SSI stage 4 submission

What is RadianceTeam SSI 4 concept

By Stage 2 and Stage 3 of Everscale SSI Framework our team created the fundamentals for the SSI infrastructure (DID component and SDK).

Also our DID-Everscale method was approved and published by W3C as a reference implementation:
https://w3c.github.io/did-spec-registries/#did-methods

approved and added in Universal Resolver driver:
https://github.com/decentralized-identity/universal-resolver#drivers

Our team for the #48 Everscale Self-Sovereign Identity Framework (Stage 4) contest developed core instruments for Verifiable Credentials (VC) issuance, storage, and verification with justified usage of the Everscale blockchain infrastructure. Also our submission included smart-contracts deployed in Everscale Mainnet, prepared tools for developers and web interfaces with demo applications (Issuer and Holder Interface and Verifier Interface).

We created the VC module which consists of triangle SSI Concept (Issuer, Holder, Verifier). This system allows verifiers not only to authorize its' users but also create the system of attribute verification.

SSI Mobile Application makes it easy and affordable for users to log in through SSI. Users can pass authorization just like they use Google Auth – by scanning the QR code using a mobile device, the user has the opportunity to log in to the site of any organization (such as finance, education, gaming, art industry, and many others). Also Mobile Application supports the SSI system of attribute verification controlled by the end-user.

Our project scope

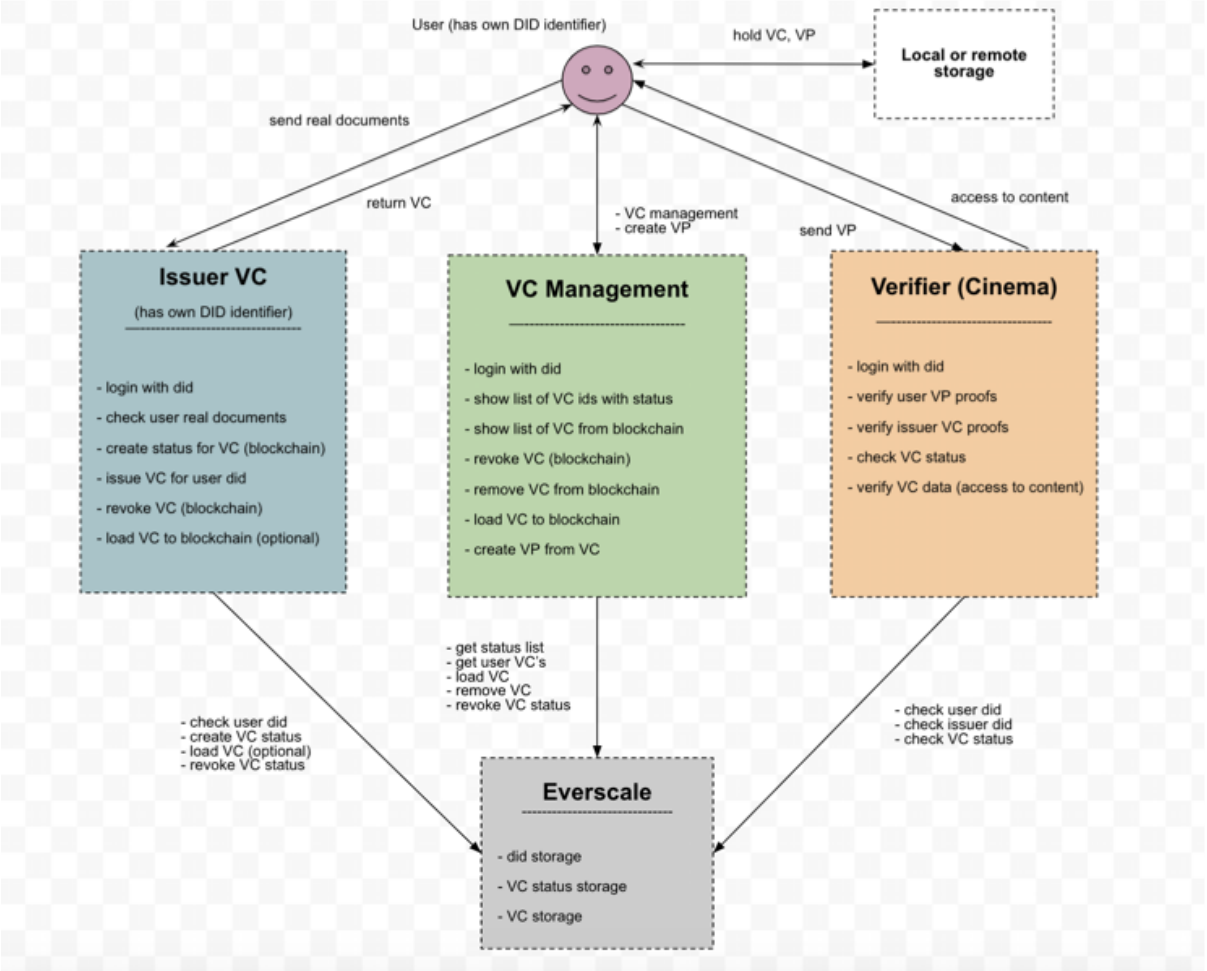
Our primary objective was to build up basic Verifiable Credentials (VCs) infrastructure atop the Everscale blockchain and enable the external developers to deploy primitive SSI systems using ready-made components.

What we have done:

- SDK with detailed documentation (code)
- Everscale Smart-contracts (code)
- Frontend + Backend Issuer (codeBackend, codeFront, webApp)
- Frontend Holder (codeFront, webApp)
- Frontend + Backend Verifier (codeBackend)
- Mobile App (code, demo application)

All source codes for above components are licensed under Apache 2.0.

Interaction diagram of SSI Trust Triangle concept (Issuer, Holder, Verifier)

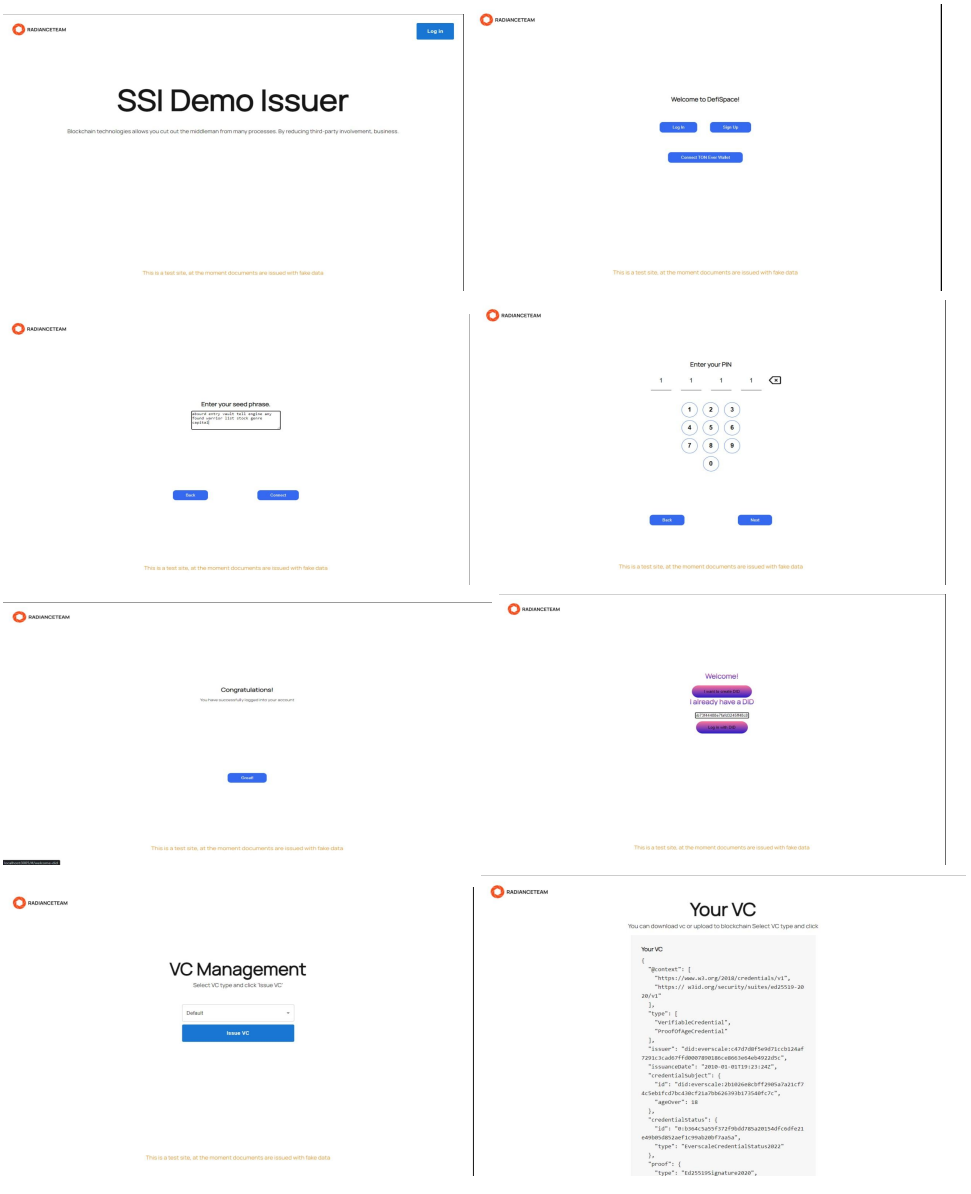


The Issuer Verifiable Credentials

The Issuer - issuing site that checks the user's documents and issues Verifiable Credentials based on them.

- The user has own DID and is authenticated on the issuer's website using DID.
- The issuer has its own DID that points to the VC that the Verifier site can find out who issued the VC.
- The issuer specifies the user's DID in the VC that the Verifier can check to whom the VCs were issued.
- The issuer creates a status for the VC in the smart contract, the address of the contract is entered into the VC so that the Verifier can check the status of the VC.
- The issuer or user can deactivate the VC status.

Web Interface



Web Interface

https://ssi4-issuer.vercel.app/#/

Description and documentation of the code

Backend

Frontend

The repository contains instructions for launching the web interface locally.

The Holder Verifiable Credentials

Holder is a site for viewing and managing VCs.

- The user can get a list of VCs previously issued to him with DID in the form of VC statuses
- The user can deactivate the status of any VC issued to him.
- The user can upload the previously received VC to the blockchain.
- The user can find a list of previously loaded VCs.
- The user can delete previously loaded VCs.
- The user can create a Verifiable Presentation based on a Verifiable Credential using did.
- VPs are required for verification on the Verifier site.

Web Interface

https://ssi4-manager.vercel.app/#/

Description of realization and documentation of the code is here

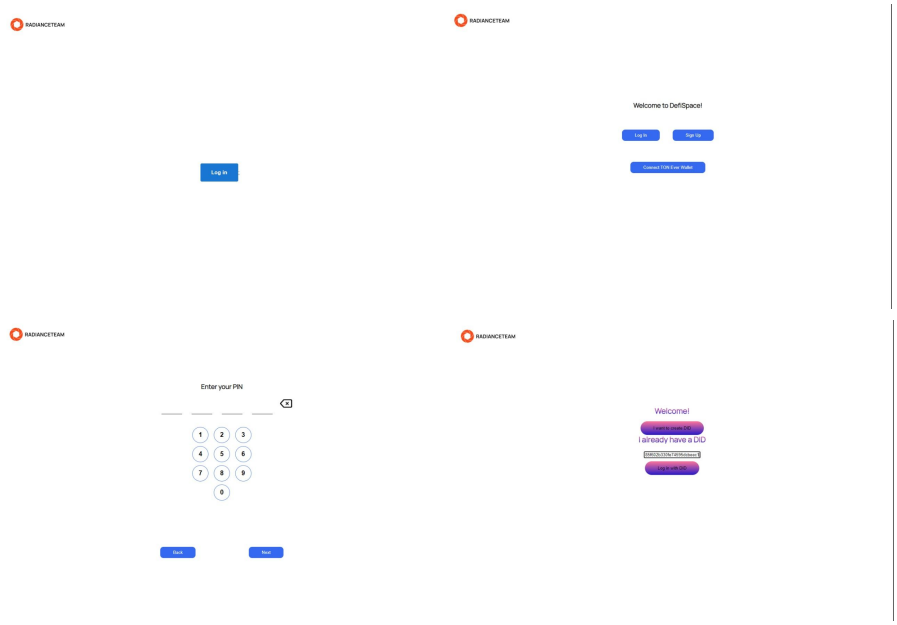
The repository contains instructions for launching the web interface locally.

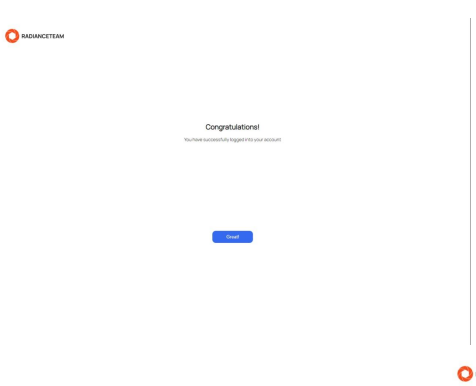
The Verifier ensures

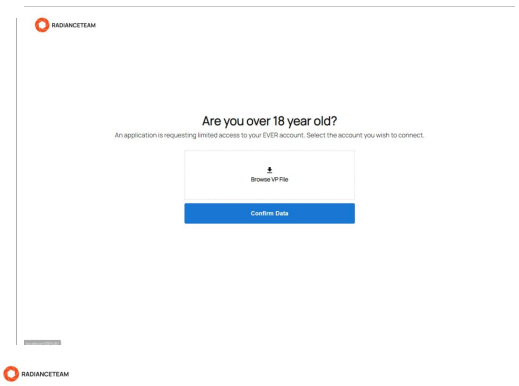
This site is a demo application with SSI authentication for online cinema. This will make it possible to show 6+, 12+, 18+, etc. rated movies only to respective audiences. On this platform users can register and log in as a user using an Everscale wallet by SSI technology. Also we implemented user authorization in the showcase using a QR code. By scanning the QR code using a mobile device, the user has the opportunity to log in to the site.

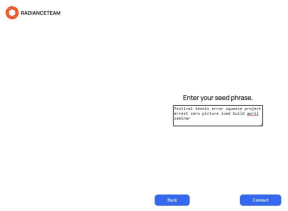
- Site Verifier - receives from the user VP, which stores the assertions from the Issuer necessary for verification.
- Checks the user's DID with authentication.
- Accepts the user's VP and checks the integrity of the VP.
- Checks the integrity of the VCs that are found inside the VP.
- Checks the issuer's DID to see if it can be trusted.
- Checks the status of the VC by referring to the contract.
- Validates the data issued by the issuer (in this case, age) to allow access to the site.

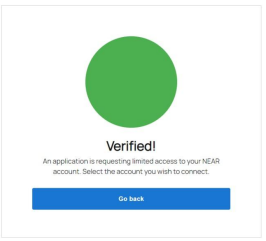
Web Interface











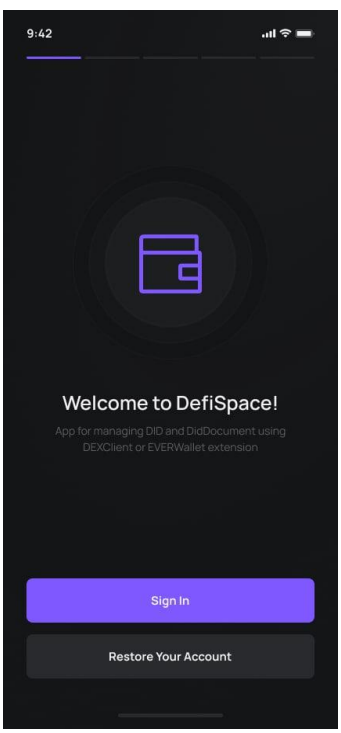
Description and documentation of the code

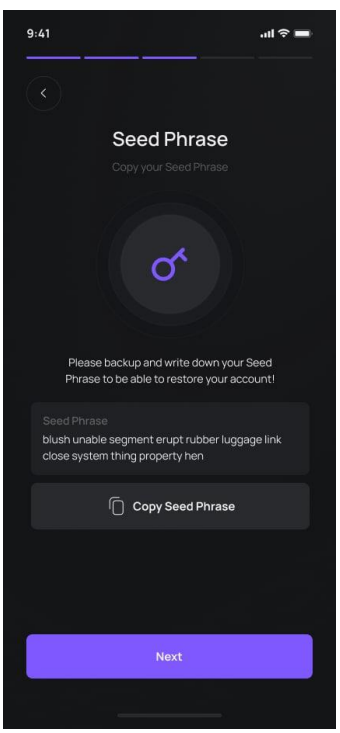
https://git.defispace.com/ssi-4/everscale-vc-backend-verifier

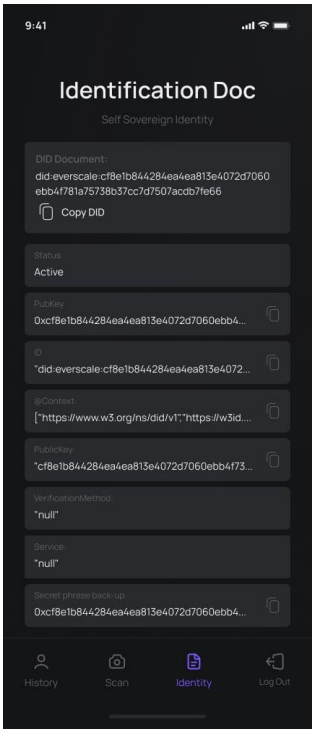
The repositories have instructions for running the demo locally.

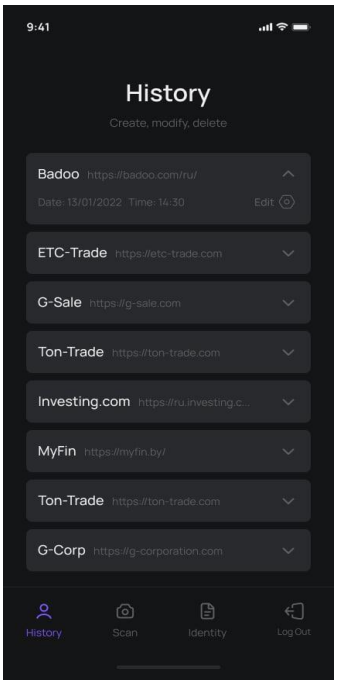
Mobile App SSI

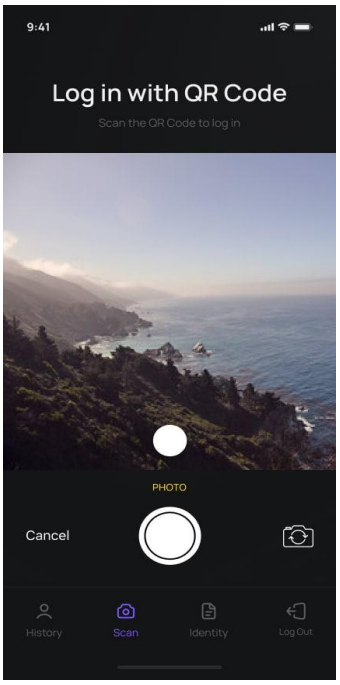
Screen-shots

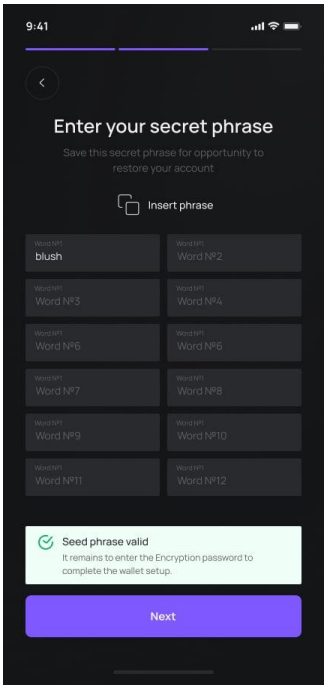












The code and description - https://git.defispace.com/ssi-4/everscal-ssi-mobile-app

How to use Mobile App SSI

Demo Interface

https://www.youtube.com/watch?v=-qQWggS-O_8

Description

- Step 1 User access a website that supports SSI
- Step 2 For authorization on the site, the user is shown a QR code
- Step 3 The user scans the QR code using their mobile application
- Step 4 After that, the user becomes authorized on the site

Additional management of VC (Verifiable Credentials) is implemented on the SSI Manager Site (Holder Verifiable Credentials).

Smart-contracts

Instruments for deployment is here

SDK Specification

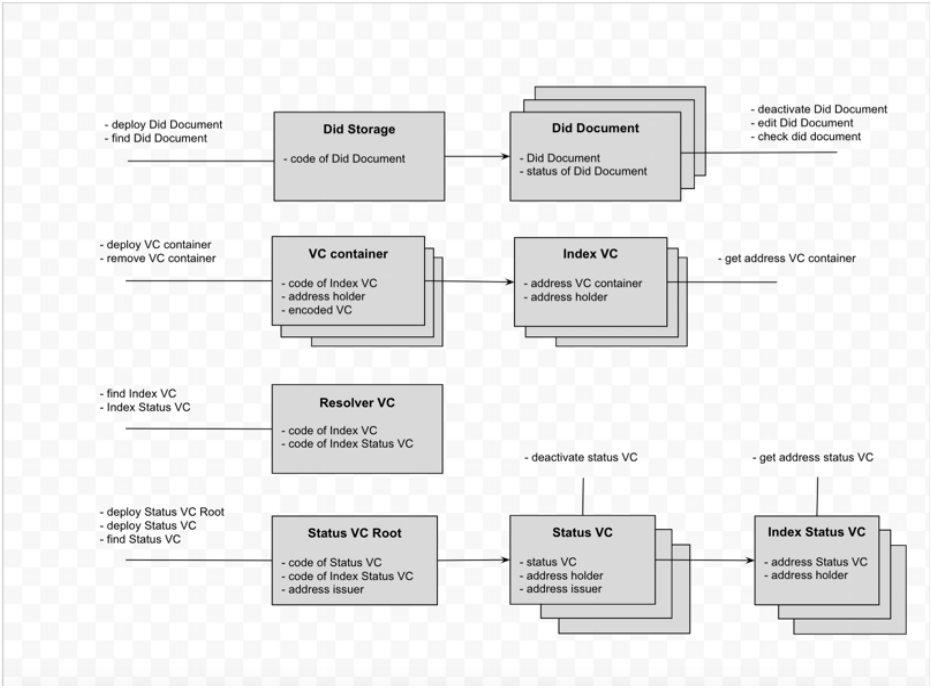
The SDK specification is here

SDK Documentation

The SDK documentation is here

Technical Overview

General architecture



Our stack

Backend

- Node.js + TypeScript
- NestJs
- PostgreSQL (for demo showcase only)
- TypeORM

Frontend

- HTML+CSS+JavaScript(JS)
- React as JS framework
- Tonclient as JS framework
- EverWallet Extension

Smart contract

- Solidity language for smart contracts

Sdk

- Node.js
- Tonclient as JS framework
- crypto-js
- jsonld
- noble-ed25519

Mobile App

Front-end:

- React Native,
- React, JS,
- TonClient as JS Framework

Back-end:

- Node.js,
- TypeScript,
- NestJS

Contacts

Wallet:

0:835ebc5dc3b3370b77f15ecf4e62add730f67ef5605c9b2c976e38c0ec6ce3d6

Project credits:

Snezhana Kolesnik, Roman Omelusik, Pavel Soldatov, Evgeniy Harlamov, Anton Zadorozhniy, Dmitry Samorodkin.

Contacts:

- Снежана Колесник **(**https://t.me/skaisy)
- Роман Омелюсик (https://t.me/amel007)
- Дмитрий Самородкин (https://t.me/dnugget)