Software Architecture Document

MegaTon Wallet Extension

Team:

Arsen Mkrtchyan

04/15/2021

Revision History

Version	**Description**	**Name**	**Date**
v 1.1	Initial Version	Arsen Mkrtchyan	04/15/2021

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed architecture overview of the MegaTon Wallet extension, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document describes the various aspects of the application design that are considered to be architecturally significant. These elements and behaviors are fundamental to understanding this project as a whole.

1.3 Acronyms, and Abbreviations

WASM – Web Assembly

UML – Unified Modeling Language

1.4 References

Material UI GuideLines - https://material.io/archive/guidelines/

1.5 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use case view to the application
Section 3: describes the logical view to the application
Section 4: describes data storage, and data flows
Section 5: describe build and deployment of the application

2. Use-Case View

The purpose of use case view is to describe the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

2.1 Actors

Initial User – Initial MegaTon user, who did not set master password yet

User – MegaTon user, who set master password

2.2 Set Master Password

Initial User set master password and redirects to wallet list page**.** Master password is used to encrypt all the wallet secrets**.**

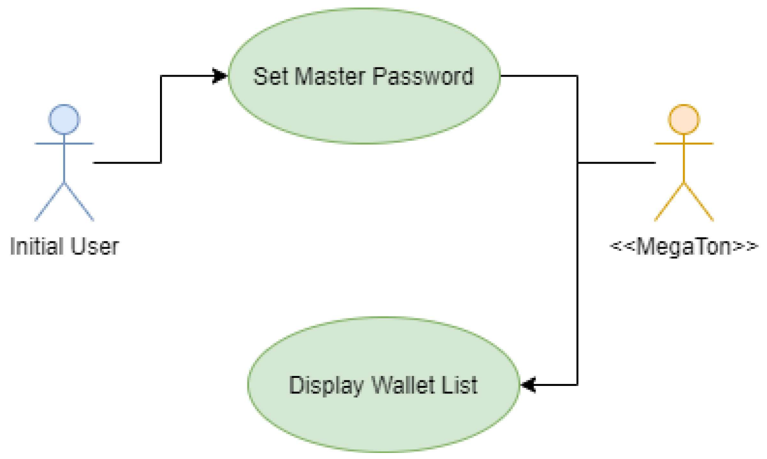


Figure 1: Set Master Password User Case Diagram

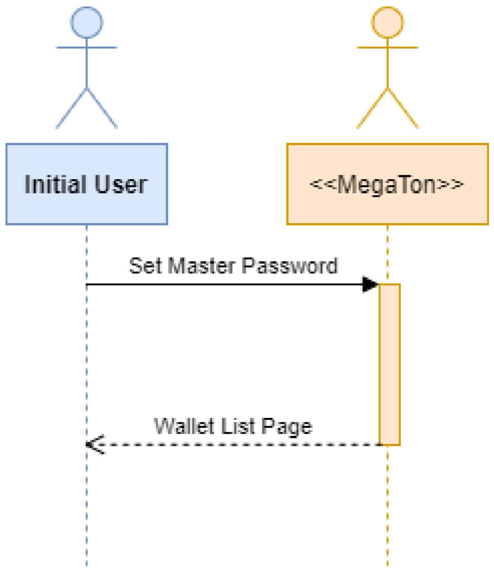


Figure 2: Set Master Password Sequence Diagram

2.3 Create New Wallet

User creates a new wallet. MegaTon validates if a user stores mnemonic words safely, before creating a new wallet.

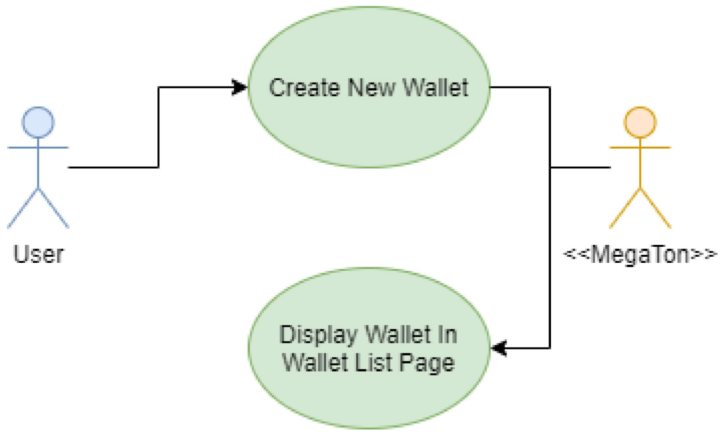


Figure 1: Create Wallet User Case Diagram

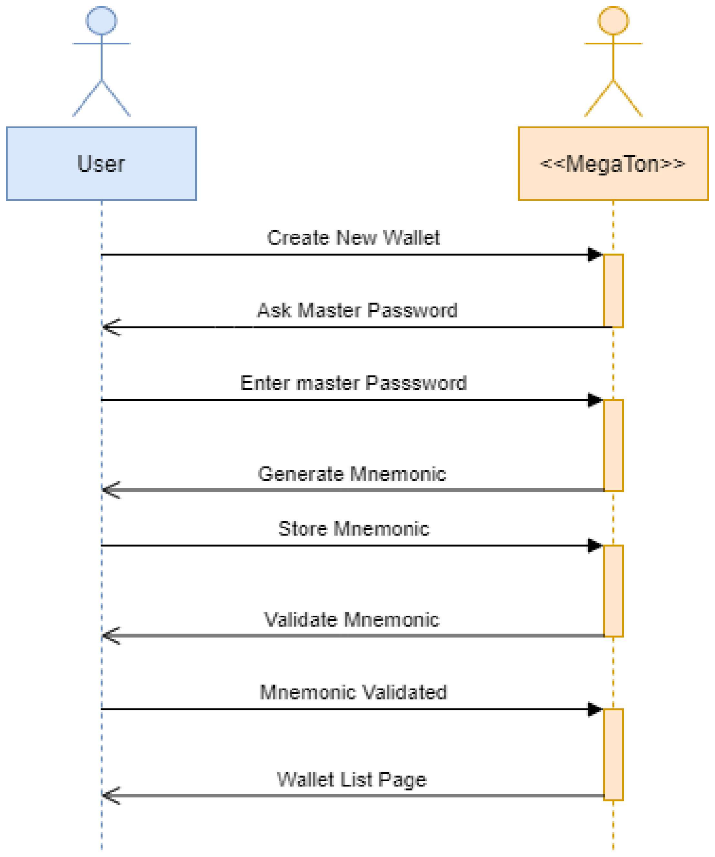


Figure 1: Create Wallet Sequence Diagram

2.4 Import Existing Wallet

User can import existing wallet using 12 or 24 length seed phrase, or wallet secret (private key)

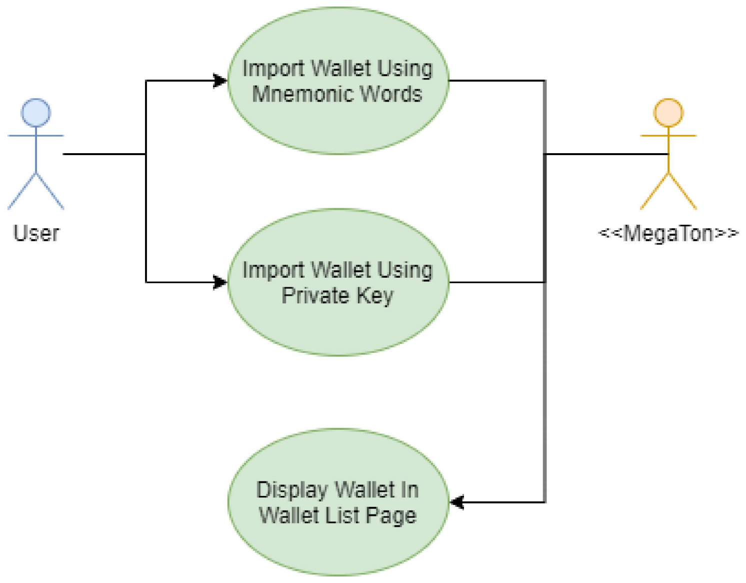

Import Wallet Using Mnemonic Words
Import Wallet Using Private Key
Display Wallet In Wallet List Page
User
<<MegaTon>>

Figure 1: Import Wallet User Case Diagram

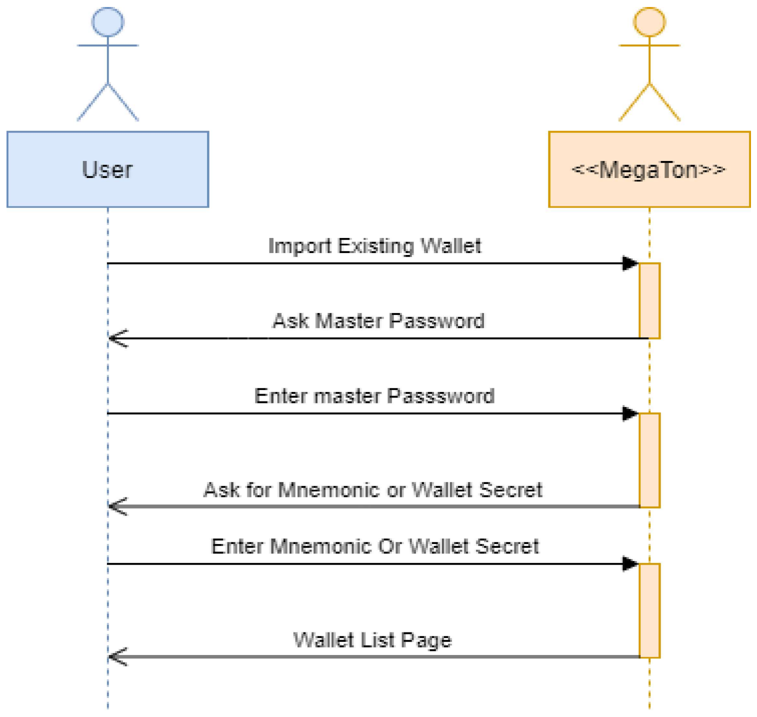


Figure 2: Import Wallet Sequence Diagram

2.5 Wallet Management

User check balances, copy address, explore transactions and delete owning wallets from the wallet list page.

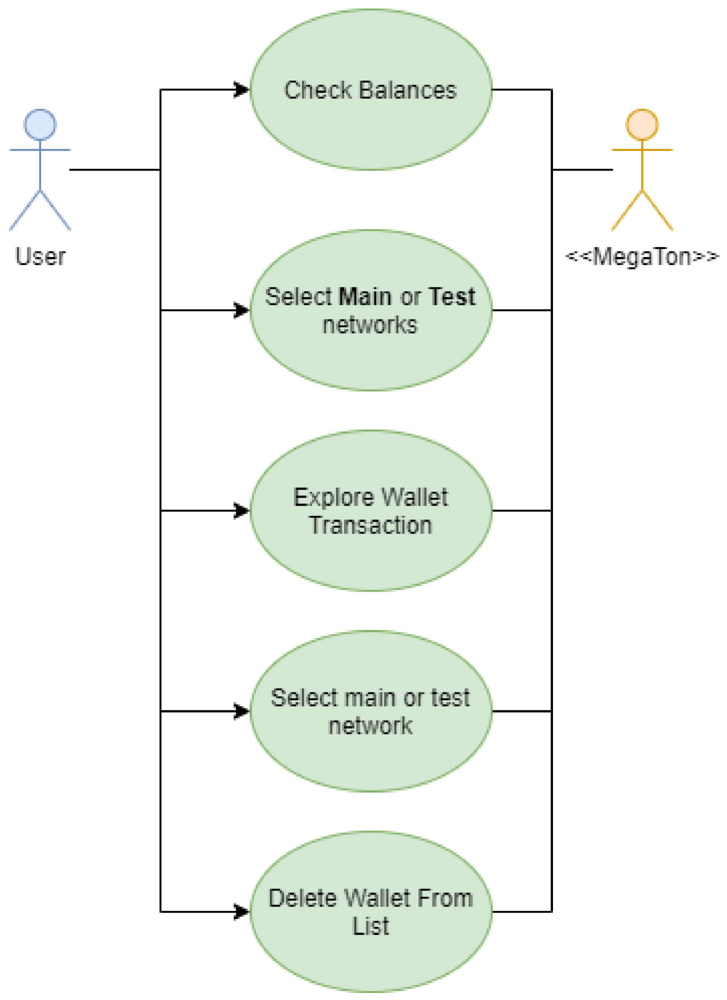

User
Check Balances
Select **Main** or **Test** networks
Explore Wallet Transaction
Select main or test network
Delete Wallet From List
<<MegaTon>>

Figure 1: Wallet Management Use Case Diagram

2.6 Send/Receive tokens

User deploy wallet contracts (if no contract on address), send/receive tokens in Main or Test networks.

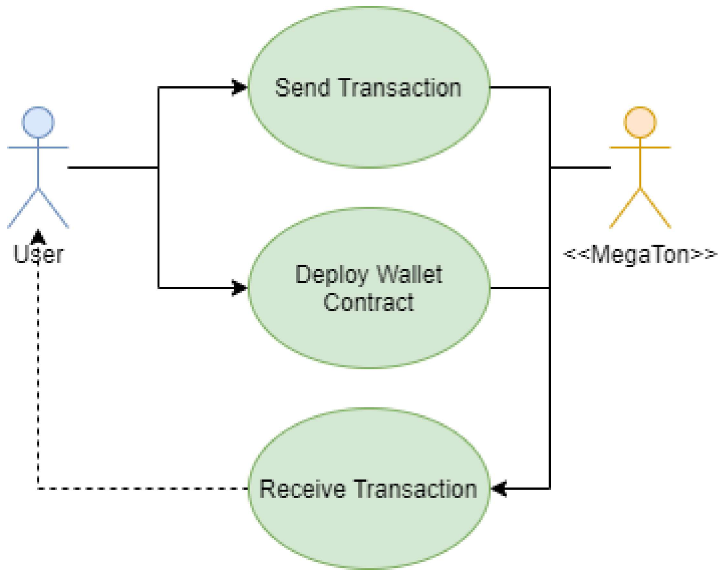


Figure 1: Transactions Use Case Diagram

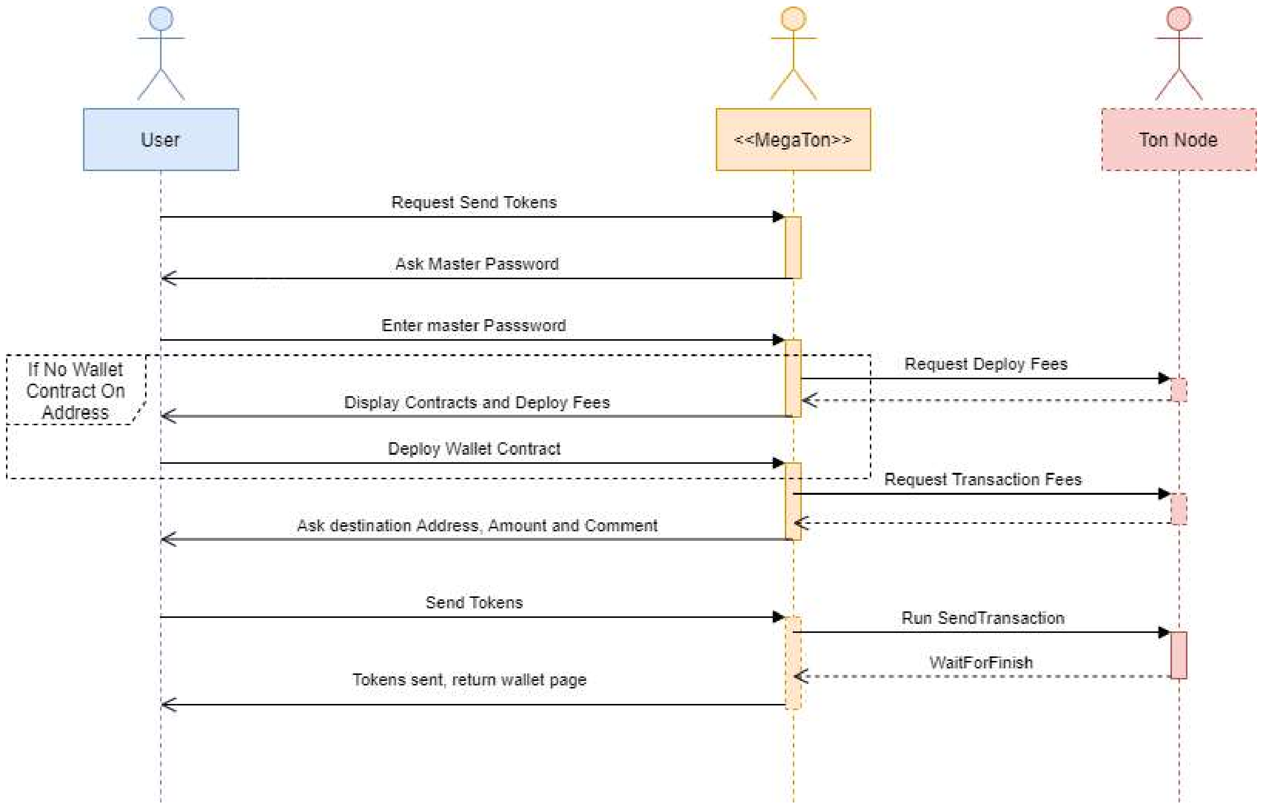


Figure 1: Transactions Sequence Diagram

2.7 User Interface

MegaTon user interface is built with Material UI guidelines in mind. All MegaTon forms are inheriting the default theme's style and pallets, which gives a possibility to create new themes easily.

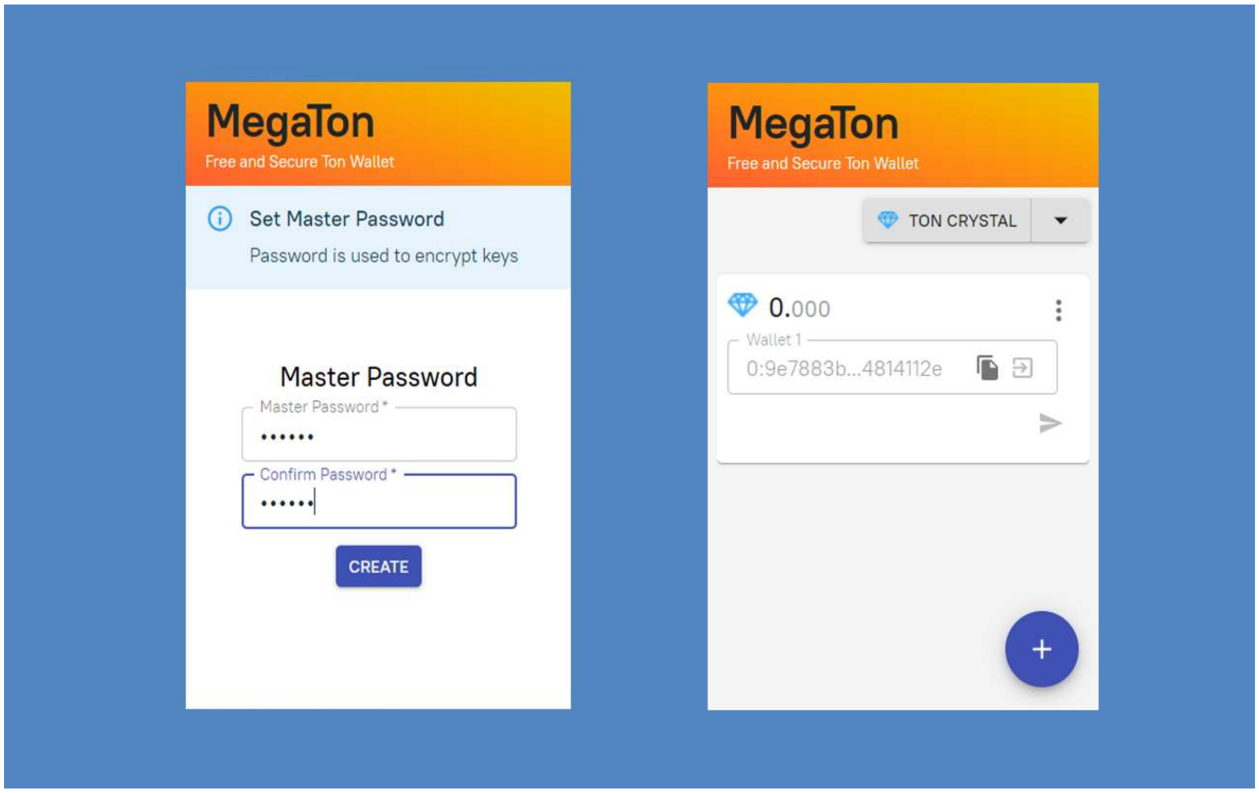
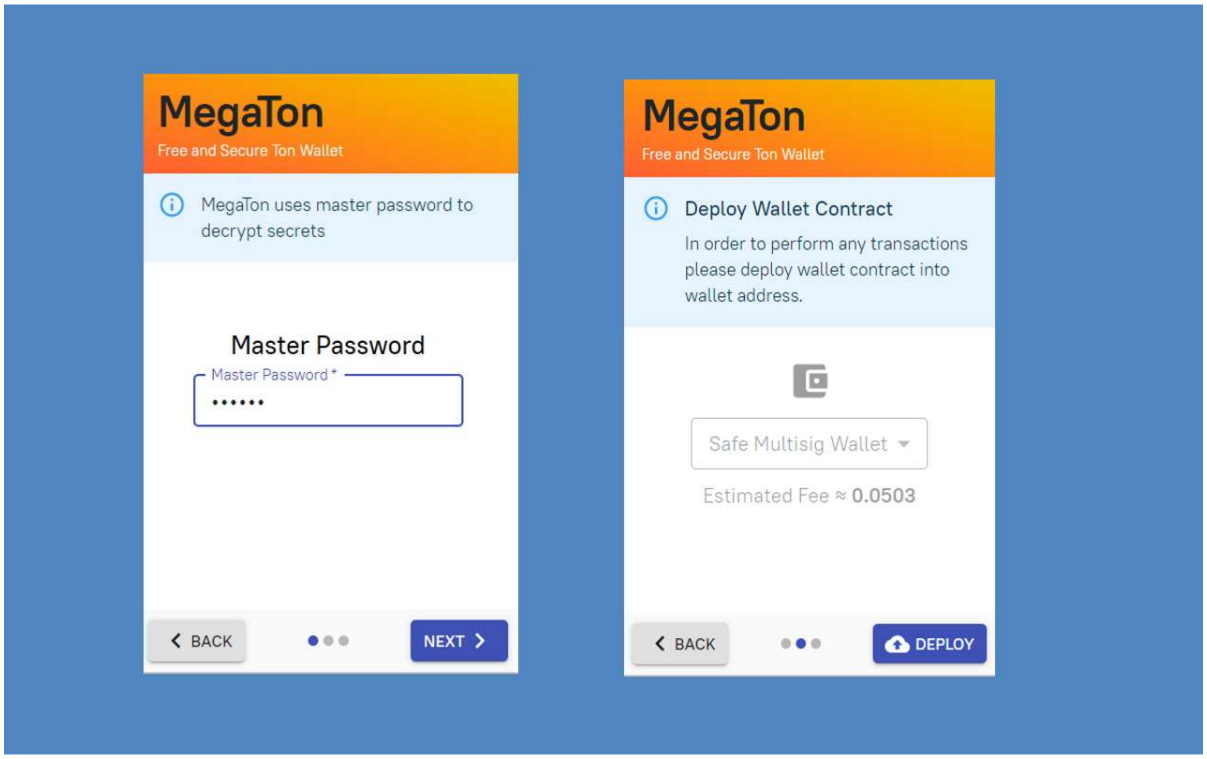


Figure 1. Set Master Password and Wallet List



MegaTon uses master password to decrypt secrets

Master Password

Master Password *: ••••••

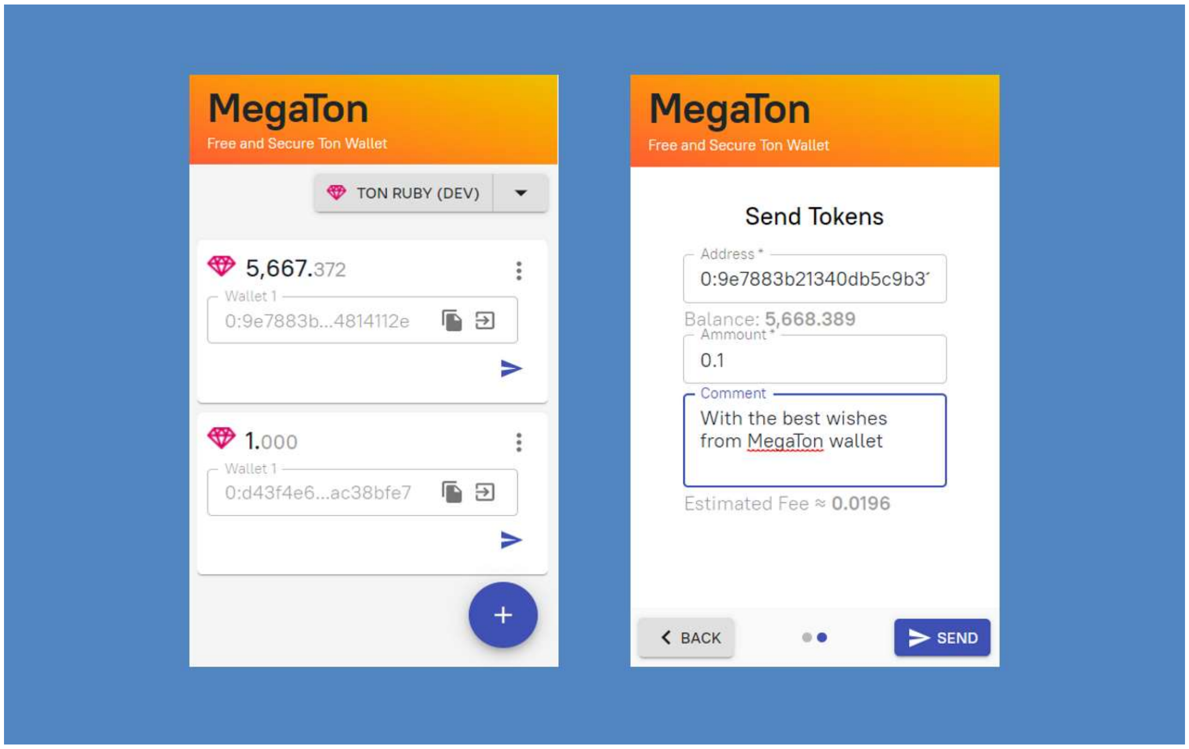
Deploy Wallet Contract

In order to perform any transactions please deploy wallet contract into wallet address.

Safe Multisig Wallet

Estimated Fee ≈ 0.0503

Figure 2. Ask Master Password and Deploy Wallet Contract



5,667.372

Wallet 1: 0:9e7883b...4814112e

1.000

Wallet 1: 0:d43f4e6...ac38bfe7

Send Tokens

Address *: 0:9e7883b21340db5c9b3

Balance: 5,668.389

Ammount *: 0.1

Comment: With the best wishes from MegaTon wallet

Estimated Fee ≈ 0.0196

Figure 3. Test Wallets and Send Tokens

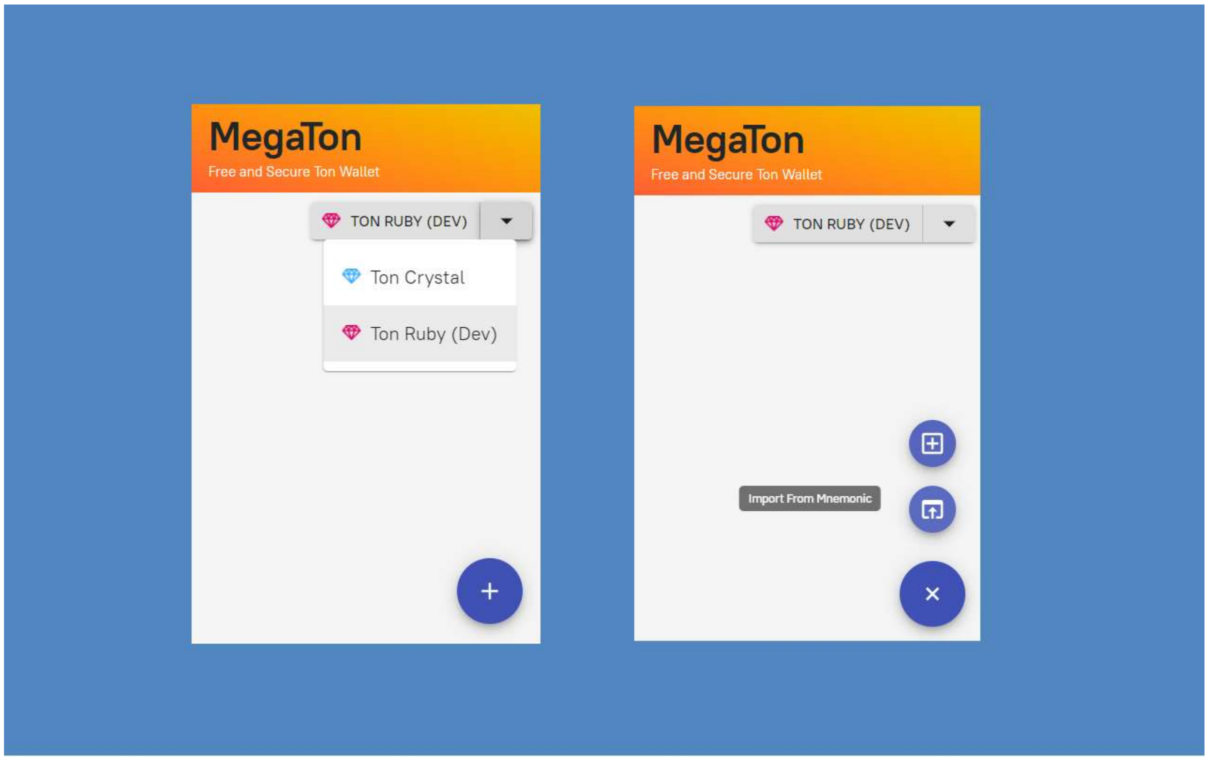


Figure 4. Select Network, Add / Import Wallet

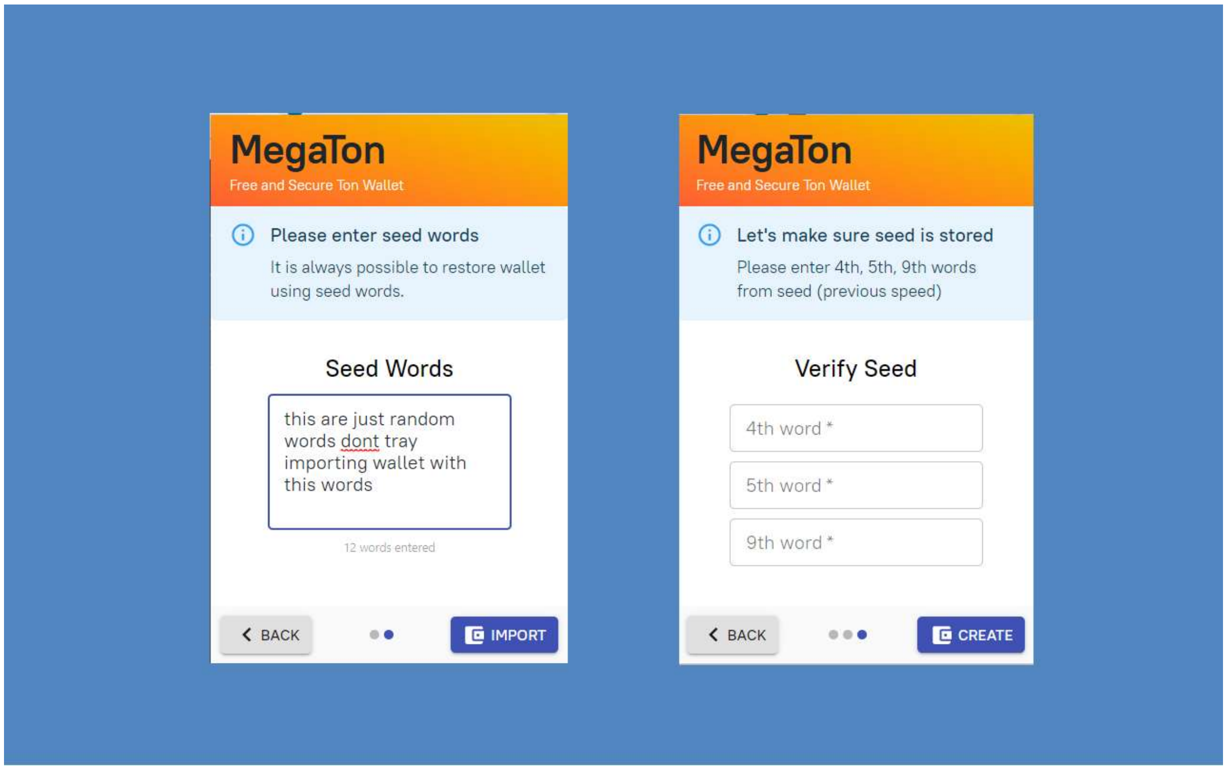

MegaTon
Free and Secure Ton Wallet
Please enter seed words
It is always possible to restore wallet using seed words.
Seed Words
this are just random words dont tray importing wallet with this words
12 words entered
BACK
IMPORT
Let's make sure seed is stored
Please enter 4th, 5th, 9th words from seed (previous speed)
Verify Seed
4th word *
5th word *
9th word *
BACK
CREATE

Figure 5. Enter Seed and Verify Seed

3. Logical View

3.1 Overview

The purpose of logical view is to describe the most important classes, their organization in packages, and the organization into layers. Also describes the most important use-case realizations.

3.2 Big Picture

MegaTon connects directly to free ton node using the official JavaScript Free TON SDK.

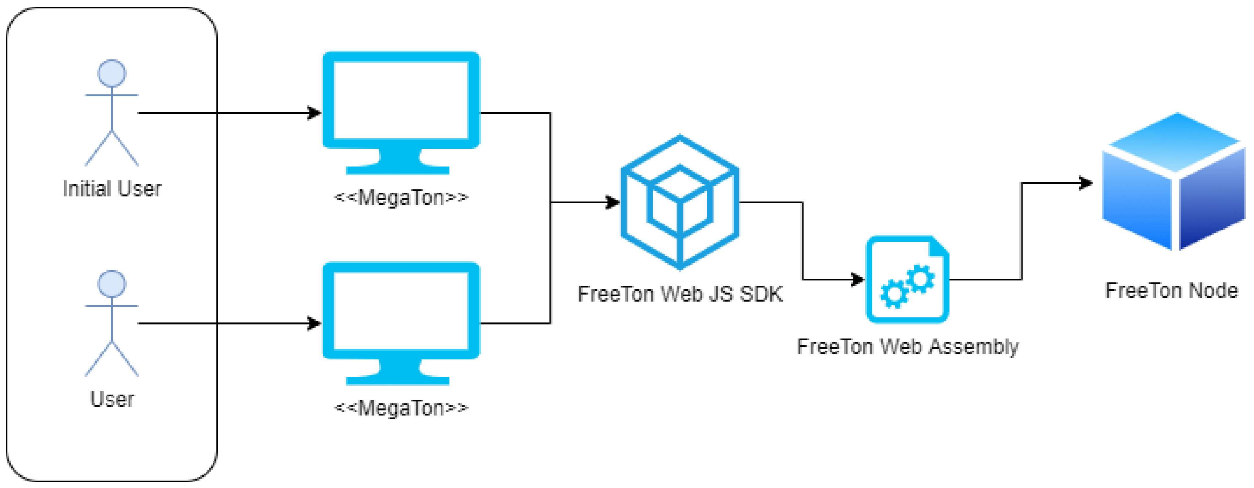


Figure 1. High level view

3.3 Layering

MegaTon application consists of 3 logical Layers. **Presentation, Wallet API, Network and other APIs.**

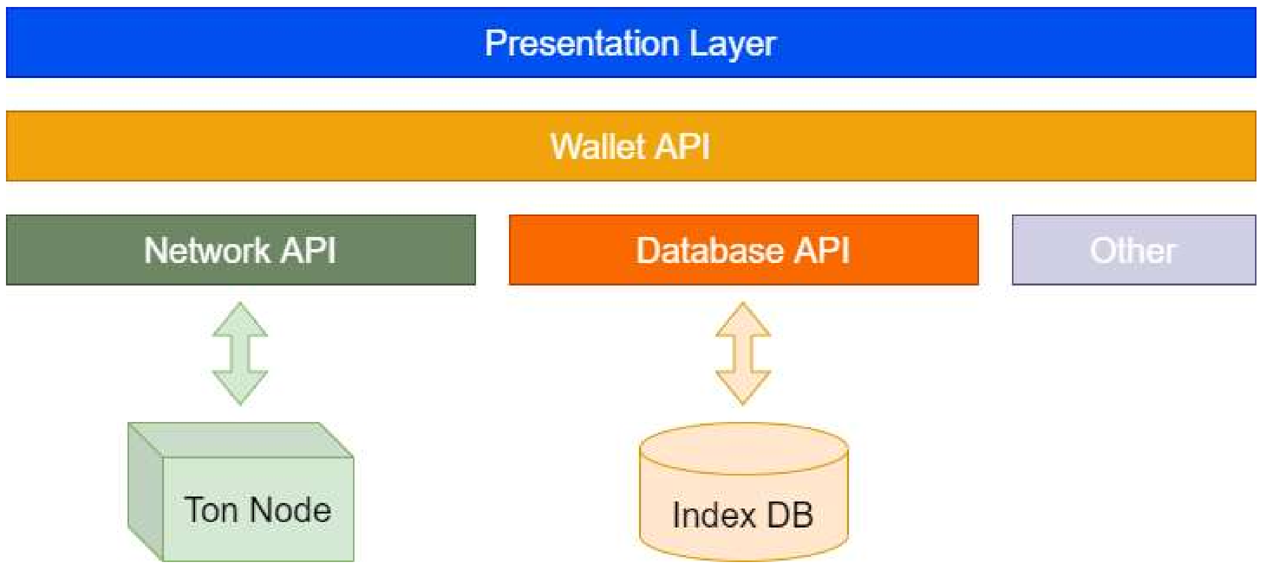


Figure 1. Application layers

Presentation Layer represents the forms and controls displayed on the screen. All the forms are thin and solve only **View** related tasks. All communication between presentation layer and other layers, go throw Wallet API.

MegaTon forms are built using React Material UI, and best material ui practices.

Wallet API is the business logic layer of the application. It provides API to all wallet management methods(example dreateWallet, deployContract etc.).

All the methods that store/retrieve mnemonic words or private keys are using a master password to encrypt / decrypt these secrets. Below diagrams show the algorithms used to encrypt/decrypt secrets.

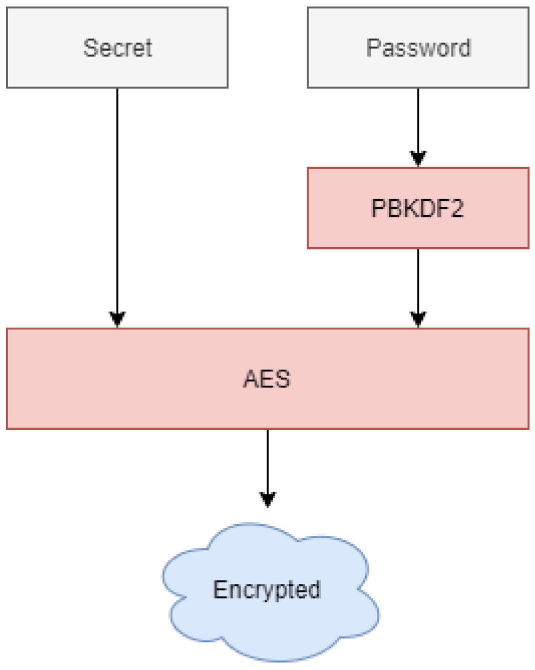


Figure 2 Encrypt algorithm

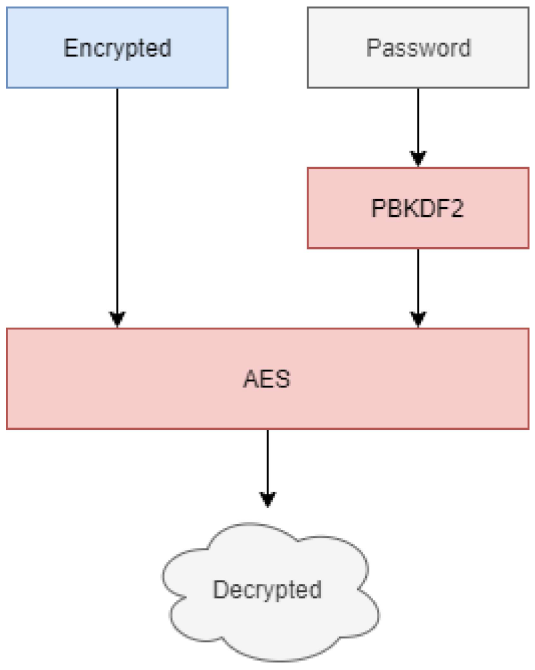


Figure 3 Decrypt algorithm

Wallet API does not reference any specific network, or storage providers, instead it aggregates abstractions, which make it possible to easily inject other implementations.

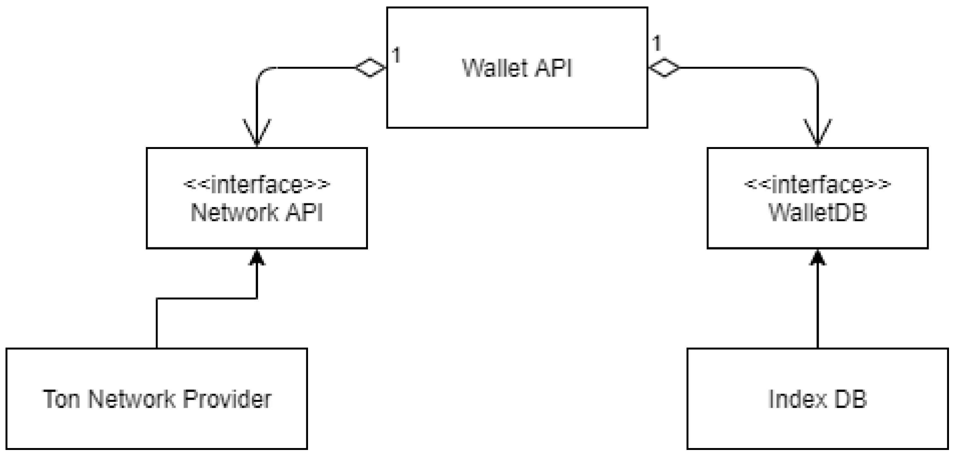


Figure 4 Wallet API relationship

4. Data View

4.1 Database Design

Current MegaTon implementation is storing data into IndexDB, however WalletAPI abstract this reference, so it could be easily replaced by any other storage.

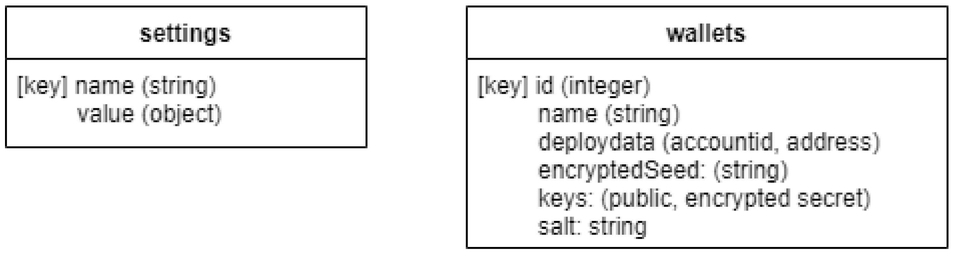


Figure 1 MegaTon DB diagram

The **settings** table is a key-value storage for MegaTon settings. It stores master password hash (with salt), network selection and other user preferences.

The **wallets** table contains all the wallets that users have created or imported into the application. All the secrets are encrypted using PBKDF2 + AES algorithms before storing.

5. Build and Deployment

5.1 Overview

MegaTon application is built using react create-react-app react application builder. All the create-react-app scripts are valid.

5.2 Available Scripts

You'll need to have Node 10.16.0 or later version on your local development machine.

Run the following command in the project directory to start the app in the development mode. Open http://localhost:3000 to view it in the browser.

```
npm start
```

Run the following command to build the app for production. All the artifacts are bundled together, and put into **/build** subfolder

```
npm run build
```

5.3 Deploying as a browser extension

After running the build command (npm run build) all the artifacts will be outputted int **/build** subfolder.

1. In the browser, open **chrome://extensions** url
2. Enable development mode in the opening page.

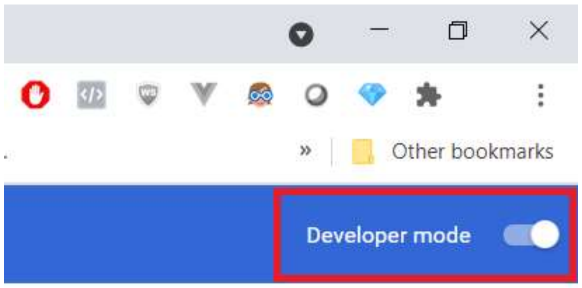


Figure 1. Enable development mode in Chrome (top right corner)

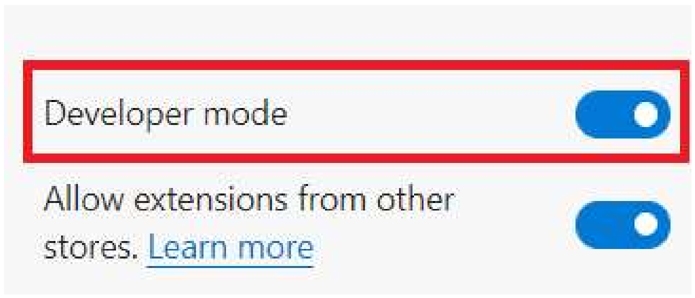


Figure 1. Enable development mode in Edge (bottom left corner)

3. Click the **Load Unpacked** button from the extensions page, and select **/build** subfolder with artifacts.
4. Extension will be added into browser extension list. Happy using.

5.3 Other Deployments

MegaTon is a single page web application, and can be deployed as browser extension, mobile application or a web application.