# Contents

# 1 Abstract

За последнее десятилетие криптовалюты и токенизация стали неотъемлемой частью рынка, открывая революционные возможности для владения активами и обмена ценностями. Токенизация позволяет оцифровать реальные активы, способствуя повышению ликвидности, дробному владению и прозрачности. В то же время криптовалюты произвели революцию в концепции цифровой валюты, способствуя децентрализованным и безопасным транзакциям. Данный проект направлен на разработку прототипа модели токенизации реальных финансовых активов в виде токенов в сети блокчейн на примере QuickToken, открывая возможности для повышения финансовой доступности и инноваций.

**Ключевые слова:** Криптовалюта, Блокчейн, Токенизация, Decentralized finance (DeFi), Solana.

Over the past decade, cryptocurrencies and tokenization have become an integral part of the financial market, opening up groundbreaking opportunities for asset ownership and value exchange. Tokenization allows the digitization of real assets, facilitating greater liquidity, fractional ownership and transparency. At the same time, cryptocurrencies have revolutionized the concept of digital currency, facilitating decentralized and secure transactions. This project aims to develop a prototype model for tokenization of real financial assets in the form of tokens on the blockchain network, taking QuickToken as an example, unlocking new opportunities for increased financial accessibility and innovation.

**Keywords:** Cryptocurrency, Blockchain, Tokenization, Decentralized finance (DeFi), Solana.

# 2 Introduction

## 2.1 Basic Terms and Definitions

- **Cryptocurrency** - a digital currency, which is an alternative form of payment created using encryption algorithms.
- **Blockchain** - an infrastructure processed on decentralised server cluster that verify cryptocurrency transactions and maintain secured data about them.
- **Token** - a unit of digital currency or representation of value that exists on a blockchain or distributed ledger, typically used for transactions and interactions within a specific ecosystem or platform.
- **Tokenization** - the process of converting real-world assets into digital tokens that can be represented, recorded, and transferred on a blockchain or distributed ledger.
- **Solana** - is a public 3-rd generation blockchain platform with smart contracts functionality.
- **Web-wallet** - a self-custodial crypto wallet, that is a web browser extension and supports transaction signing on website.
- **Bank** - a financial institution authorized by a government or regulatory body to receive deposits, provide loans, and offer a wide range of financial services to individuals, businesses, and governments.
- **Holder** - an individual, organization, or entity that owns and holds financial assets, such as stocks, bonds, mutual funds, or other investment instruments, with the intention of generating returns or profits over the long term.

- **Asset** - a resource that has economic value and the potential to generate future benefits.

- **Bond NFT** - the tokenization of bonds using non-fungible tokens, and representing ownership and transfer of bonds digitally through NFTs on a blockchain network.

- **Portfolio Instance** - a specific collection or combination of financial assets, such as stocks, bonds, mutual funds, or other investment instruments.

- **Tokenization Platform** - a platform or service that specializes in tokenization, providing tools and infrastructure for the creation, management, and transfer of digital tokens on blockchain networks.

## 2.2 Background

As we progress toward a decentralised economy, new opportunities emerge. A number of central bank digital currency (CBDC) projects have been launched by central banks. Physical products and non-fungible assets that are represented by tokens on a blockchain are now included in digital business models, releasing billions of euros in illiquid assets and generating new revenue streams. The introduction of tokenization has significantly altered how we invest and raise money.

Tokenization is the process of creating blockchain-based tokens that may be exchanged, stored, and transferred online. These assets continue to exist "off-chain", but their underlying tokens live on the blockchain, serve as a store of value, and have all the rights associated with them. This idea is not brand-new. For instance, tokenization is commonly used in "air miles" systems, which may be changed into specific items, services, or trips. The same process is used with digital tokens.

Traditional illiquid assets can become more liquid and tradeable on secondary markets through fractional ownership, enabling a wider range of investors to engage in the ecosystem. Assets that have been tokenized enable quicker transactions with less administrative work. Many laborious manual operations can be automated and optimised via the use of smart contracts, and the clearing and settlement procedures may be made simpler and more effective.

The concept of tokenization opens up new opportunities for the financial industry as it moves toward a decentralized economy. It provides increased liquidity, accessibility and efficiency, setting the stage for innovative business models and broader investor engagement. Through the use of blockchain technology, tokenization is fundamentally changing the way assets are represented, traded and managed, shaping

the future landscape of investment and capital allocation.

## 2.3 Project goal

Develop an application that prototypes the QuickToken model: allow investors, like banks, to raise funds against standardized assets, like bonds, via blockchain. In order for investors to manage the assets, a primary and secondary market for such tokens is implemented. An NFT contract enables the investor who purchases it to make a profit in the future, as specified in the terms of the contract.

## 2.4 Project objectives

1. *Learning Vue.js and webpack:* Acquire knowledge and proficiency in Vue.js, a progressive JavaScript framework, and webpack, a module bundler, to build the website.

2. *Draw a website Concept/Template:* design and outline the visual and structural aspects of the website using Figma application.

3. *Create smart-contract architecture, establish website interaction with blockchain* define the architecture and structure of the smart contracts that will be deployed on the blockchain.

4. *Develop Solana smart-contract:* implement the smart contract functionality using the Solana blockchain, ensuring secure and efficient execution of transactions and interactions.

5. *Develop Website:* utilize Vue.js and webpack to develop the website, integrating the designed concept/template and incorporating the necessary functionalities to interact with the smart contract and blockchain.

6. *Check and test the quality:* conduct thorough testing and quality assurance measures to identify and resolve any issues, ensuring the website's functionality and performance.

7. *Prepare for presentation, possibly run on a remote server:* prepare the project for presentation, showcasing its features and functionalities. Additionally, consider deploying the website on a remote server to make it accessible to users beyond the local environment.

## 2.5 Allocation of tasks among teammates

Our team consisted of two members - Anna Gertsog and Grigory Zhitniy.
Anna was engaged in the design of the website layout and all its elements in Figma, then made the website make-up using HTML and CSS markup languages and Vue.Js framework to develop the user interface.

Grigorii was responsible for the engineering of the whole system, developed smart contracts in the Rust language for Solana, built and deployed the site using Docker as well as integrated Phantom wallet into the site.

# 3 Review comparative analysis of sources

## 3.1 QuickToken

The original project, unlike QuickToken, follows the same idea of being an intermediate between legal organisations and crypto-investors, presenting packed loans as tokens into Ethereum blockchain. Unlike QuickToken we use the Solana Blockchain that provides much cheaper and faster transactions than any Etherium compatible blockchains. Solana has "closed" smart-contract code that provides stronger security.

Also we have managed to pack all data into blockchain, so our model is more trustful.

By leveraging Solana's capabilities and offering a more cost-effective and secure solution for bundled loan tokenization, our project effectively addresses the evolving needs of the financial industry. We aim to facilitate the seamless integration of blockchain technology into traditional finance, increasing efficiency and opening up new opportunities for both legal entities and crypto-investors.

## 3.2 Brickken

Brickken also specializes in presenting various assets in the form of tokens on the Ethereum blockchain. Using blockchain technology, they aim to provide greater liquidity and accessibility to traditionally illiquid assets. Unlike some other blockchain projects, Brickken does not rely on its own token as an intermediary between investors and borrowers.

This approach reduces risk for users because they are not required to buy or hold a specific token to participate in the platform. Instead, Brickken's model includes a built-in mechanism to sell and return funds in a timely manner, all enabled by blockchain. Brickken also emphasizes integration of its platform with wallets, apps, centralized exchanges and trading venues.

## 3.3 Summary table of competitors and related projects

Upon reviewing the competitors of QuickToken, the model we seek to replicate, we thoroughly examined eight other similar projects. Through comprehensive research and analysis, we assessed the strengths and weaknesses of each platform. QuickToken stands out with its competitive commission rate of only 1%, making it an appealing choice for users.

Additionally, QuickToken offers a wide range of investment options, ensuring diversity and maximizing opportunities for investors. The platform boasts a high level of liquidity, providing users with efficient and timely transactions. Furthermore, QuickToken utilizes blockchain technology, enhancing security and transparency. Notably, QuickToken operates both as a peer-to-peer (P2P) and peer-to-business (P2B) platform, catering to different investment needs.

These factors collectively contribute to the attractiveness and reliability of the project. However, in our assessment, we believe that further lowering the commission rate could attract an even larger customer base and foster increased customer acquisition.

| | QUICK TOKEN | MINTOS | CREDIMI | TWIMO | FELLOW FINANCE | OCTOBER | PEERBERRY | BLOCKFI | NEXO |
|---|---|---|---|---|---|---|---|---|---|
| Commission rates | 1% | 1,8% | 2% | 2,5% | 1,5% | 4,5% | 2% | 25% | N/A |
| Applied technology | P2P, P2B | P2P | SME LENDING | P2P | P2P, P2B | SME LENDING | P2P | P2P, P2B | P2P, P2B |
| Diversity investment | + | + | - | + | + | + | + | - | - |
| Level of liquidity (high/low) | + | - | + | - | - | - | - | - | + |
| Guarantee for investors | + | - | - | + | + | - | + | - | + |
| Blockchain Technology | + | - | - | - | - | - | - | + | + |

# 4 Functional and non-functional requirements

## 4.1 Functional Requirements

- The system lets the Bank create a new Portfolio Instance, consisting of a list of Assets.

- The system provides a built-in mint tool for Bond NFTs, each NFT keeps a combination of fraction of Portfolio's Assets. Combinations are set up by the Bank.
- The system provides a built-in mechanism for Bond NFTs release via sales on website.
- The system lets Holder to observe owned Bond NFTs and all available to buy Bond NFTs on market, displaying included Assets.
- The system distributes returns from Bank funds according to loans paid off.

## 4.2 Non-Functional Requirements

- The system must work on the Solana Blockchain.
- The system must not use any other server logic other than blockchain and must store all data in blockchain accounts.
- The system does not require any registration and does not use internal custodial wallets for both Bank and Holder.
- The system uses web-wallets for transaction signing.

## 4.3 Software Interfaces

- To communicate with the system, users must use Chrome as the only way to communicate with the system, making transactions with one of these web-wallets: "Phantom", "Solfare".
- Website and server interact via HTTPs 1.1 Solana RPC request.

# 5 Project implementation

## 5.1 Architecture

*This section was done by Grigorii Zhitnii:*

This diagram shows the relationships of the structures in a smart contract, each structure is separately recorded in the Solana accounts as small pieces of memory. The main structure is Portfolio, it includes a description of all included bonds and their parameters, the number of purchased assets and their breakdown, as well as the payout matrix. The Bond structure stores information about the filling of a particular asset, and standard NFT implementations through the Metaplex library were also used, each asset was represented as a unique NFT.

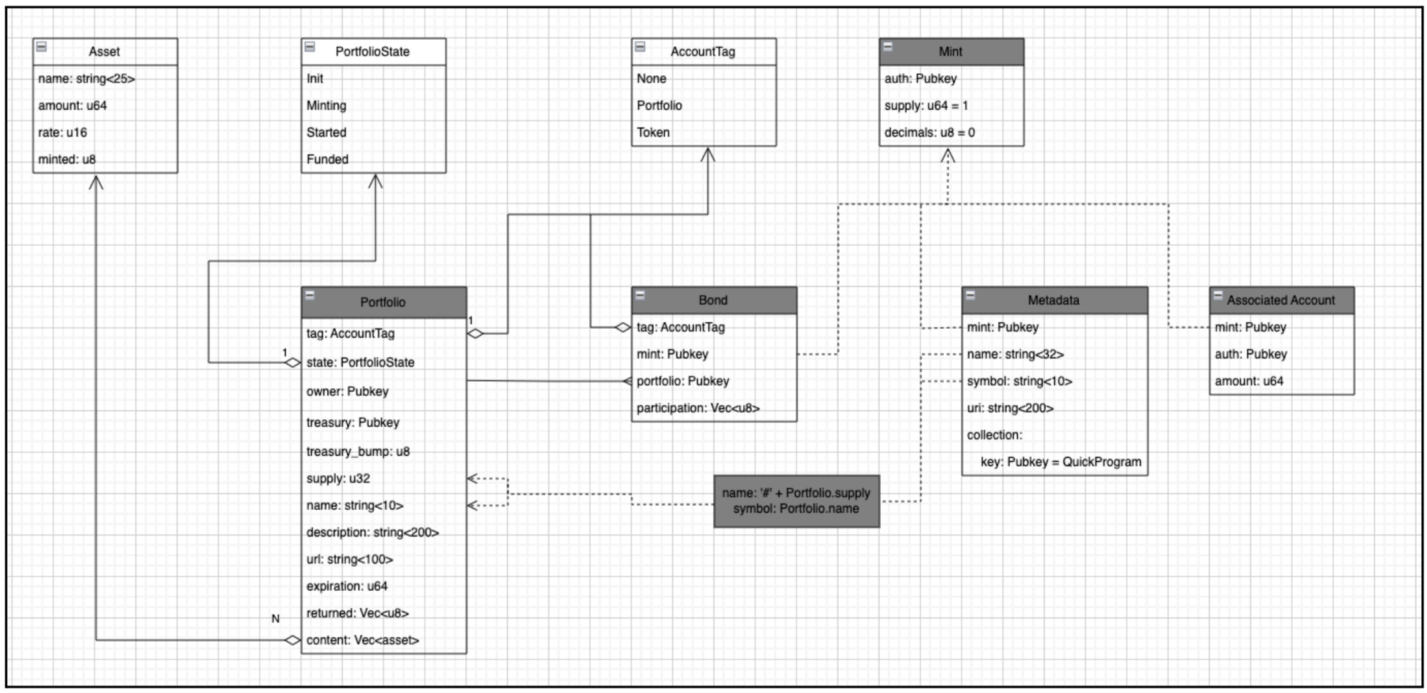


Figure 1: Smart-contracts structures

In Solana there is a concept of PDA - Program Derived Account, this feature allows you to create and sign an account from the program, to get the program address uses the native hashing algorithm - sha-256, as seeds use other accounts and the program address. We used the scheme below to get the addresses, so the

description of the asset can be procedurally obtained from the address of the prodged NFT, and the token storage account through the address of the portfolio.

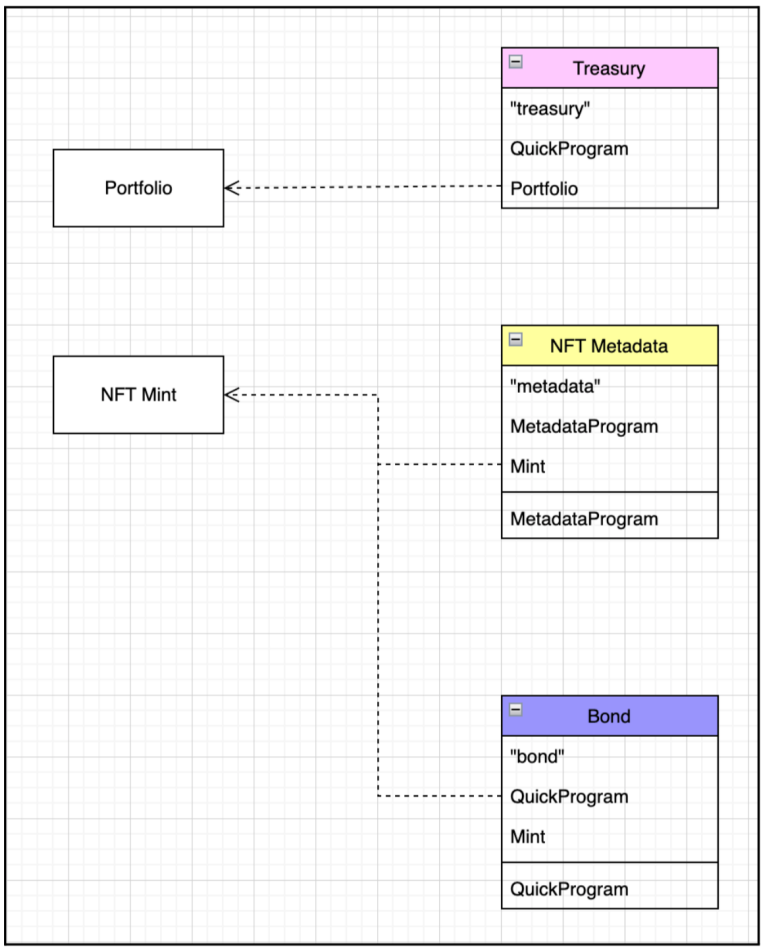


Figure 2: Accounts PDA relation

In the following, we developed the sequences of interaction between the website and the blockchain. Figure 3 shows the sequence of searching and displaying the blockchain assets using only queries to the blockchain network, with a special emphasis on the caching of the resulting values.

As mentioned above, in Solana, data is stored in separate accounts and associated with addresses. Solana, like the Rust language, follows the Resource Oriented ideology, so complex account search queries similar to SQL queries are possible in Solana, so implementation without a dedicated server is possible on this network. In order to search for saleable assets, we use multiple queries, starting with a search for all

portfolios under sale, then the search for all NFTs owned by those portfolios.

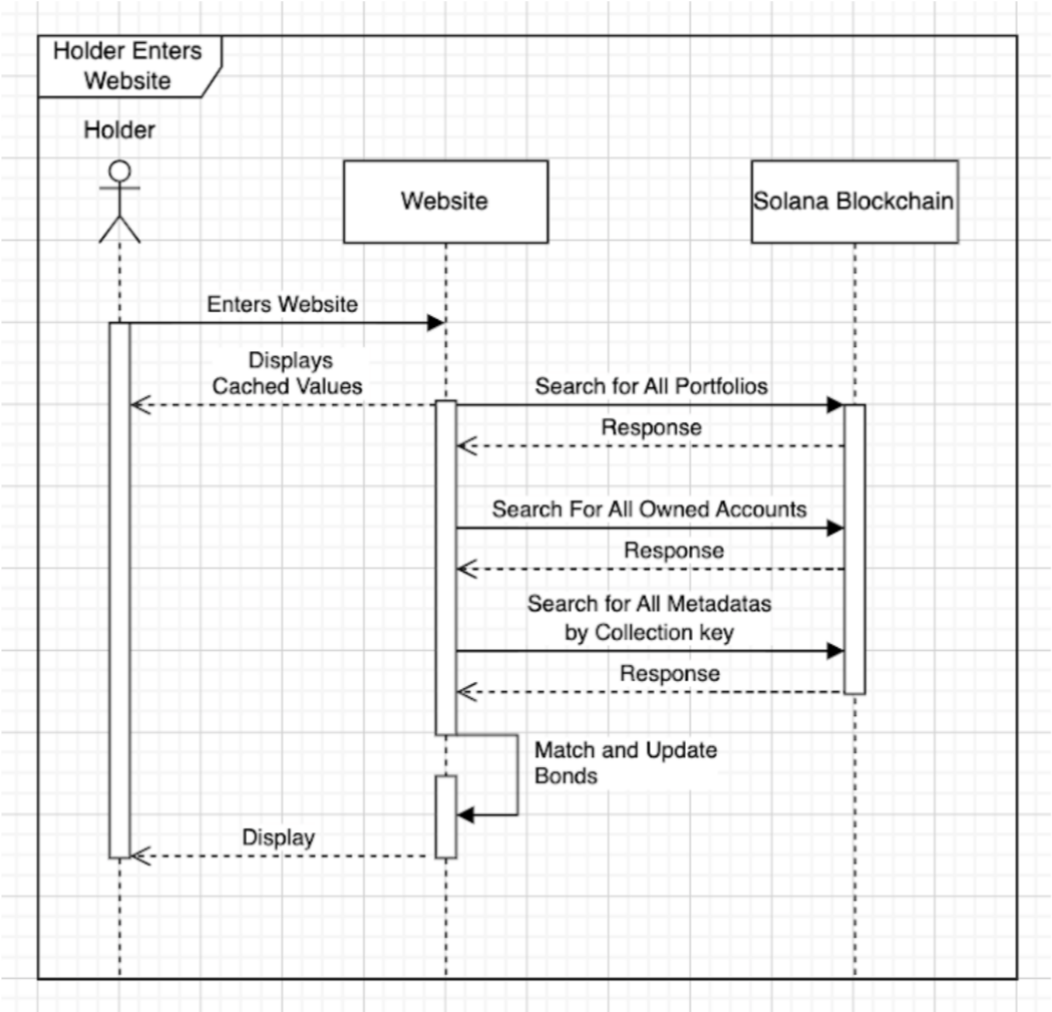


Figure 3: Interaction sequences (1)

Figure 4 represents the search sequence of all portfolios created by the bank. This is done by searching through all possible accounts with the Portfolio tag and the owner's address.

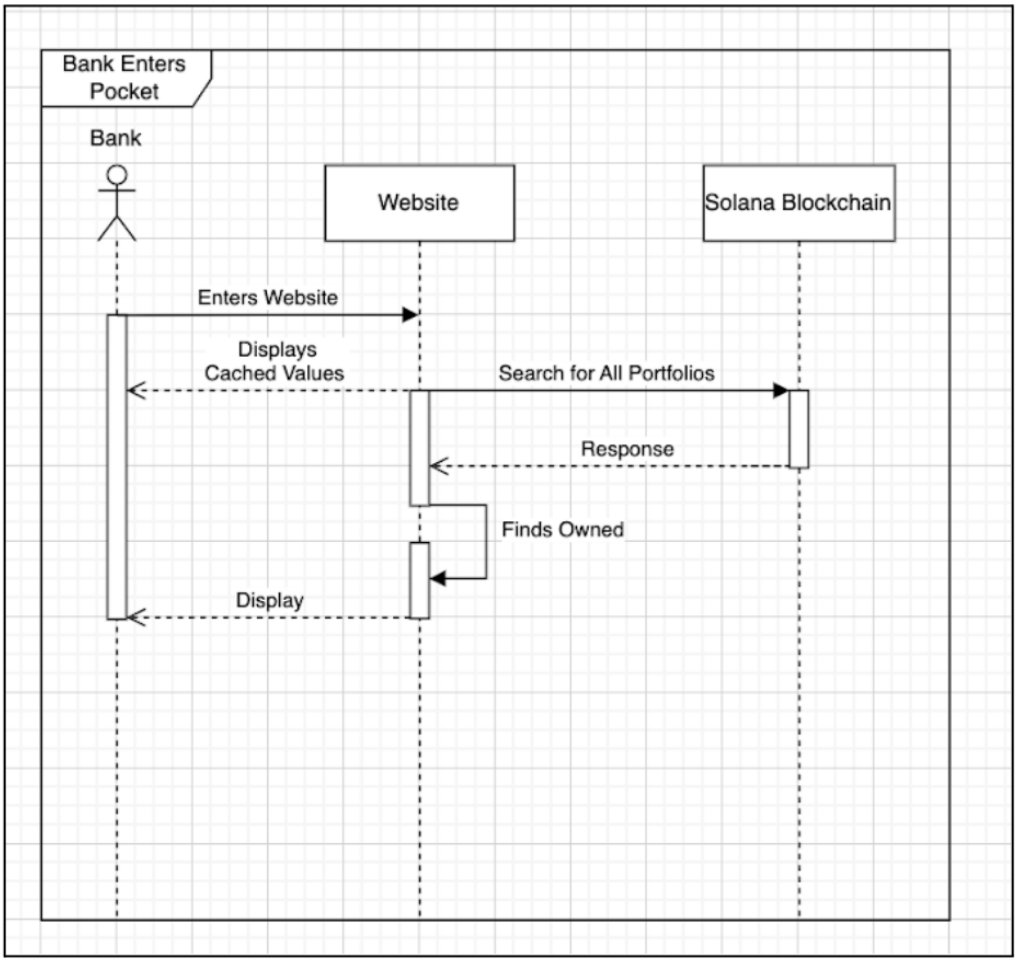


Figure 4: Interaction sequences (2)

Figure 5 illustrates the sequence of creating a portfolio, which is divided into 4 parts:

1. Creating a portfolio by adding all the information about the bank.

2. Entering information about each of the bonds used.

3. Minting NFTs under each of the assets.

4. The start of the portfolio action.

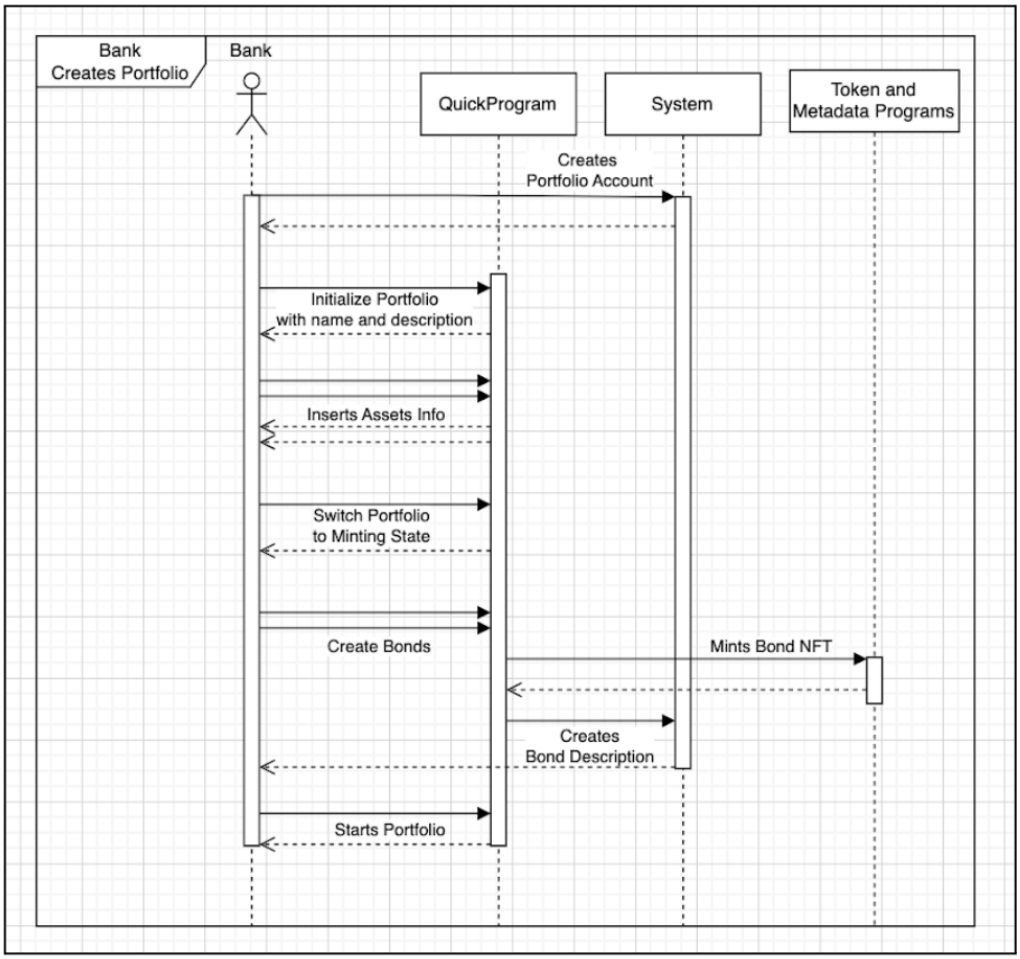


Figure 5: Interaction sequences (3)

## 5.2 Development of the website layout

*This section was done by Anna Gertsog:*

Since the beginning of the project, after discussing all the tasks and the outlined plan with the supervisor and my teammate, I started to design the future website using a special program for the interface development and prototyping - Figma.

Firstly I designed the landing page, using the official QuickToken website as a reference. On the navigation bar I added 2 buttons for switching to other pages - for banks and for users, and they were duplicated in the center of the page. Also, I placed a purple button to connect a cryptocurrency wallet - *"Select Wallet"*.

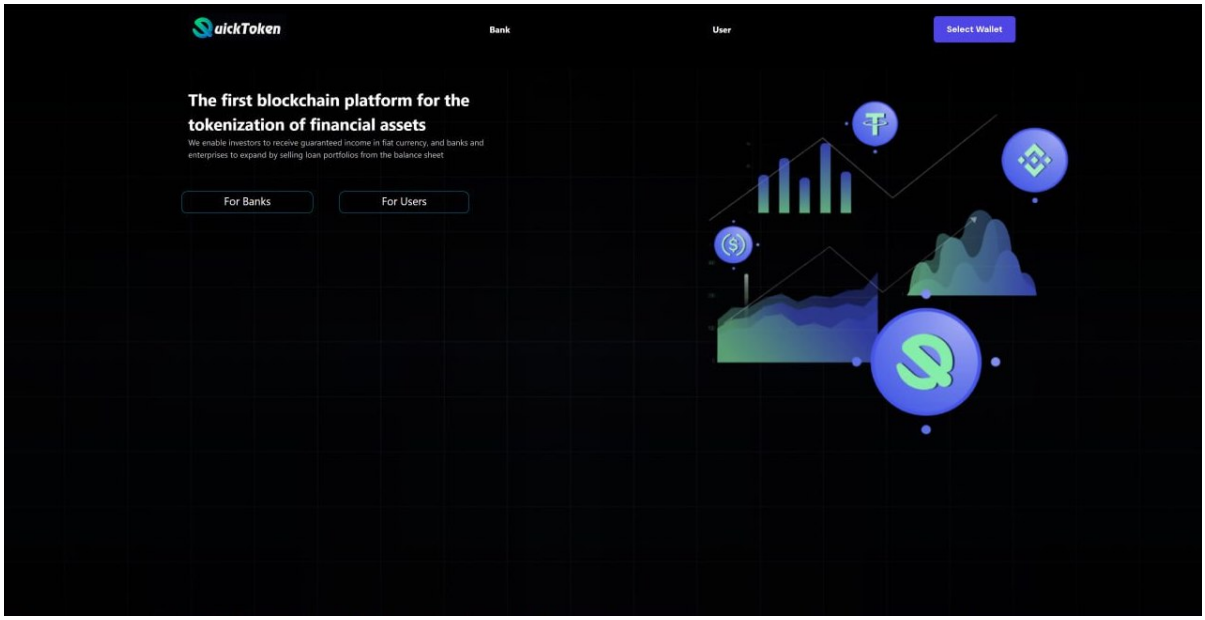


Figure 6: Landing Page

Then I designed a page for creating a portfolio under the name Query, in which the user enters a description of his bank, adds a list of bonds and specifies the division of bonds between different NFTs.

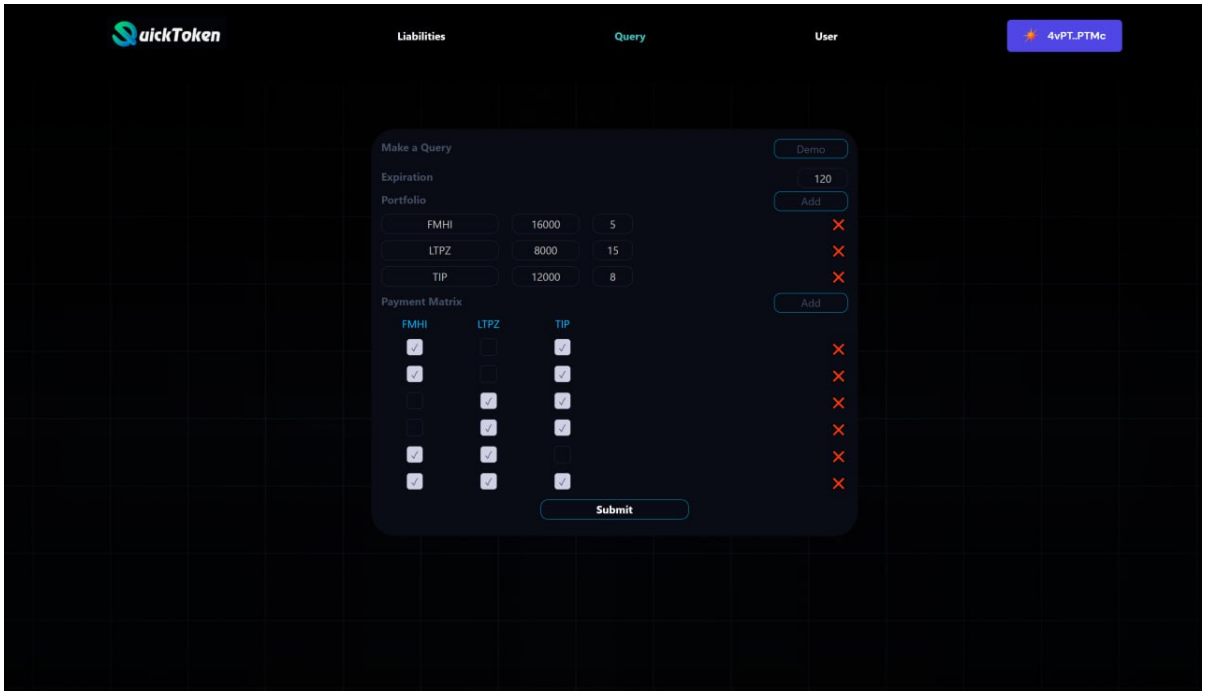


Figure 7: Query

To fill out the matrix of redeemed bonds, I created another page with a repeating design.

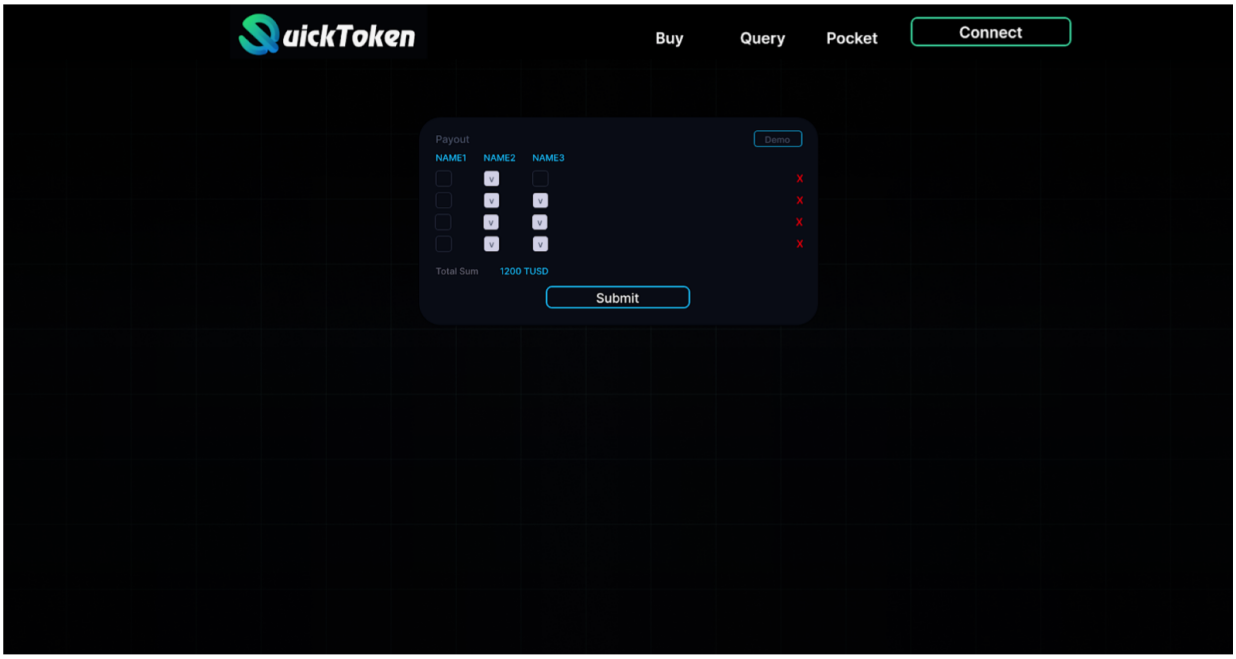


Figure 8: Payment Query

The next page presents the marketplace of our NFTs, I have generated 10 icons to distinguish the asset cards, which are randomly allocated between them. For each of the NFTs, we show information about the bank, a list of included bonds, and calculate the value and the rate of the asset.

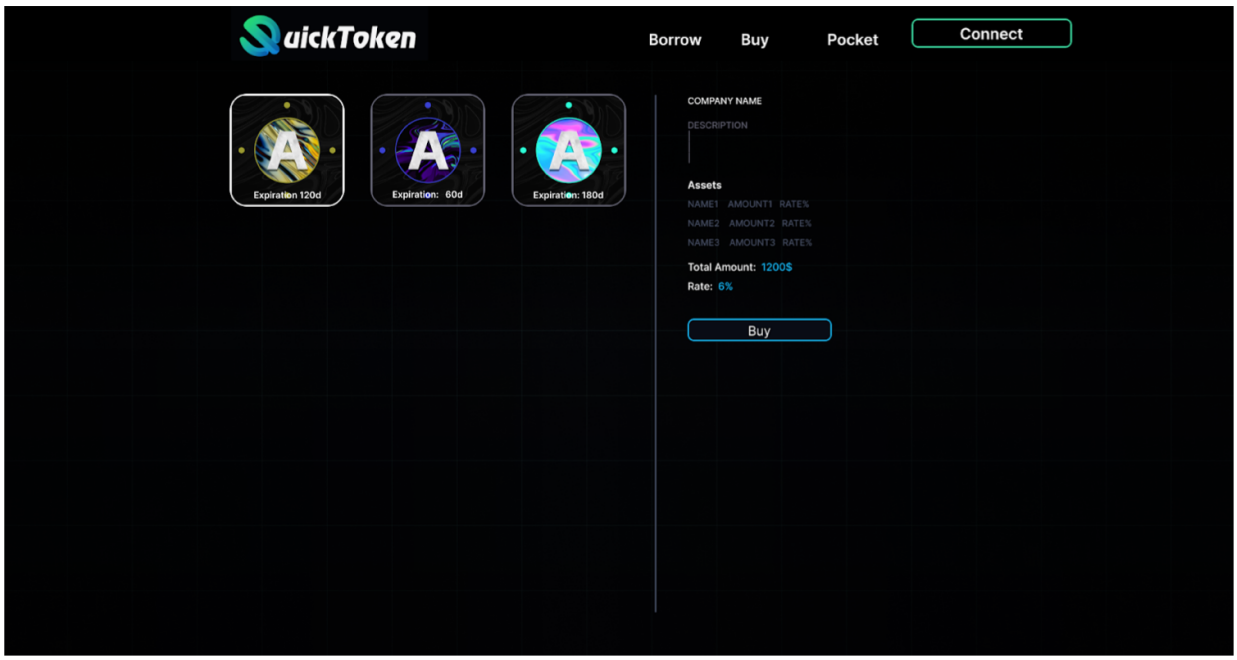


Figure 9: NFT Pool

The same elements were used for the user's wallet, only now after the bank has deposited money for the bonds, the paid positions will be highlighted and the resulting value will be recalculated.

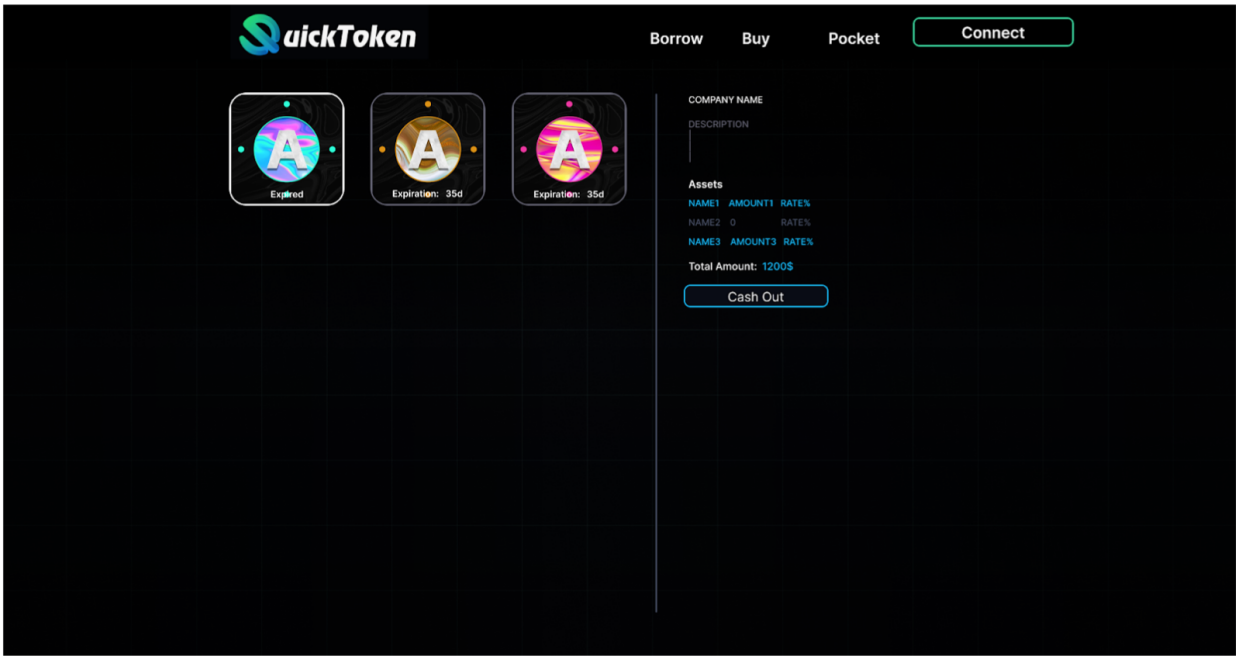


Figure 10: User pocket

For the bank portfolio, the same idea of random cards was used.

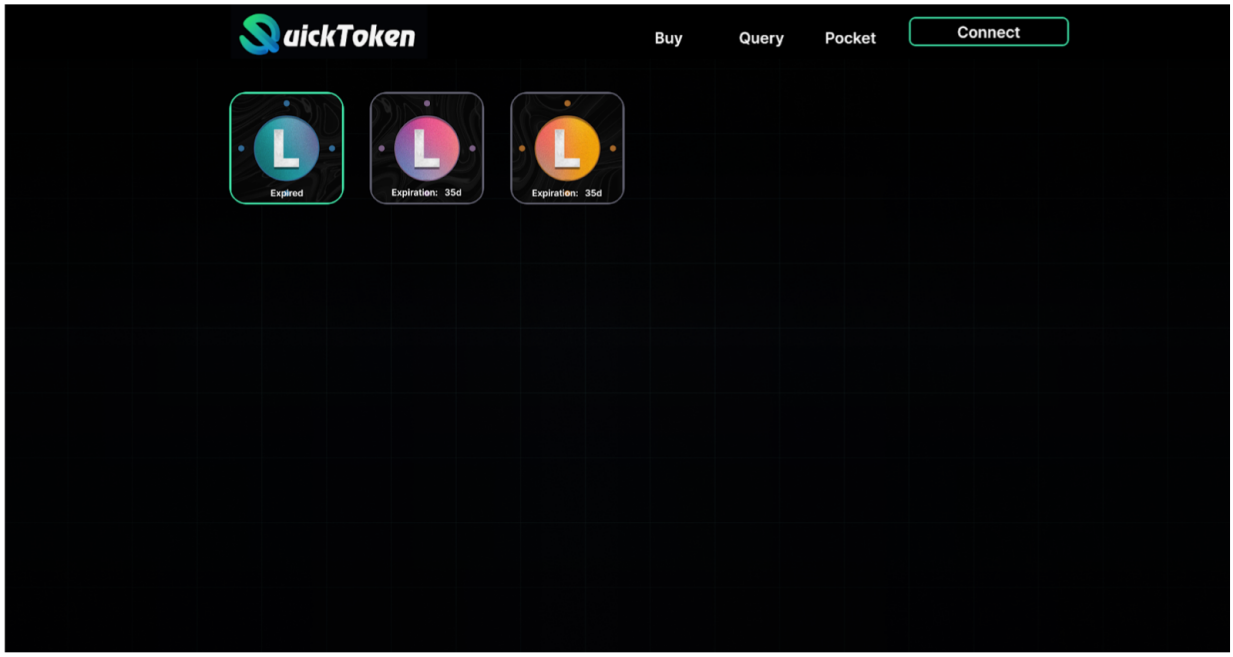


Figure 11: Liabilities

## 5.3 Backend development

### 5.3.1 Smart contract

*This section was done by Grigorii Zhitnii:*

To implement the idea, it was decided to use Solana's native library instead of the Anchor library or its substitutes, since the main work is done with custom states, for languages Rust over C++ was chosen as the most used. From the sequences presented, these instructions were highlighted:

```rust
#[derive(BorshSerialize, BorshDeserialize, Debug, Clone)]
pub enum Instruction{
    CreateCompany {name: Vec<u8>, description: Vec<u8>},
    CreatePortfolio {bond_count: u32, image: u8},
    InsertBond {name: Vec<u8>, amount: u64, rate: u16},
    MintAsset {url_id: u8, participation: Vec<u8>},
    BuyAsset,
    Start {period: u64},
    StopSales,
    Payout {mask: Vec<u8>},
    CollectPayment
}
```

Figure 12: Smart contract instructions

Based on the architecture we proposed above, the standard states have been implemented, and the portfolio characterizes all information about the portfolio, including its status, period, and stores links to the token repository and owner.

```rust
#[derive(BorshSerialize, BorshDeserialize, Debug, Clone, PartialEq)]
pub enum PortfolioStatus{
    Created,
    Started,
    Sold,
    Payed
}


#[derive(BorshSerialize, BorshDeserialize, Debug, Clone)]
pub struct Portfolio{
    pub tag: AccountTag,
    pub company: Pubkey,
    pub token_storage: Pubkey,
    pub bump: u8,
    pub status: PortfolioStatus,
    pub period: u64,
    pub end: u64,
    pub image: u8,
    pub bonds: Vec<Bond>,
    pub payout_mask: Vec<u8>
}
```

Figure 13: Portfolio state

The Asset stores a link to the portfolio, to the NFT address to which it belongs, and a list of bond numbers, with no duplicate information about the bonds.

```rust
#[derive(BorshSerialize, BorshDeserialize, Debug, Clone)]
pub struct Bond{
    pub name: Vec<u8>,
    pub amount: u64,
    pub rate: u16,
    pub used: u16,
    pub sold: u16
}
```

Figure 14: Asset state

For the release of NFTs we have written a separate functionality, the NFT format fulfills the Metaplex Metadata V3 standard, which was the most relevant at the stage of development. In addition to the recorded data in the blockchain, the standard requires metadata in json format and pictures placed on external sources, we were able to use Git as the storage, which allows you to change the parameters in real-time.

```rust
pub fn create_metadata<'a>(url_id: u8, auth_ai: AccountInfo<'a>, mint_ai: AccountInfo<'a>, metadata_ai: AccountInfo<'a>, collection: &Pubkey,
                           system_program: AccountInfo<'a>, metadata_program: AccountInfo<'a>, sysvar: AccountInfo<'a>) -> Result<(), ProgramError> {
    let idx = create_metadata_accounts(
        *metadata_program.key,
        *metadata_ai.key,
        *mint_ai.key,
        *auth_ai.key,
        *auth_ai.key,
        *auth_ai.key,
        String::from("Asset"),
        String::from("AAA"),
        String::from(ASSET_METADATA_URLS[url_id as usize]),
        Option::Some(Vec::from([Creator{address: *auth_ai.key, verified: true, share: 100}])),
        0,
        true,
        true,
        Option::Some(
            Collection{
                verified: false,
                key: *collection
            }
        ),
        Option::None,
        Option::None
    );
    invoke(
        &idx,
        &[auth_ai, mint_ai, metadata_program, system_program, sysvar, metadata_ai]
    )?;
    msg!("Created metadata");
    Ok(())
}
```

Figure 15: Metadata initialization

## 5.4 Frontend development

*This section was done by Anna Gertsog:*

### 5.4.1 Website layout

The website was developed using HTML and CSS, the two main and most popular languages for creating web pages. HTML is primarily used to structure and content of the page, defining various elements such as headings, paragraphs, images and links. CSS is used to design and style elements, allowing you to specify colors, fonts, element placement, and other visual attributes.

Also in the process of developing our site, I decided to use the powerful Bootstrap library to improve its navigation, page layout, and overall user experience. Bootstrap is a popular framework that offers a wide range of pre-designed components, styles, and a responsive grid system. It allows developers to quickly and efficiently create visually appealing and mobile-friendly web pages.

The utilization of Bootstrap's responsive grid system proved instrumental in crafting a flexible and adaptable layout for our website. By leveraging the intuitive row and column structure provided by Bootstrap's grid classes, we ensured that our content seamlessly adjusted to various screen sizes and device types. This level of responsiveness is crucial in today's digital landscape, as users access websites through a multitude of devices. Additionally, Bootstrap equipped us with an array of CSS styling options and convenient classes, streamlining the customization process and enabling us to enhance the visual presentation of elements without resorting to complex CSS rules.

Also, I integrated the Vue router into our website development to optimize navi-

gation and enhance user experience. The Vue router allows us to define routes for each page, enabling users to easily access specific content through URLs or navigation menus. Its dynamic routing capabilities enable us to load and display content based on user interactions or conditions. Additionally, the Vue router supports nested routes, improving the organization and scalability of our website's structure.

```
const routes = [
    { path: '/', component: MainPage, name: 'Home' },
    { path: '', component: MainPage, name: 'Home' },
    { path: '/bank/query', component: PortfolioCreate, name: 'Query' },
    { path: '/bank/', component:  LiabilitiesPage, name: "Bank"},
    { path: '/bank/payout', component:  PortfolioPay, name: "Payout"},
    { path: '/user/', component:  UserPocket, name: "User"},
    { path: '/user/buy', component:  AssetMarket, name: "Market"},
]
```

Figure 16: Router elements

We implement the base page in the App.vue component, where we define the main navigation menu and implement injections for individual pages. In the App.vue component, we also implement common functionality that applies to all pages:

- *Wallet Verification:* to implement wallet verification and validation of the user's wallet to ensure the safety and validity of their actions.
- *Signing and sending transactions:* to enable the user to sign and send transactions through our platform.

```
<template>
  <LoadPopup/>
  <NavBar/>
  <img src="@/assets/cells.png" style="position: absolute; z-index: -1; width: 100vw; height: calc(100vh - 120px);"/>
  <router-view/>
</template>
```

Figure 17: App.vue component

In terms of the main components of the page, we can distinguish 6 of them:

1. **NFT card:** it displays relevant information such as the token's image, which is randomly generated.


Figure 18: NFT card

2. **Submit button:** this component allows users to submit their actions or transactions.


Figure 19: Submit button

3. **Asset description component:** this component provides detailed information about a specific asset or NFT, such as the name of the bank and its brief description, information about assets, total amount, rate of interest and time of expiration.

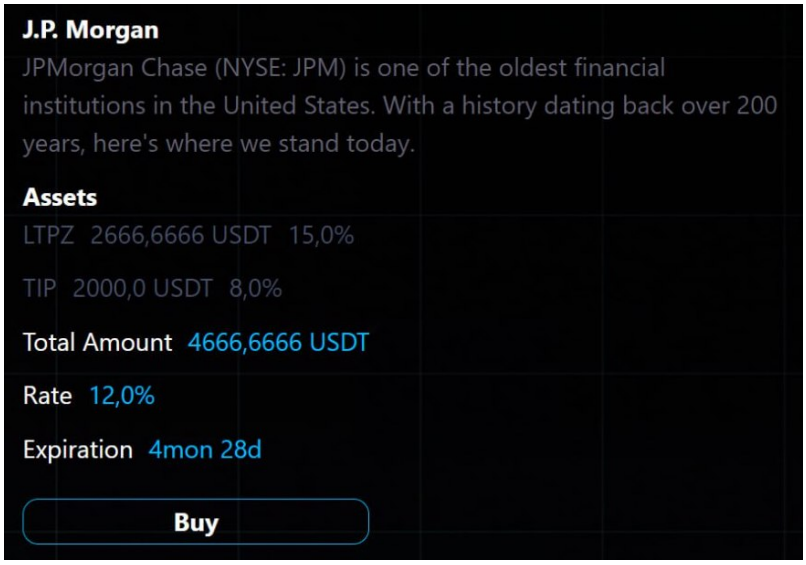


Figure 20: Asset description component

4. **Navigation bar:** the navigation bar is a fundamental component that facilitates easy navigation throughout the application. In this case here are directions to Liabilities, Query, User and the button to click on the wallet button.


Figure 21: Navigation bar

5. **Input field:** which is used to capture user input, such as text, numbers, or other data, for instance, the information on Figure 20.


Figure 22: Input field

6. **Load pop-up:** the load pop-up is a user interface element that appears when the application is processing data or performing an action that requires some time.

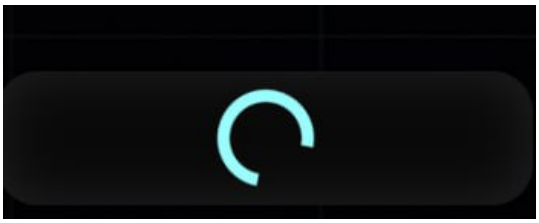

Figure 23: Load pop-up

In conclusion, the frontend development was the crucial stage of our project, where we used modern technologies and tools such as HTML, CSS, Vue.js and Bootstrap to create an interactive and user-friendly UI. We successfully implemented functionality that provided navigation, interaction with blockchain, display and interaction with NFT, as well as overall application logic.

# 6 Conclusion

## 6.1 Results

The application for tokenization of fiat financial assets was completely developed, prepared and deployed on a server. Our application is compatible not only with bonds but also with all financial instruments with a final maturity date. The potential clients of our application will be financial organizations of SPV - Special Purpose Vehicle direction, which want to expand into cryptocurrencies, and from the end-stakeholders side - physical and legal investors, who want to use liquidity and convenience of cryptocurrencies, but do not want to bear its risk.

## 6.2 Project perspectives

To sum up the current results, we can add about the future and prospects of our project. Our project anticipates many options for developing its capabilities and proposals. Based on the current structure, the project aims to revolutionize the way digital assets are managed and traded by introducing a number of transformative features and capabilities.

A key aspect of the future roadmap of the project is the integration of wallets, given the importance of secure and convenient solutions for such an area, the project plans to integrate with popular cryptocurrency wallets. In addition, the project's near-term prospects include the development of a mobile application on IOS and Android platforms, which will provide users with easier access to a number of features and functions, including portfolio management, asset tracking and optimized trading capabilities.

Moreover, the project plans to provide NFT registration on various trading plat-

forms, and aims to connect with centralized exchanges and trading venues, providing users with enhanced liquidity and trading capabilities. This strategic integration will allow tokens to trade seamlessly on established platforms, leveraging the liquidity and user base of these centralized exchanges to maximize trading efficiency and accessibility.

Finally, the project may include the creation of its own decentralized exchange which will facilitate an unbiased and transparent environment for P2P token trading, allowing users to retain full control over their assets and opening up a variety of trading opportunities.

## 6.3 Important Links

- **GitHub Repository:** https://github.com/Zarve8/QuickToken
- **Our website:** https://www.zarve.xyz/
- **Docker:** https://hub.docker.com/repository/docker/zarve1/quick_website

# 7 Bibliography

# References


[1] Metaplex docs | nft standard. URL: https://docs.metaplex.com/.

[2] Rust documentation. URL: https://doc.rust-lang.org/.

[3] Solana documentation. URL: https://docs.solana.com/.

[4] Brickken. Official website. URL: https://brickken.com/consultation/.

[5] Gans J. S. Catalini, C. Some simple economics of the blockchain. 2016.

[6] Phantom. The crypto wallet for tokens. URL: https://phantom.app/.

[7] QuickToken. Official website. URL: https://quicktoken.org.

[8] Islam M. R. Rahman, M. R. An improved tokenization model for nft trading on solana blockchain. 2022.

[9] Nugroho Y. A. Handayani P. W. Rizqullah, R. Asset tokenization on solana blockchain network: A case study in the real estate sector. 2022.

[10] Solana. Official website. URL: https://solana.com/ecosystem.

[11] Solfare. A secure and powerful. solana wallet. URL: https://solflare.com/.

[12] TechTarget. Official website. URL: https://www.techtarget.com.