Software Project Specification

&

Implementation schedule

for

NextNonce: Crypto Portfolio Tracker

NextNonce is an Android application designed to provide secure and effortless tracking of cryptocurrency wallets. The project leverages blockchain technology and features a user-friendly interface, making crypto tracking accessible and intuitive for everyday users.

0.1.1

Oleksandr Yuriichuk

19.11.2024

Contents

Revision table

Name	Date	Reason for change	Version
Oleksandr Yuriichuk	19.11.2024	The initial text	0.1.0
Oleksandr Yuriichuk	11.12.2024	Add project name	0.1.1

1. Basic information

1.1. Description and scope of the software project

The project, **NextNonce,** is a crypto portfolio tracker Android application that enables users to monitor their cryptocurrency wallets securely and effortlessly. Most Web3 applications are primarily accessible from PCs or notebooks. Even when they are available on mobile phones, they often have unattractive designs, poor performance, and unclear interfaces. This creates a gap, as most people spend the majority of their time on mobile devices - something they carry in their pocket and can access anytime. When users want to check the state of their crypto portfolio, they need a solution that is fast, simple, and accessible from anywhere.

This application addresses this need by focusing on on-chain tracking, providing a clear and intuitive interface tailored for average users who want to track their holdings without technical complexity.

What it brings new:

- A mobile, user-friendly solution for tracking cryptocurrency wallets.
- Token aggregation across chains for a consolidated portfolio view.
- Support for non-EVM blockchains, starting with Starknet, offering a broader range of compatibility.
- An intuitive and minimalist interface designed for quick and seamless use, even for those new to cryptocurrencies.

Target group
The app is designed for individuals new to cryptocurrency or casual investors who need a straightforward tool to monitor their wallets without requiring deep technical knowledge.

Alternatives:

- DeBank**:** DeBank is evolving into a social media-like platform with an overloaded interface that can be overwhelming for users who just want to track their portfolio quickly and clearly.
- CoinMarketCap**:** CoinMarketCap requires users to manually update all portfolio changes, as it does not support on-chain tracking, making it inconvenient.

Differentiation:
Unlike these alternatives, this tracker avoids unnecessary complexity and emphasizes ease of use. It provides automatic on-chain tracking and a streamlined, read-only solution specifically tailored for simplicity, accessibility, and clarity.

1.2. The technologies used

Frontend

- **Kotlin**: The primary programming language for Android development, chosen for its modern features and seamless integration with Android APIs.
- **Jetpack Compose**: A declarative UI toolkit for building modern, intuitive, and responsive user interfaces on Android.
- **Room**: A database library for local storage, used to cache data like token metadata and wallet information.
- **Ktor**: A Kotlin-first HTTP client for making API calls to the backend and processing responses, designed for cross-platform compatibility.
- **Coroutines**: Kotlin's native solution for asynchronous programming, ensuring smooth and efficient handling of background tasks like API calls and database operations.

Backend

- **JavaScript + TypeScript**: The languages used for backend development, providing type safety and scalability for the project.
- **NestJS**: A framework for building modular and maintainable server-side applications.
- **Axios**: A promise-based HTTP client used for making requests to external APIs.
- **Ethers.js**: A library for interacting with Ethereum and EVM-compatible blockchains.
- **Starknet.js**: A library for interacting with the Starknet blockchain.
- **Redis** (optional): For caching frequently accessed data, such as token metadata or prices, to optimize performance.
- **PostgreSQL** (or similar): A relational database used to store user account data, wallet information, and other persistent records.

2. Brief description of the software project

2.1. The reason for the creation of the software project and its basic parts and the goals of the solution

The primary purpose of the project is to address the gap in mobile solutions for tracking on-chain crypto assets. Most existing tools are either limited to PCs or notebooks, overly complex, or require manual data entry. This application simplifies the process by focusing on mobile accessibility, ease of use, and automatic on-chain data tracking.

Basic Parts of the System
The application will be composed of the following parts:

1. **Frontend (Android App):**
    - Provides a user interface for entering wallet addresses and viewing portfolio data.
    - Displays aggregated token balances across chains in a visually clear and simple manner.
    - Includes features for fetching real-time token prices to display portfolio value.

2. **Backend Service:**
    - Handles API requests to blockchain data providers (e.g., Etherscan, CoinGecko) and token price APIs.
    - Aggregates and caches data to reduce latency, API costs, and rate-limit issues.
    - Manages token aggregation across chains (e.g., summing balances of the same token across different blockchains etc.) to provide a unified portfolio view.
    - User account management, allowing users to log in from new device to access saved wallets, portfolios, and user settings

2.2. Motivational usage example

Imagine a casual cryptocurrency investor, Bob, who holds assets across multiple wallets and blockchains. Bob often checks the status of his portfolio but finds it inconvenient to switch between different platforms or manually update changes in apps like CoinMarketCap.

With the NextNonce tracker, Bob can simply add his wallet addresses to the app. Within seconds, the app displays the total value of his holdings, aggregating tokens across supported blockchains into a unified view.

2.3. Application environment

The application is designed to run on mobile devices with **Android 11 (API Level 30)** or higher.

2.4. Limitations of the project

1. **Dependency on Third-Party APIs**
    - The application relies on external APIs (e.g., Etherscan, CoinGecko) for blockchain data and token prices. These services often impose rate limits, usage restrictions, and potential changes to their APIs, which could affect the app's performance and data accuracy. Additionally, any downtime or discontinuation of these services would impact the application's functionality.

2. **Security Requirements**
    - While the app will not store sensitive information such as private keys, it must ensure secure handling of public wallet addresses and API keys to prevent unauthorized access or misuse.
3. **Scalability**
    - Adding support for additional blockchains, particularly non-EVM chains, introduces complexity due to differences in their data structures and APIs. The initial implementation will focus on Ethereum and Starknet, with further expansion contingent on available time and resources.
4. **Dependency on Internet Access**
    - The application requires a stable internet connection to fetch real-time data from APIs. Offline functionality will not be supported, limiting usability in areas with poor connectivity.
5. **Time Constraints**
    - The project scope is limited by the development timeline. Features like additional blockchain integration may need to be postponed or excluded if time does not permit their full implementation.

2.5. Project documentation

To-do.

3. System Interfaces

3.1. User Interaction and Data Flow

The application provides a user-friendly interface, designed for quick and intuitive interactions. Users interact with the app primarily through the following:

Inputs

- **Wallet Addresses**:
    - Users enter public wallet addresses to track cryptocurrency balances and transaction history.
    - Multiple wallet addresses can be added for simultaneous tracking of various accounts.

Outputs

- **Portfolio Overview**:
    - Displays aggregated balances of tokens across chains, consolidating identical tokens into a unified total.
    - Shows the total value of the portfolio in a fiat currency (default USD).
- **Transaction History**:
    - Displays a timeline of transactions with token amounts and (if implemented) corresponding fiat values at the time of execution.

Data Flow

1. **User Actions**:
    - Users input wallet addresses or interact with displayed portfolio data.
2. **Data Requests**:
    - The frontend sends requests to the backend for wallet balances, token metadata, and transaction details.
3. **Local Caching**:
    - Frequently accessed data, such as token metadata and recent balances, is cached locally to improve responsiveness.

3.2. Internal Software Integration

The application comprises the following key components:

Frontend (Android App):

- **Role**:
    - Provides the user interface for inputting wallet addresses and viewing portfolio data.
    - Caches frequently accessed data (e.g., token metadata, wallet balances).
- **Interaction with Backend**:
    - Sends requests to fetch data for balances, transactions, and token metadata.
    - Receives and displays processed data from the backend.

Backend:

- **Role**:
    - Aggregates wallet data, token metadata, and transaction history from blockchain APIs.
    - Caches global data (e.g., token prices) for optimized performance.

- **Interaction with Database**:
    - Stores user accounts and linked wallets for retrieving portfolio data when logging in on a new device.
    - Maintains metadata for tokens and chains, reducing redundant API calls.

Database:

- **Role**:
    - Stores user accounts (if implemented), wallet addresses, and token metadata.
    - Supports backend operations with structured data for user portfolios and transaction histories.

3.3. Network Communication

The application communicates over the network for various tasks:

Protocols:

- **HTTP/HTTPS**:
    - Used for all frontend-backend and backend-external API communications.

Security Measures:

- All communication is encrypted using HTTPS to protect data in transit.
- Sensitive operations (e.g., user authentication, API key handling) are managed securely on the backend.

Synchronization:

- Limited to retrieving user portfolios during account login on a new device, avoiding real-time cross-device synchronization.

4. Other (non-functional) requirements

4.1. Performance requirements

Wallet Data View

- Viewing the list of tokens and their valuation for a single wallet should take no longer than 3 seconds under normal conditions.

Transaction History View

- Retrieving and displaying the transaction history for a single wallet should take no longer than 4 seconds under normal conditions.

Portfolio Valuation

- No performance requirement is specified for the total portfolio valuation, as the computation scales linearly (O(n)) with the number of wallets added by the user.

4.2. Requirements for extensibility and integrability

Support for Additional Blockchains:
The backend must be designed to allow the addition of new blockchains (e.g., other EVM-compatible chains, non-EVM blockchains like Solana) with minimal changes to the existing codebase.

Future Features:
The application must allow for the seamless addition of new features (e.g., cross-chain transaction history, DeFi data tracking) without requiring significant refactoring of the existing architecture.

Cross-Platform Compatibility:
The project should be designed in a way that creating a similar app for **iOS** would not require rewriting the entire codebase. To achieve this, the backend must remain platform-agnostic, exposing APIs that are consistent and reusable for any platform. Additionally, reusable business logic (e.g., API communication, data parsing, and caching) should be abstracted into a shared layer where possible, enabling easy adoption on other platforms like iOS.

5. Exclusions

The following aspects are explicitly not part of the guaranteed scope of this work:

1. **Centralized Exchange (CEX) Integration**:
    - The app will not support tracking assets held in centralized exchanges like Binance, Coinbase, or Kraken. It is limited to on-chain tracking only.
2. **Built-in Wallet Functionality**:
    - The application will not allow users to perform transactions or directly interact with their wallets' private keys. It is designed solely as a read-only tool for monitoring wallet balances and transactions.
3. **DeFi Tracking**:
    - While the project does not promise any support for tracking DeFi-related features (e.g., liquidity pools, staking, or yield farming), if time permits, some functionality might be explored. However, if the final project lacks DeFi tracking, this will be considered as planned.

4. **Support for All Blockchains**:
    - The initial implementation focuses on Ethereum, Starknet, and some EVM-compatible chains. While the project may explore adding other blockchains if time allows, support for any other blockchain is not guaranteed. If the final project does not include additional chains, this will align with the intended scope.
5. **Offline Functionality**:
    - The app will not provide comprehensive offline capabilities.
6. **Extensive Portfolio Analytics**:
    - Advanced analytics features, such as portfolio growth charts or detailed historical performance tracking, are out of scope for this project.
7. **Cross-Device Synchronization**:
    - The application will not provide real-time synchronization of user data across multiple devices. It could only support logging in on a new device to retrieve the same portfolio information.

Appendix A: Definition of Terms

Blockchain

A decentralized and distributed digital ledger that records transactions across multiple computers securely. Each transaction is grouped into a block and linked to the previous one, forming a chain.
Learn More

Wallet

A digital tool used to store public and private keys, enabling users to interact with blockchains to send, receive, and monitor cryptocurrency. In this project, the focus is on public wallets, which can be used to track balances without accessing private keys.
Learn More

Token

A digital asset issued on a blockchain, representing a unit of value. Tokens can represent currencies, governance rights, or other utilities in blockchain ecosystems.
Learn More

EVM (Ethereum Virtual Machine)

A runtime environment for smart contracts on the Ethereum blockchain and other compatible chains. It allows developers to deploy decentralized applications (dApps). EVM-compatible chains share Ethereum’s functionality.
Learn More

Starknet

A Layer-2 blockchain that operates on top of Ethereum and uses advanced cryptographic technology (zk-STARKs) to process transactions faster and cheaper while maintaining Ethereum's security.
Learn More

DeFi (Decentralized Finance)

A blockchain-based form of finance that removes intermediaries like banks by enabling users to interact directly with financial services such as lending, borrowing, and trading through smart contracts.
Learn More

On-Chain Data

Information stored directly on the blockchain, including wallet balances, transaction history, and smart contract interactions.
Learn More

Centralized Exchange (CEX)

A cryptocurrency trading platform operated by a centralized entity, such as Binance or Coinbase, where users deposit funds to trade. Unlike wallets, CEXs hold user assets on their behalf.
Learn More

Web3

A decentralized internet paradigm that leverages blockchain technology to enable peer-to-peer interactions without relying on centralized intermediaries. It emphasizes user ownership of data, decentralized applications (dApps), and programmable contracts (smart contracts) for transparency and autonomy.

Learn More