx_context::TxContext,  
7 ) {  
8     let new_message = Message {  
9         id: new(ctx),  
10        message,  
11        date,  
12        sender,  
13    };  
14  
15    vector::push_back(&mut chat.messages, new_message);  
16 }
```

Image 3.10: On-Chain Function

This multi-tier architecture, with its carefully selected technologies, allows NexusChat to offer a seamless AI Interaction experience while pioneering a user-sovereign data management model on the Sui blockchain.

4 System Demonstration and User Workflow

4.1 Introduction

Following the detailed examination of NexusChat's conceptual architecture and technology stack in Chapter 3, this chapter provides a practical demonstration of the implemented application. The objective is to illustrate the end-to-end user experience, showcasing how the system's components work in concert to deliver a secure, decentralized, and user-sovereign AI interaction. This walkthrough will guide the reader through a series of core use case scenarios, from initial authentication to data creation. Each step will be accompanied by screenshots from the live application to provide clear, visual evidence of the described functionality. The following sections will cover the user's journey, beginning with wallet authentication, proceeding through AI interaction and data persistence, and concluding with the management of chat history and personal notes.

4.2 Wallet-Based Authentication and Session Initiation

The user's journey with NexusChat begins on the application's home page. As shown in Image 4.1, this initial landing area is designed to be welcoming and informative, providing the user with a concise overview of the application's core features. This page serves to orient the user before they commit to an interactive session.

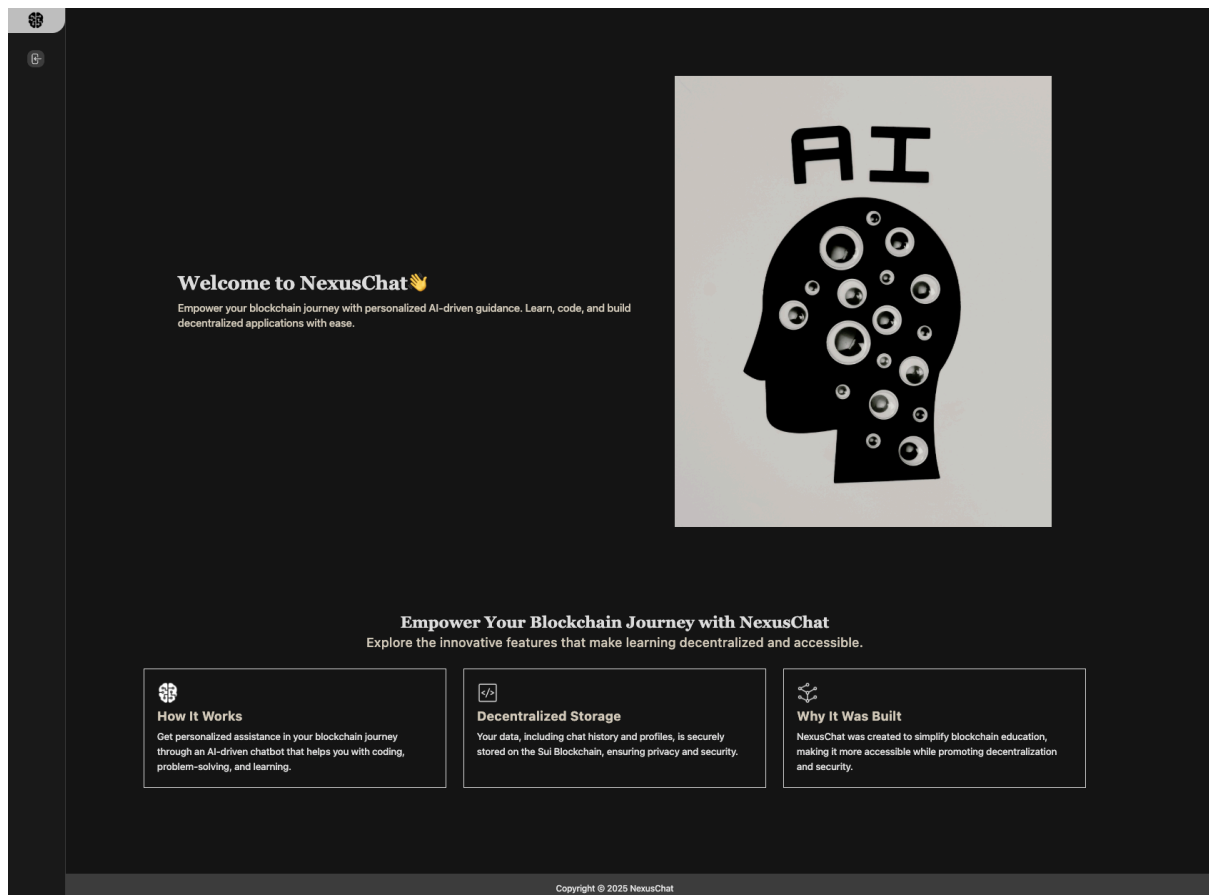


Image 4.1: Home Page

While the home page provides this introductory context, the primary interactive functions are consolidated within a dynamic side navigation menu. To proceed with authentication, the user hovers their cursor over the collapsed menu bar on the side of the screen. This action triggers the menu to expand, revealing the main navigation options. For a new or unauthenticated user, the key call to action is the "Connect" button (image 4.2):

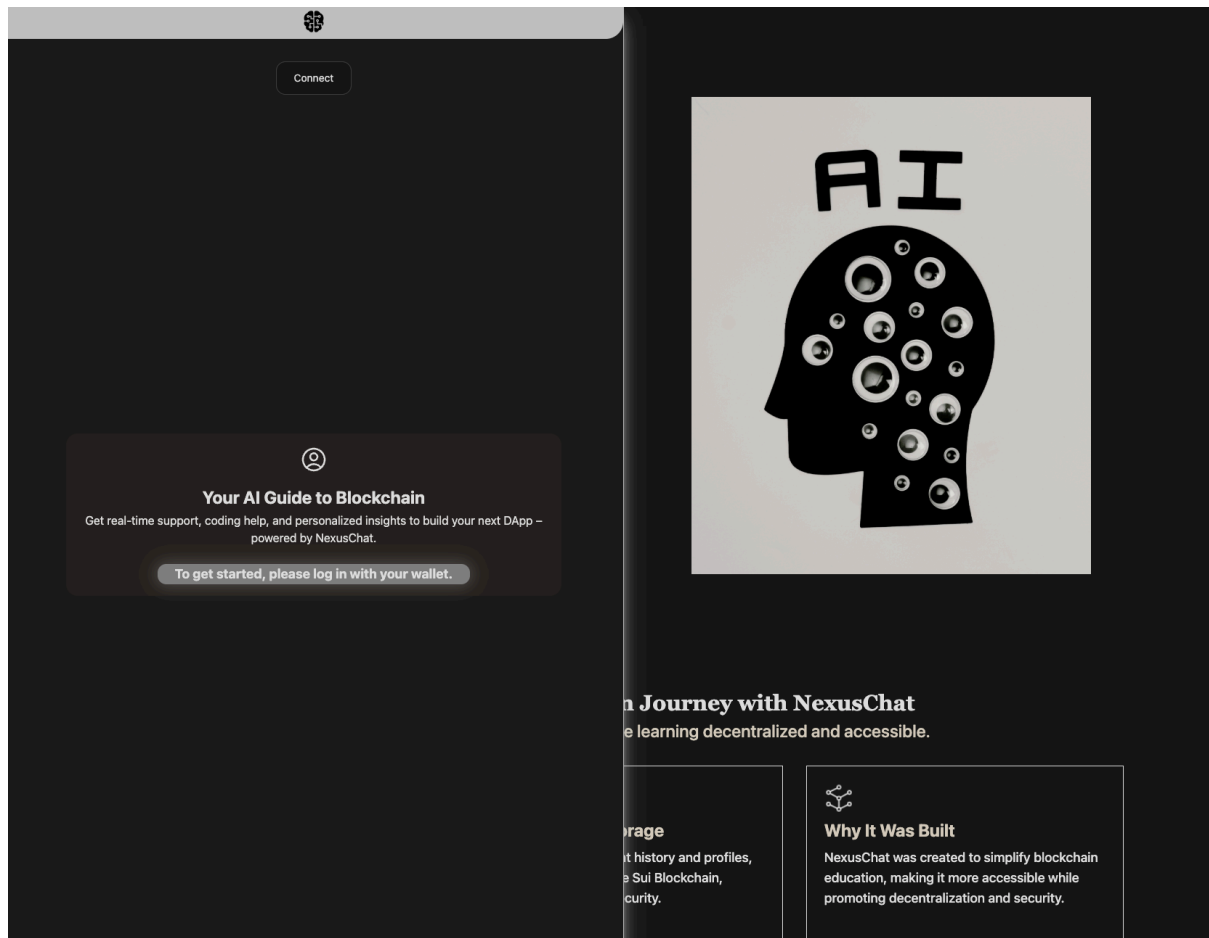


Image 4.2: Side Navigation Menu with the “Connect” Action Button

Clicking "Connect Wallet" does not immediately trigger a specific wallet's prompt. Instead, it first opens a modal window that presents the user with a list of all detected Sui-compatible wallet extensions installed in their browser. This wallet selection interface, facilitated by the @mysten/dapp-kit, is a key feature for ensuring broad compatibility and user choice, as shown in image 4.3:

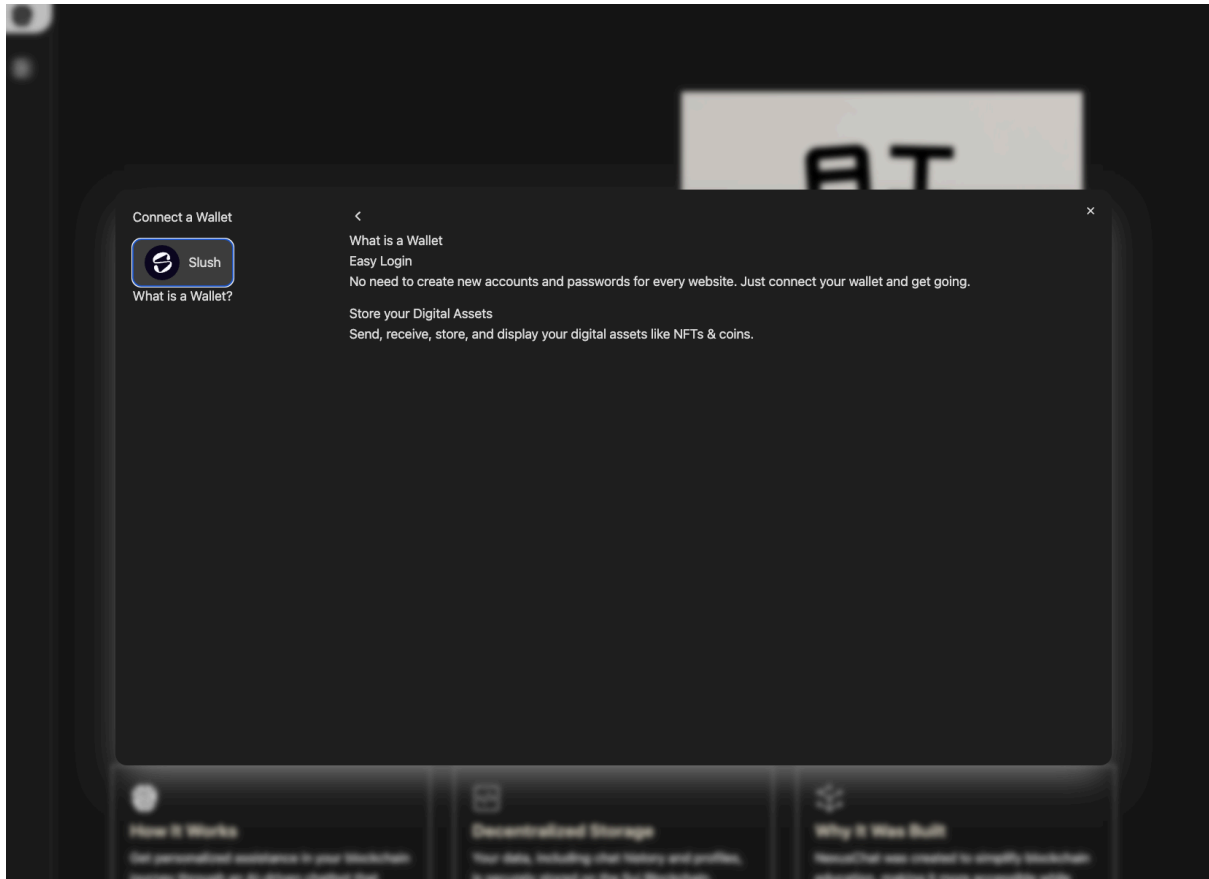


Image 4.3: list of all detected Sui-compatible wallet extensions

Once the user selects their preferred wallet from this list—for instance, the official 'Slush Wallet'—the application then proceeds to invoke the authentication prompt specific to that chosen wallet. As illustrated (image 4.4), this prompt is a secure interface generated by the wallet extension itself, asking for the user's explicit permission to connect to the NexusChat dApp.

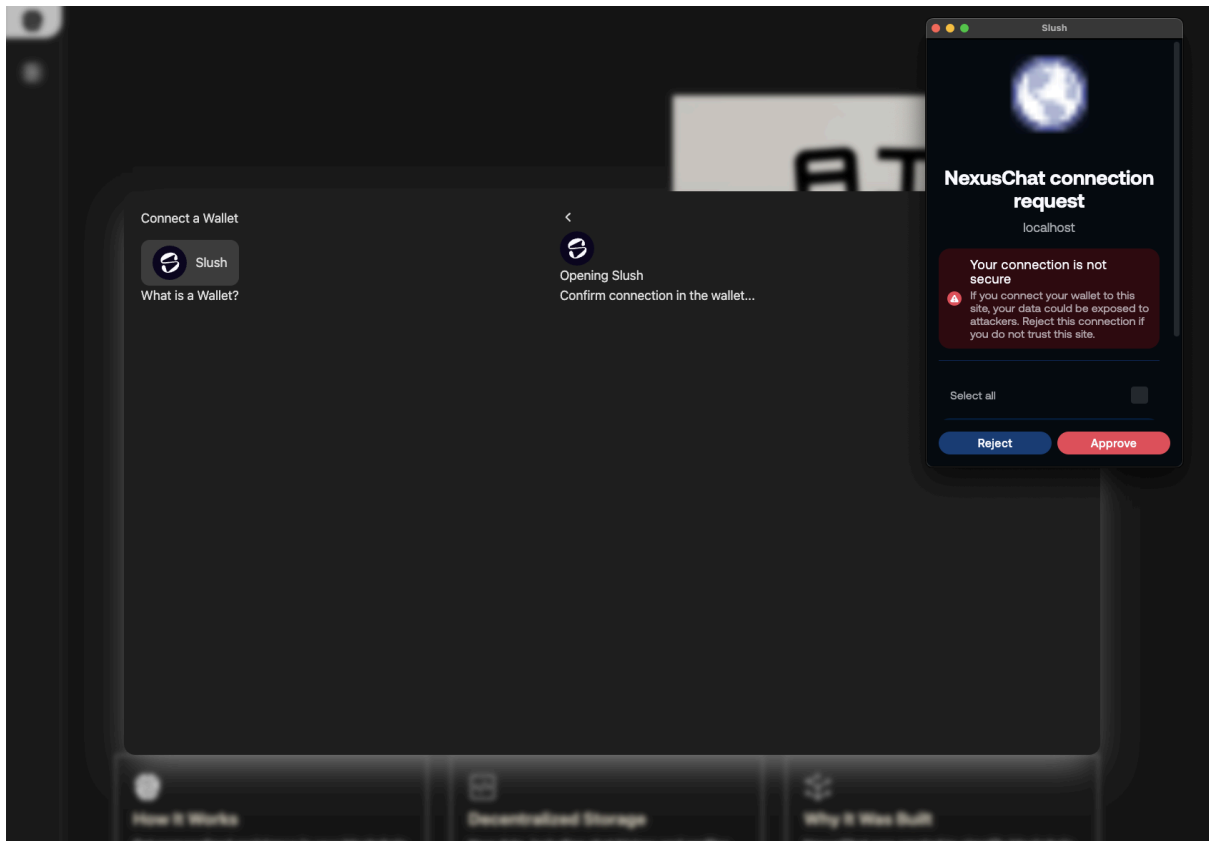


Image 4.4: Wallet Authorization for application access

Upon user approval within the wallet interface, a cryptographically signed confirmation is returned to the NexusChat backend. The server validates this signature and issues a JSON Web Token (JWT) to establish and maintain a secure session. The side menu then updates in real-time to reflect the authenticated state, replacing the "Connect" button with the "Disconnect" button and unlocking full access to the application's features.

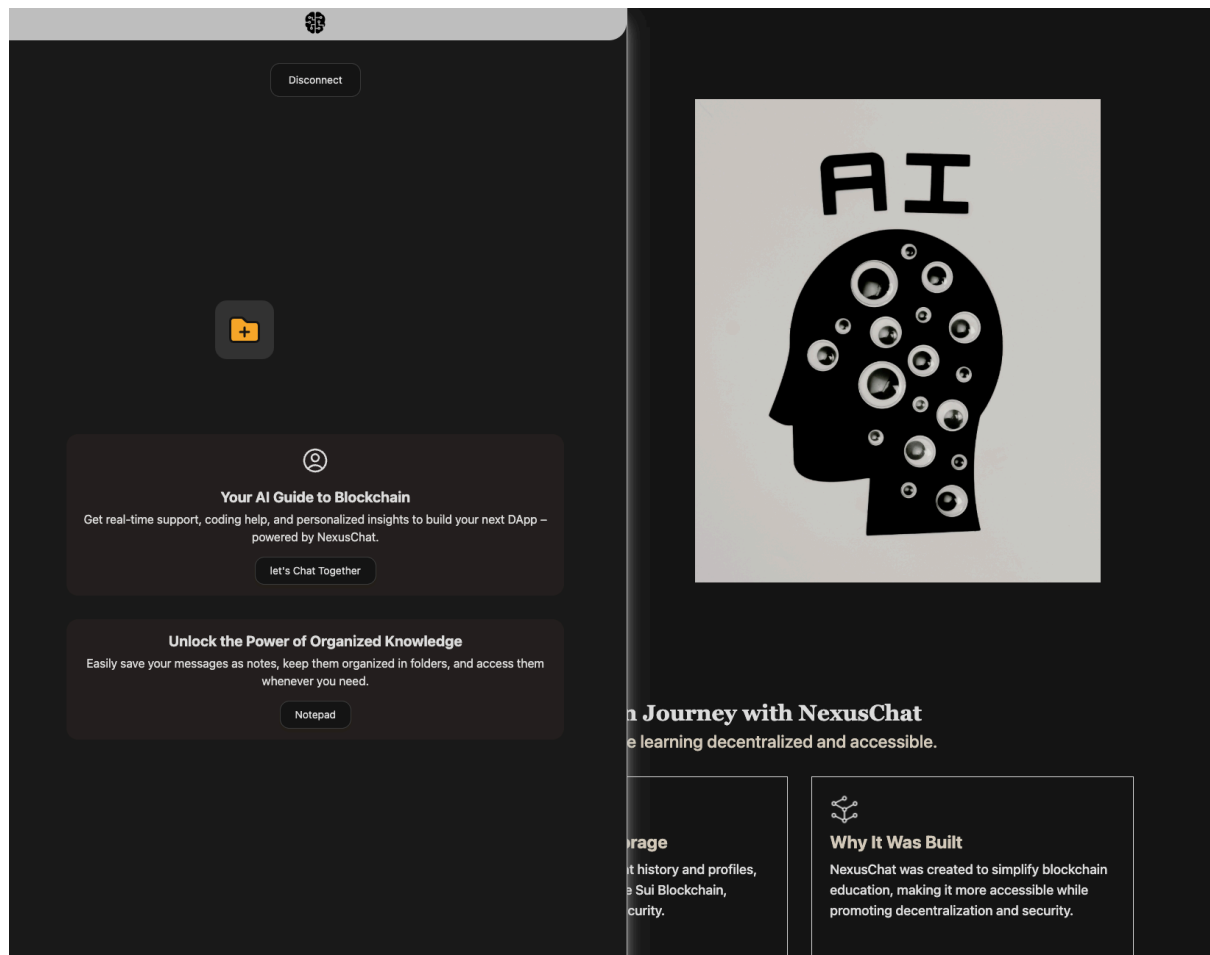


Image 4.5: Authenticated State with application Features

As illustrated in Image 4.5, the authenticated side menu contains several key components. At the top, a button that is shown as a folder allows the user to begin organizing their note-taking system. Below this, the interface dynamically displays a list of any folders the user has previously created, providing quick access to their saved information. Further down, a prominent banner with the text "Let's Chat Together" serves as the primary entry point for engaging with the AI. Finally, a link provides navigation to the main "Notepad" page, which offers a comprehensive view of the user's entire folder and note structure.

The user's first step to engage with the AI is to click the "Let's Chat Together" banner. This action navigates the user to the dedicated chat interface. As shown in Image 4.6, this page presents a clean, conversational layout, featuring a text input field at the bottom where the user can type their query and a button to submit it to the AI model. The interface is now ready to accept the user's first message, marking the beginning of the interactive AI dialogue.

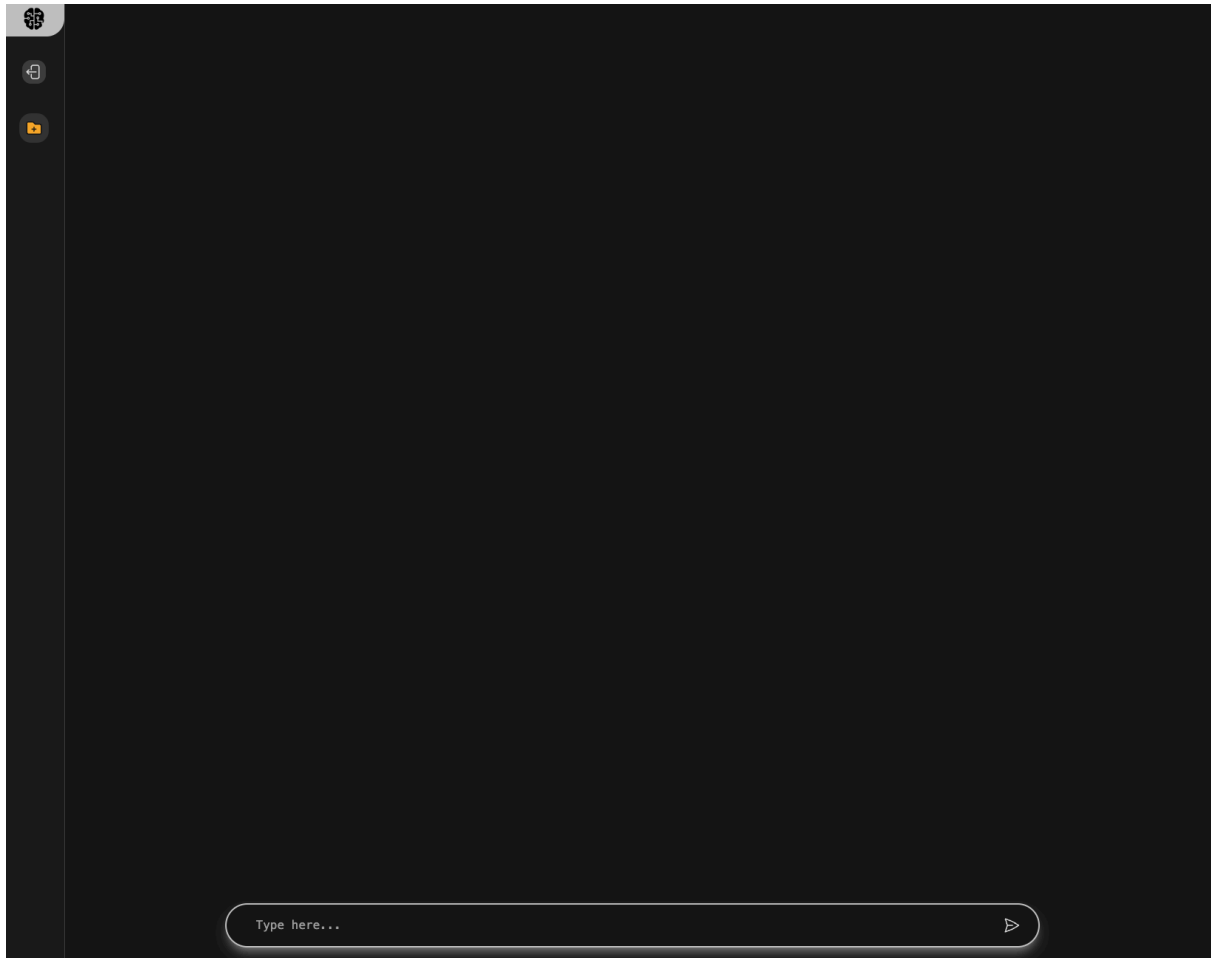


Image 4.6: Chat Page

4.3 Core AI Interaction and On-Chain Data Persistence

With the chat interface open, the user can begin their conversation with the AI. The process starts by typing a query into the text input field and submitting it. The application's backend relays this query to the integrated OpenAI model, which processes the input and generates a response. This response is then sent back and displayed in the chat interface, creating a complete conversational turn, as shown in Image 4.7.

Immediately following the AI's response, NexusChat initiates its unique, user-authorized data persistence workflow. This process is intentionally designed as a two-step sequence to provide the user with granular control over exactly what data is saved.

First, the application invokes the user's wallet to ask for permission to save the user's own message. As illustrated in Image 4.7, a wallet pop-up appears, presenting the details of a transaction that will mint the user's message as an encrypted, individually owned Sui Object on the blockchain.

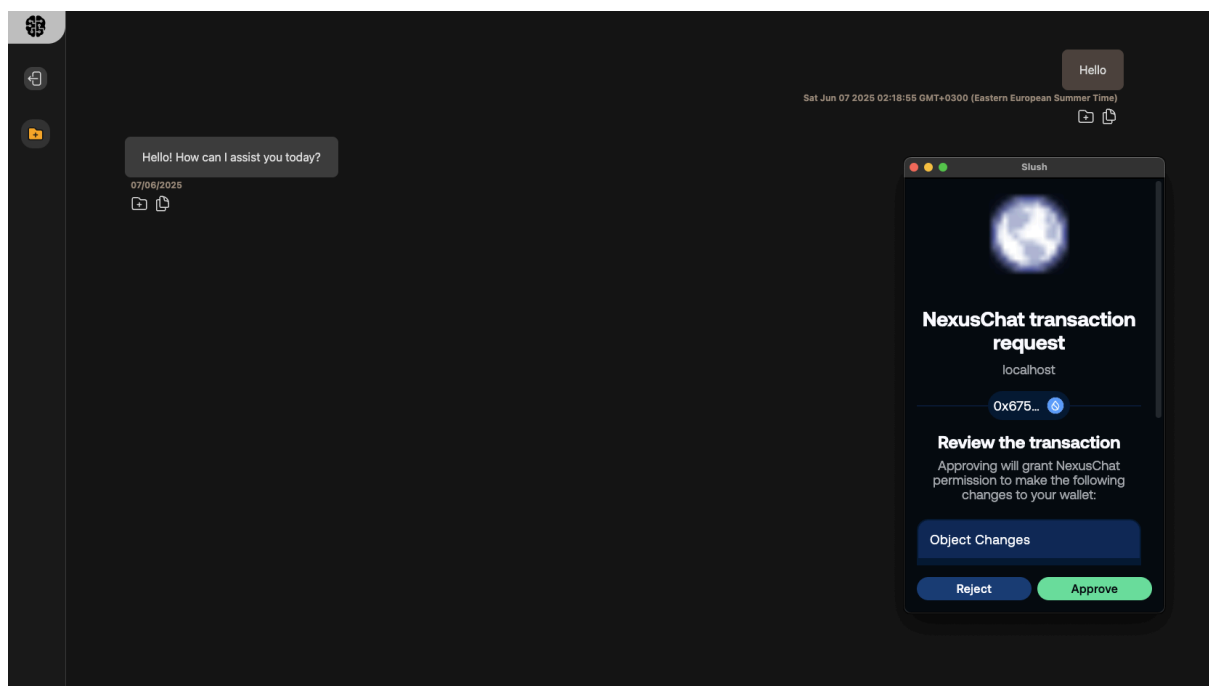


Image 4.7: AI Interaction and Wallet authorization

4.4 Archiving Information: The Decentralized Note System

Beyond the real-time persistence of the chat log, NexusChat provides users with a powerful system for archiving valuable information into a structured, decentralized notepad. This process is initiated directly from the chat interface. When a user identifies a particularly important message—either their own or the AI's—they can click the dedicated save icon located beneath the message bubble.

Clicking this icon launches the "Notepad" workflow, starting with a modal window that displays the user's existing note folders. For a first-time user, this list will be empty, presenting a clean slate with a primary call to action: a Create Folder icon, as shown in Image 4.7.

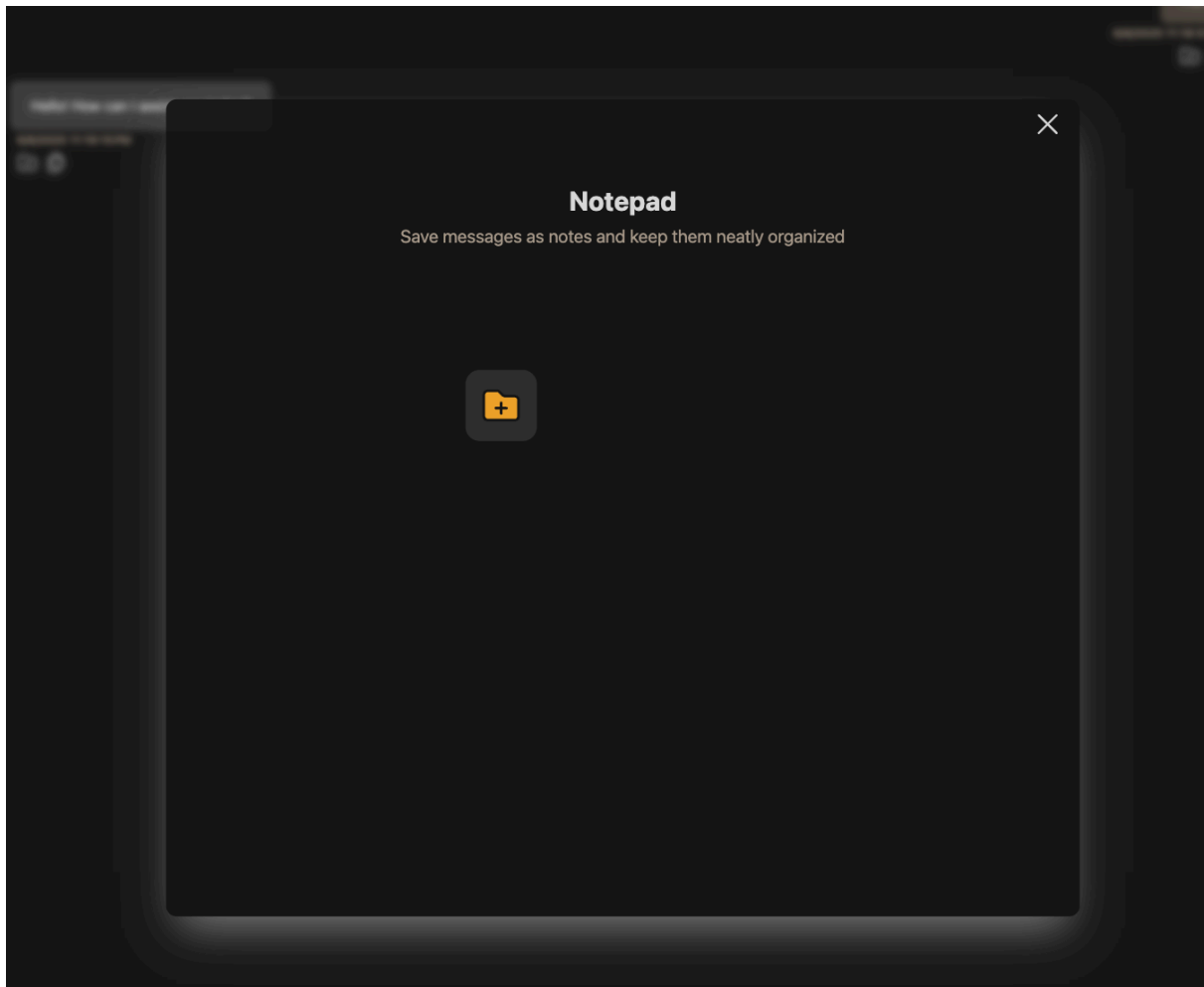


Image 4.8: Notepad Modal

To create a new organizational category, the user clicks the "Create Folder" button. This action opens a second, more focused modal that prompts the user to enter a name for

their new folder (Image 4.9).

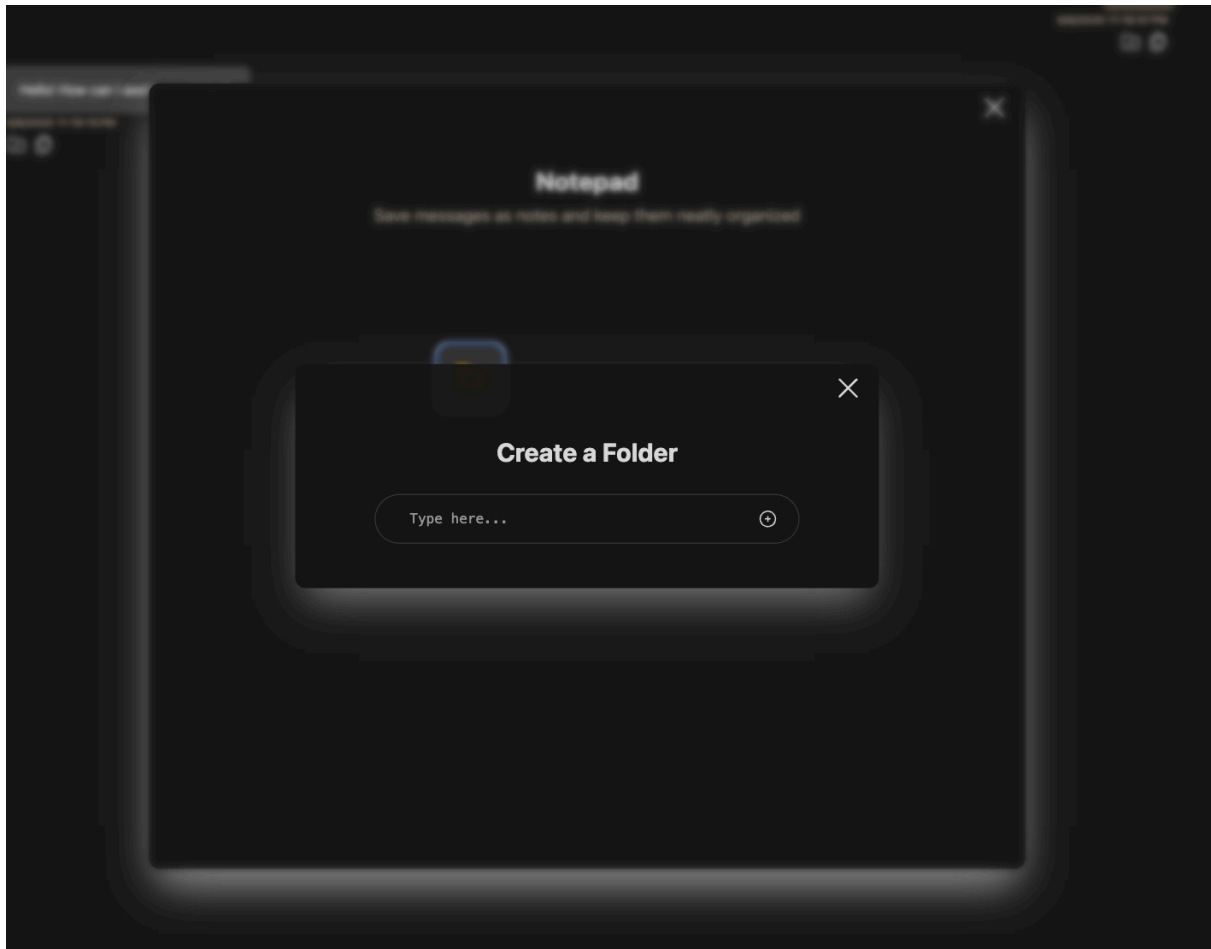


Image 4.9: Create a Folder

After the user types a name and confirms, the application initiates an on-chain transaction. This requires wallet approval to mint a new, user-owned Folder object on the Sui Network. Once the transaction is successfully processed, the initial "Notepad" modal dynamically updates to include the newly created folder as a selectable option, as illustrated in Image 4.10.

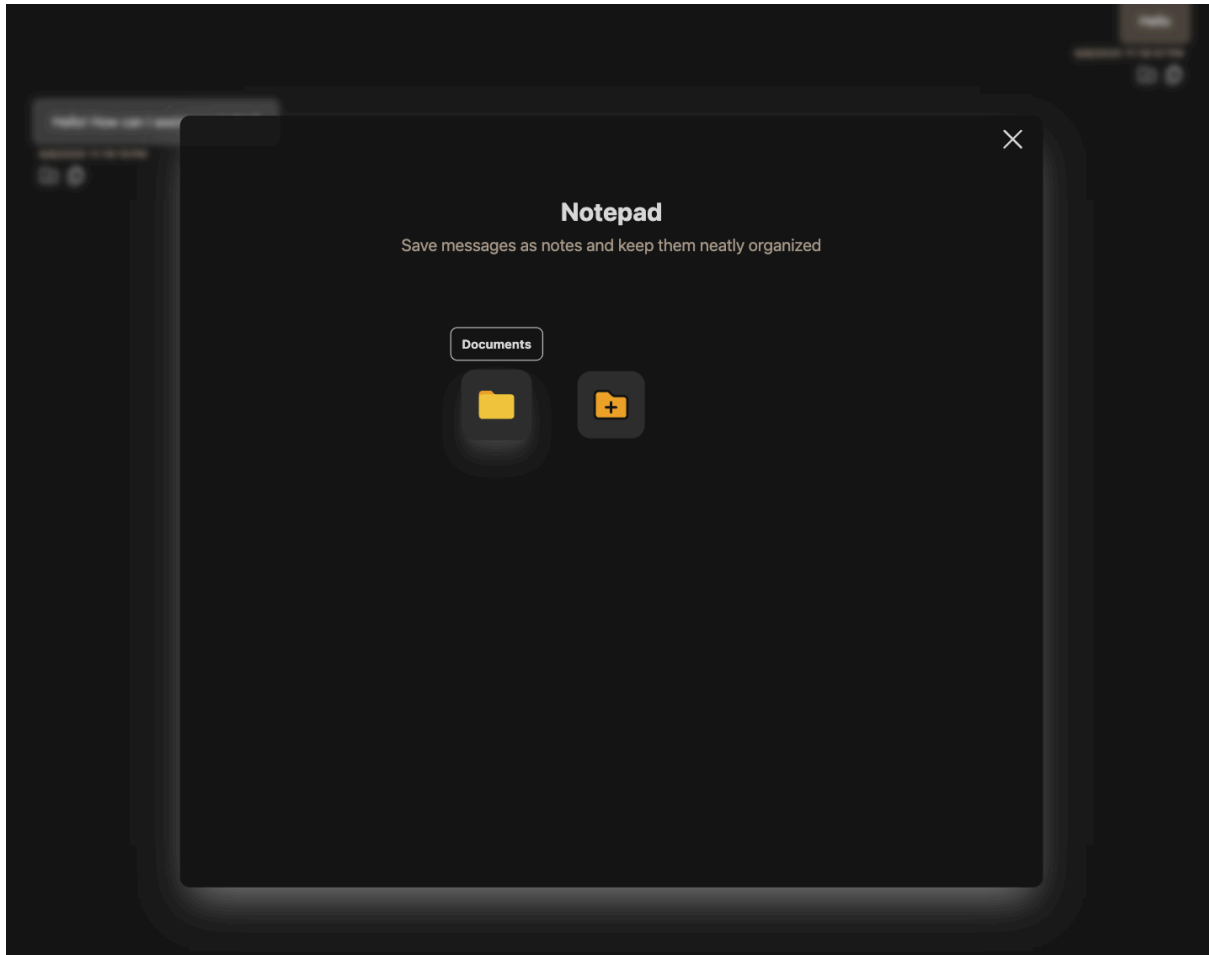


Image 4.10: Folder Creation

With the destination folder now available, the final step is to click on the folder's name. This action initiates the last transaction in the sequence, which again requires wallet authorization. This transaction mints the selected message as an encrypted Note object and programmatically links it to the chosen Folder object on the blockchain. Upon successful completion, the message is securely and permanently archived, and the user can continue their chat, confident that their valuable insight has been saved under their complete control.

Once a note is securely archived on the Sui Network, NexusChat provides the user with two distinct and convenient methods for accessing and viewing it. This flexibility allows for both quick, direct access and comprehensive organizational management.

The first and most immediate method is through the dynamic side navigation menu. As illustrated in Image 4.11, after a folder is created, it is automatically listed in the side menu. This provides an at-a-glance view of the user's organizational structure.

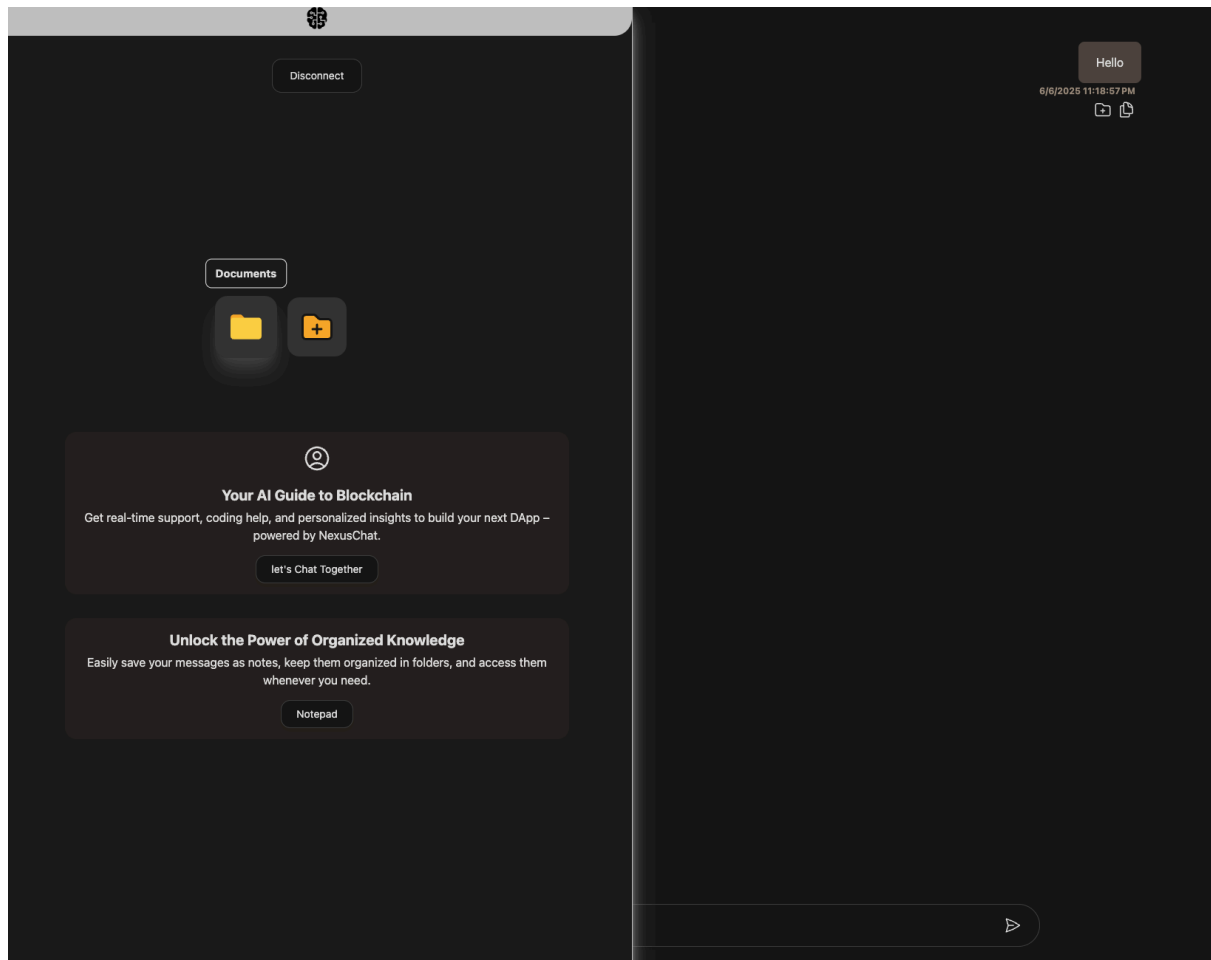


Image 4.11: Folders Display in Menu

The second method is designed for a more comprehensive overview of the entire note system. By clicking the "Notepad" link from the side menu, the user is navigated to a dedicated page that displays their full collection of folders in a more expansive layout, as shown in Image 4.12. This view is ideal for users who wish to manage a larger number of folders and notes.

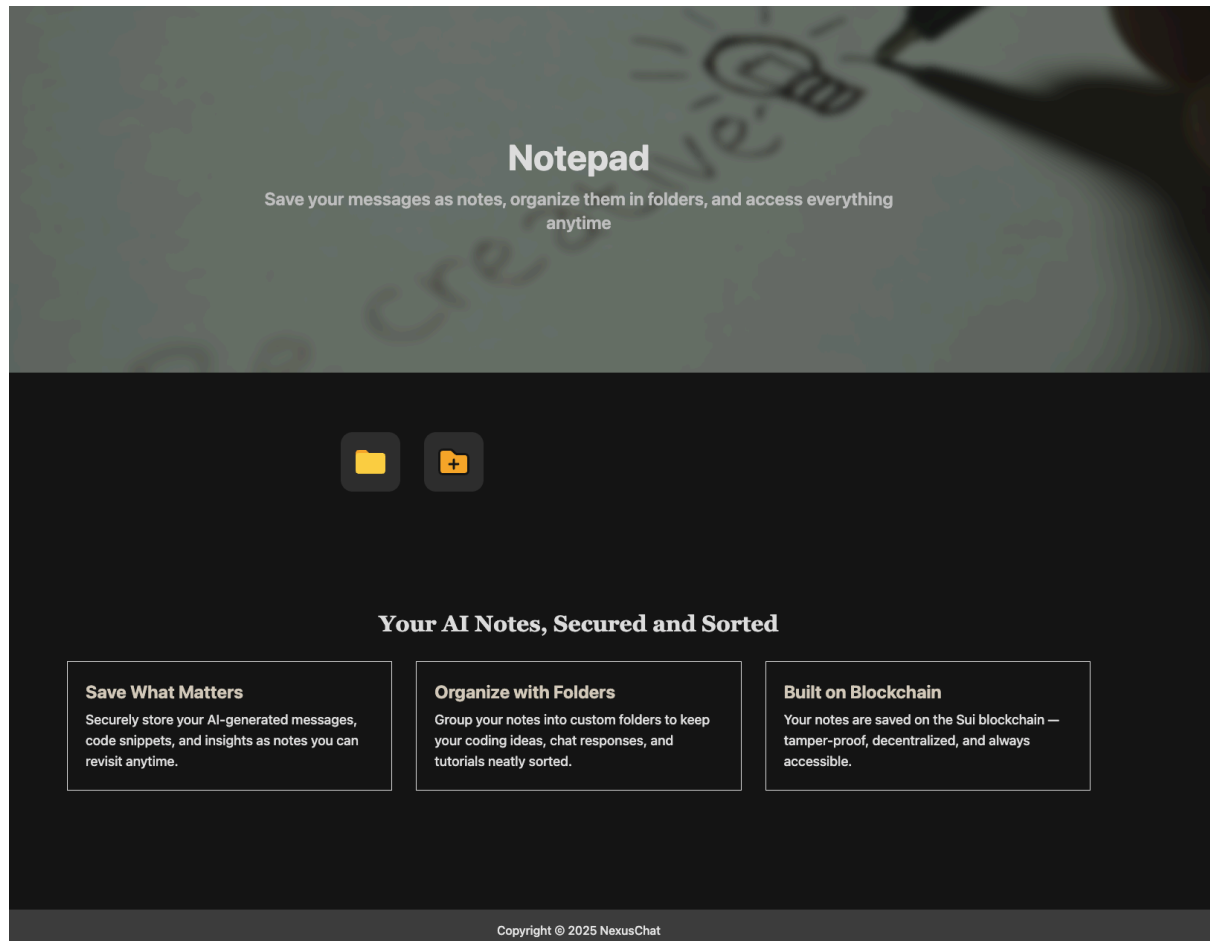


Image 4.12: Notepad Page with Folders

Regardless of which path the user chooses—clicking the folder directly from the side menu or from the main Notepad page—the result is the same. The action triggers a request to the backend, which fetches the encrypted Note object(s) associated with that Folder from the user's assets on the Sui blockchain. The server decrypts the data and presents it to the user in a clean, readable format within a modal window.

As depicted in Image 4.13, this modal displays the content of the saved note, making the archived information readily available. This seamless retrieval process completes the data lifecycle, demonstrating how users can not only save information with full sovereignty but also access it effortlessly from multiple points within the application.

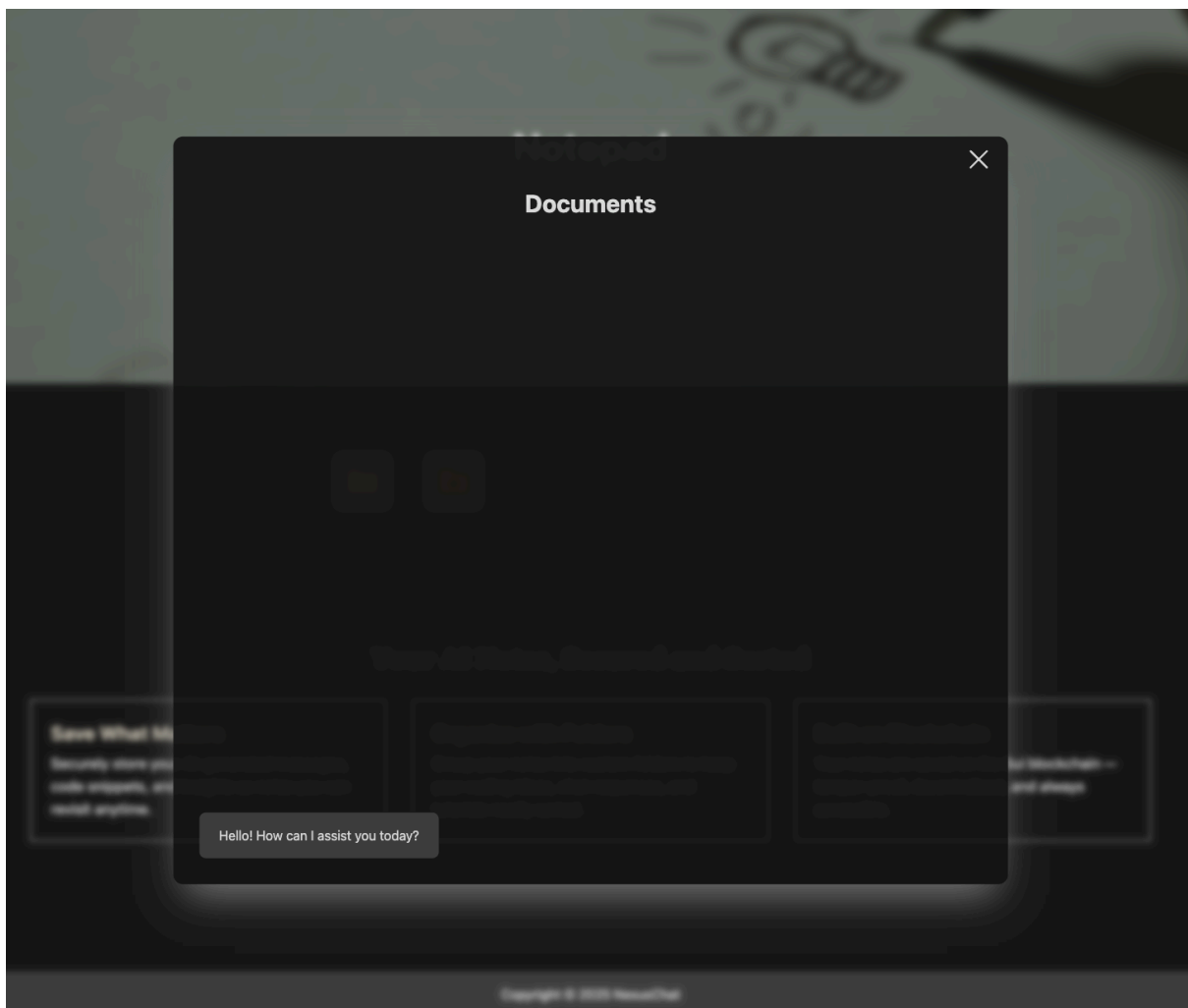


Image 4.13: Note Display

5 Conclusion and Future Work

5.1 Introduction

This thesis set out to address a fundamental problem in the modern digital world: the balance that needs to be kept between the utility of advanced AI applications and the critical need for user data privacy and sovereignty. The NexusChat application was conceived, designed, and implemented as a practical solution to this challenge. Finally, a roadmap is been proposed for future enhancements and research directions that can build upon the foundation established, and by the team that will be formed in the future for the needs of further research.

5.2 Future Work and Potential Enhancements

The foundation laid by NexusChat opens up numerous exciting possibilities for future development. The following enhancements could address the current limitations and significantly expand the application's capabilities:

- **Hybrid Storage Models (Off-chain Data, On-Chain Proof):** To minimize cost and latency, a future version could adopt a hybrid storage model. The encrypted content of messages and notes could be stored on a decentralized storage network like IPFS or Arweave, while only the lightweight proof of ownership (a hash of the data) is stored as a Sui Object. This would drastically reduce on-chain storage costs while maintaining verifiability and user control.
- **User Selectable AI Model Integration:** To enhance flexibility and empower users further, a future implementation would allow users to choose which AI model they interact with. The backend architecture could be extended to support multiple leading AI providers (such as Google's Gemini, Anthropic's Claude, or open-source alternatives). The user interface would include a settings panel where users could select their preferred model based on their specific needs—whether prioritizing the highest possible quality, the fastest response time for quick queries, or even a more cost-effective model for general use. This feature would give users direct control over the balance between performance, cost, and the specific capabilities of the AI they engage with.

6 Acknowledgements

I would like to express my sincere gratitude to my supervisor, Christos K. Georgiadis, for his invaluable guidance throughout this thesis. I also wish to extend a special thanks to PhD candidate Giatzis Antonios for their consistent support and helpful advice. Finally, I am deeply grateful to my family and friends for their endless love, support, and encouragement.

References

Giatzis, A., Georgiadis, C. K., & Digkas, G. (2023, September). A Comparative Analysis of Ethereum Solidity and Sui Move Smart Contract Languages: Advantages and Trade-Offs. In 2023 6th World Symposium on Communication Engineering (WSCE) (pp. 34-38). IEEE.

Antal, C., Cioara, T., Anghel, I., Antal, M., & Salomie, I. (2021). Distributed ledger technology review and decentralized applications development guidelines. *Future Internet*, 13(3), 62.

Blackshear, S., Chursin, A., Danezis, G., Kichidis, A., Kokoris-Kogias, L., Li, X., ... & Zhang, L. (2024). Sui lutris: A blockchain combining broadcast and consensus. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (pp. 2606-2620).

De Sanctis, V. (2023). *Building web APIs with ASP.NET Core*. Simon and Schuster.

Jones, M., Bradley, J., & Sakimura, N. (2015). RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force. <https://doi.org/10.17487/RFC7519>

Latifa, E. R., & Omar, A. (2017). Blockchain: Bitcoin wallet cryptography security, challenges and countermeasures. *Journal of Internet Banking and Commerce*, 22(3), 1-29.

Liu, W., Cao, B., & Peng, M. (2023). Web3 technologies: Challenges and opportunities. *IEEE Network*, 38(3), 187-193.

Mao, R., Chen, G., Zhang, X., Guerin, F., & Cambria, E. (2023). GPTEval: A survey on assessments of ChatGPT and GPT-4. *arXiv preprint arXiv:2308.12488*.

Meta. (2024). React [JavaScript Library]. Retrieved from <https://react.dev>

Mysten Labs. (2024). Sui dApp Kit [Software Library]. Retrieved from <https://sdk.mystenlabs.com/dapp-kit>

Mysten Labs. (2024). Sui TypeScript SDK [Software Development Kit]. Retrieved from <https://docs.sui.io/guides/developer/sui-sdk>

Tailwind Labs. (2024). Tailwind CSS [CSS Framework]. Retrieved from <https://tailwindcss.com>

Vercel. (2024). Next.js [Web Framework]. Retrieved from <https://nextjs.org>

Welc, A., & Blackshear, S. (2023, October). Sui Move: Modern blockchain programming with objects. In Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (pp. 53-55).

Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4), 352-375.

Zou, W., Lo, D., Kochhar, P. S., Le, X. B. D., Xia, X., Feng, Y., ... & Xu, B. (2019). Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, 47(10), 2084-2106.

Tatipatri, N., Arun, S. L.: A comprehensive review on cyber-attacks in power systems: Impact analysis, detection, and cyber security. *IEEE Access* 12, 18147-18167 (2024).