

**In The Name Of GOD**

# **DEVVIDs PROJECT REPORT**

Prepared by  
Mohammad Amin Baghbanbashi  
Saina Daneshmandjahromi  
Zahra Mohammad Pour  
Mohsen Pakzad

December 31, 2021

# Table of Content

<b>Methodology</b>	<b>2</b>
Why Scrum?	2
Remote Pair Programming	2
Project Management	2
<b>Analytics &amp; Requirements Elicitation</b>	<b>3</b>
Requirement Elicitation	3
Similar Products	3
Research Topics	3
Concepts of decentralized identifiers	3
Engagement With Product Owner	4
Backlog items	9
Back-End Backlogs	9
Front-End Backlogs	9
Deployment Backlogs	9
<b>Architectural Design and System Model</b>	<b>9</b>
Decentralized Identifier Mechanism	9
High-Level Issuer Architecture	10
High-Level Holder Architecture	11
High-Level Verifier Architecture	12
<b>Implementation</b>	<b>15</b>
Back-end Technologies	15
Front-end Technologies	15
DevOps	16
<b>Testing</b>	<b>16</b>
Issuer tests	16
Holder tests	17
Verifier tests	17
<b>References</b>	<b>19</b>

# Methodology

## Why Scrum?

- A. Requirements are not clearly defined
- B. Our client's culture is open to innovation and adapts to change
- C. New changes can be implemented at very little cost
- D. Our team has self-management skills

We have deployed Agile software development methodology to center around time-boxed project cycles known as sprints. We have divided our project into multiple "user stories" for different types of users. As such, the sprint consists of a significantly smaller number of features than a waterfall project. Limiting the features in this manner provided us with a more manageable product development and release cycle.

Our team consists of developers, the product owner, and the Scrum master. During a sprint, we participated in daily stand up meetings where we discussed the progress of our project. A link to meeting reports is available at the appendix of this report.

## Remote Pair Programming

Due to the fact that our team members had different levels of expertises in blockchain development and solidity language, we have used **Remote pair programming practice** from **Extreme Programming (XP)** which is a part of the agile software development model. In order to make this work, We used screen sharing tools and programmed parts of the project as pair programmers.

Even though there were some **challenges** for making this practice happen since we all had packed schedules and it was difficult to find the optimal time window, this decision brought us several **benefits**, for instance during development of our solidity contract, we all shared our opinions about structures of **VC** and **VP**. Furthermore, pair programming helped us to find edge cases and bugs faster than usual since more than one person was thinking about the possibilities that could occur.

## Project Management

**Version Control** Since we were a group of 4, we used git as a version control, and github as a git provider to collaborate on this project.

**License** For our project, we used GPL3 as our license, as the product owner recommended it. This license is used for most software and other practical works designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended

to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users.

# Analytics & Requirements Elicitation

**Abstract** “For today’s digital lives, digital identities are used across every app, service, and device. Decentralized Identifiers (DIDs) are expected to empower users to have greater control over their data. For DIDs, blockchain is considered to be a platform to register identifiers that users can use to identify themselves. This project aims to develop a prototype of software developers’ DIDs on blockchain to enable them to present verifiable claims of the past activities in the belonging software development projects.”[\[1\]](#)

Major requirements of the system :

- Issue credentials by issuer
- Verify Identifiers and use schemas by issuer
- Register Identifiers and use schemas by holder
- Verify identifiers and schemas by verifier

## Requirement Elicitation

### - Similar Products

- The Identity Overlay Network (ION) is a DID Method implementation using the Sidetree protocol atop Bitcoin [\[2\]](#)
- A code sample demonstrating how to use Azure Active Directory's preview functionality to issue and consume verifiable credentials.[\[3\]](#)

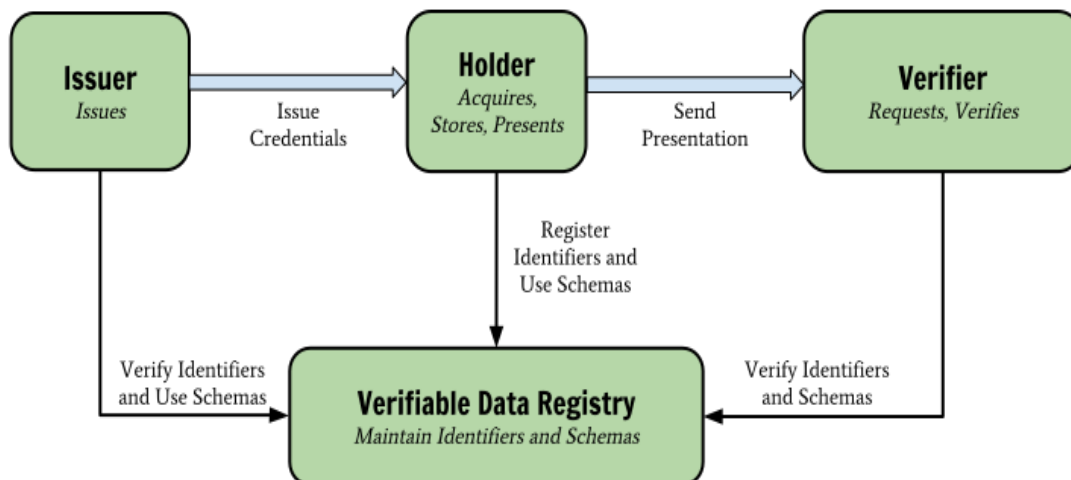
### - Research Topics

- Types of credentials and authenticators[\[4\]](#)
- Introduction to Azure Active Directory Verifiable Credentials [\[5\]](#)
- Verifiable Credentials Data Model [\[6\]](#)

### - Concepts of decentralized identifiers

- A credential is the thing that a person presents—in person or remotely—to say “this is who I am”.

- Physical: This must be physically carried by a person in order to use them
- Digital: They are machine readable and therefore can be used in a digital environment
- Verifiable Credentials (VCs) are an open standard for digital credentials
- Holder is a role an entity might perform by possessing one or more verifiable credentials and generating verifiable presentations from them
- Issuer is a role an entity performs by asserting claims about one or more subjects, creating a verifiable credential from these claims, and transmitting the verifiable credential to a holder
- Verifier is a role an entity performs by receiving one or more verifiable credentials, optionally inside a verifiable presentation, for processing



### - Engagement With Product Owner

We had to contact our client several times to gain a better understanding of the requirements. The following emails were exchanged during different phases of our development.

# Final Project Report

---



**Mohsen Pakzad** <mohsen137853@gmail.com> Sun, Oct 17, 2021, 7:29 PM



to hata ▾

Dear Mr.Hata

We are a team from Shiraz University and we chose DevDIDs project among the recommended projects for our Software Engineering course as we found its documentation really interesting.

We are extremely excited to collaborate on this project.

We found out from our research that the best platform for this project is ethereum.

Ethereum smart contracts seem to be a reasonable fit if it is alright with you.

We also would like to know what you expect to be the prototype.

Thank you for your consideration.

Best regards.



**Hideaki Hata** <hata@is.naist.jp>

Oct 18, 2021, 12:43 PM



to me ▾

Hi Mohsen,

Note that I have never implemented this. I don't know what you are capable of and how much effort you can put into it, so please decide for yourselves how much to implement.

Ethereum smart contracts sound good to me.

I think it might also be good for you to check this one as it is very well prepared.

<https://docs.microsoft.com/en-us/azure/active-directory/verifiable-credentials/decentralized-identifier-overview>

Best wishes,

Hideaki Hata



# Final Project Report

---



**Mohsen Pakzad** <mohsen137853@gmail.com>

to Hideaki ▾

Fri, Nov 5, 2021, 10:55 PM



Dear Mr.Hata,

I hope all is well with you.

We have read the Microsoft document and we have few questions about DevDID's project:

1. What exactly is "issuer" ? Is issuer an API or a person? And should we implement user interface for issuer too?
2. Do you think using ION project (<https://github.com/decentralized-identity/ion> ) would help?
3. What do you think is acceptable prototype for project? Is UI implemented using Adobe XD acceptable for the prototype?

Also,

Here's a list of requirements and back logs we wrote, what is your opinion about these items? would you like to add or edit some of these?

1. Holder should be able to request for vc to issuer
2. Issuer should be able to issue a Verifiable credential for specific holder
3. Holder must be able to present the vc to verifiers
4. Verifier should be able to verify holders presentation
5. Holder must have list of VCs that have been issued for him/her

Thank you for your support.

Best regards.

\*\*\*



**Hideaki Hata** <hata@is.naist.jp>

to me ▾

Mon, Nov 15, 2021, 6:47 PM



Hi Mohsen,

This sample scenario may be helpful for you.

<https://docs.microsoft.com/en-us/azure/active-directory/verifiable-credentials/decentralized-identifier-overview#a-sample-scenario>

You can use any of the products and libraries like ION if there is something available.

I think this project is quite challenging. Not every feature needs to be perfect, and I think any prototype is OK. I think you need to be clear about what requirements are important to you and make sure that they are met.

Best wishes,  
Hideaki Hata

\*\*\*

# Final Project Report

---



**Mohsen Pakzad** <mohsen137853@gmail.com>

Wed, Dec 1, 2021, 3:45 AM



to Hideaki ▾

Dear Dr.Hata

I hope all is well with you.

We are about to enter the implementation phase, would you please provide us with some real world test cases?

Anything that helps us know how the system should behave in each scenario is helpful.



**Hideaki Hata** <hata@is.naist.jp>

Wed, Dec 1, 2021, 9:58 AM



to me ▾

Hi Mohsen,

You can check if the system works properly from the perspective of the user, the issuer, and the verifier.

<https://docs.microsoft.com/en-us/azure/active-directory/verifiable-credentials/decentralized-identifier-overview#a-sample-scenario>

Best,

Hideaki





# Final Project Report

---



**Mohsen Pakzad** <mohsen137853@gmail.com>

Sat, Jan 1, 5:05 AM



to Hideaki ▾

Dear Dr. Hata,  
We hope this email finds you well.

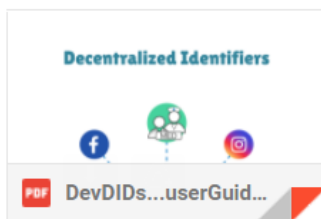
We are glad to let you know we have implemented the initial version of devDID's project, and we would appreciate that if you could test this product and give us your valuable feedback.

You can find below the URL of our product as a website:  
[dev-dids.netlify.app/](https://dev-dids.netlify.app/)

Also, you can find attached a simple user guide to give brief information about how the product works.

We look forward to hearing your thoughts from you.

Kind Regards,  
DevDIDs Team



Re: SCORE21



January 3, 2022 3:10 AM

Hideaki Hata

Details

Hi Mohsen,

I think it's a great product. It is very well made.

Can I show it to some developers I know?

Best regards,  
Hideaki Hata

### Backlog items

Based on the previous discussions, we divided our project into separate backlogs as detailed below.

#### - Back-End Backlogs

- Set up the Verifiable Credentials service
- Issue A Verifiable Credentials
- Issuer assign a VC to a user
- Verify a Verifiable Credential
- VP
  - The holder signs a verifiable presentation (VP) with their DID and sends it to the verifier.
  - The verifier then validates the credential by matching with the public key placed in the DPKI.
- The verifier must be able to generate a verified solution for a VP and approve the VP.
- Exciting Requirements
  - Revoke a previously issued verifiable credential

#### - Front-End Backlogs

- A page for Issuer to assign a VC to a user
- A verify page so that the issuer of the credential attests that the proof the user provided is accurate and creates a verifiable credential signed with their DID and the user's DID is the subject.
- A page for the holder to generate verifiable presentation for its verifiable credentials.
- Verifier's front-end
- Issuer's front-end

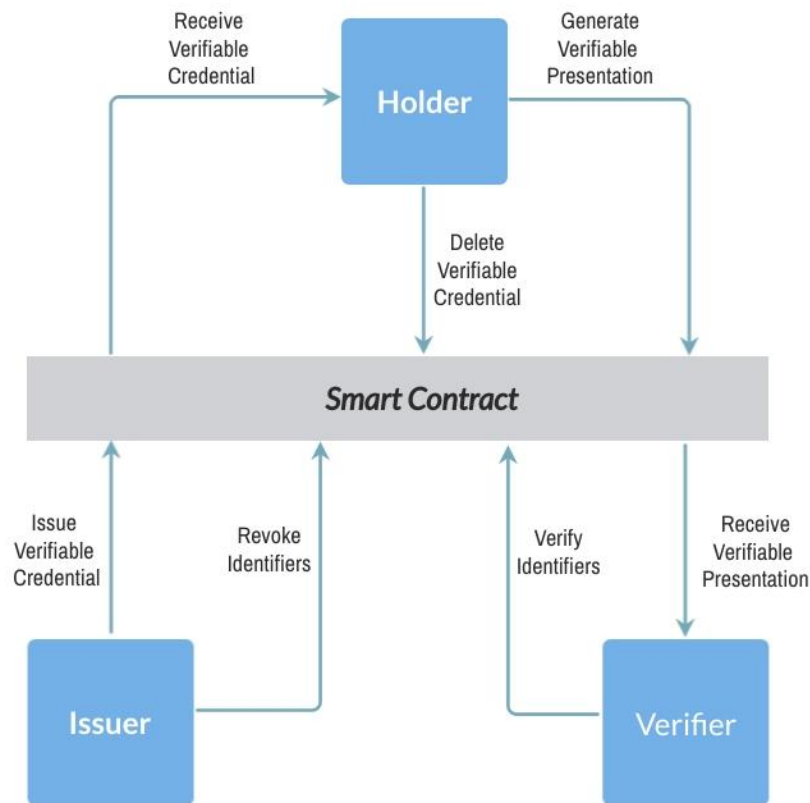
#### - Deployment Backlogs

- Deploy smart contract on rinkeby network
- Deploy front-end project on netlify app
- Setup wallet provider (MetaMask)

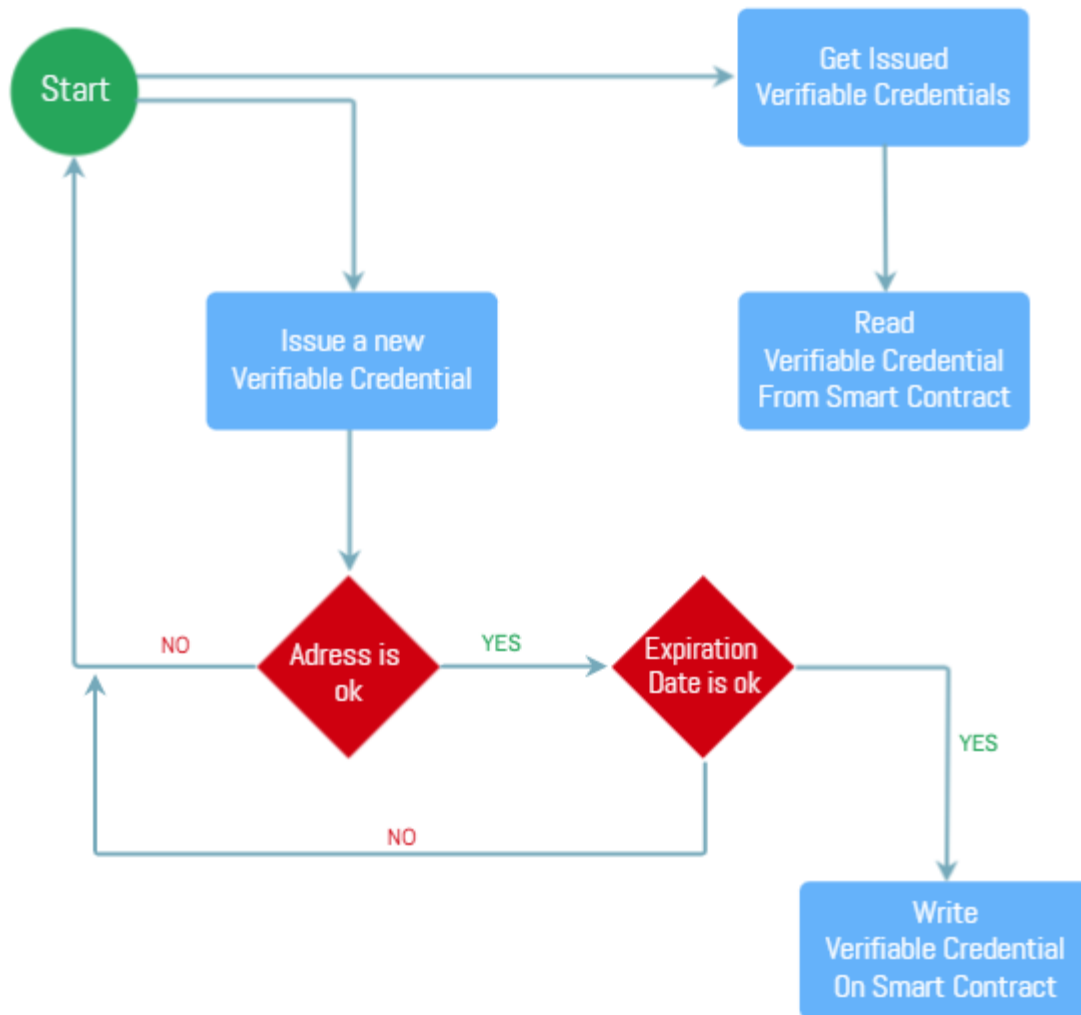
## Architectural Design and System Model

### - Decentralized Identifier Mechanism

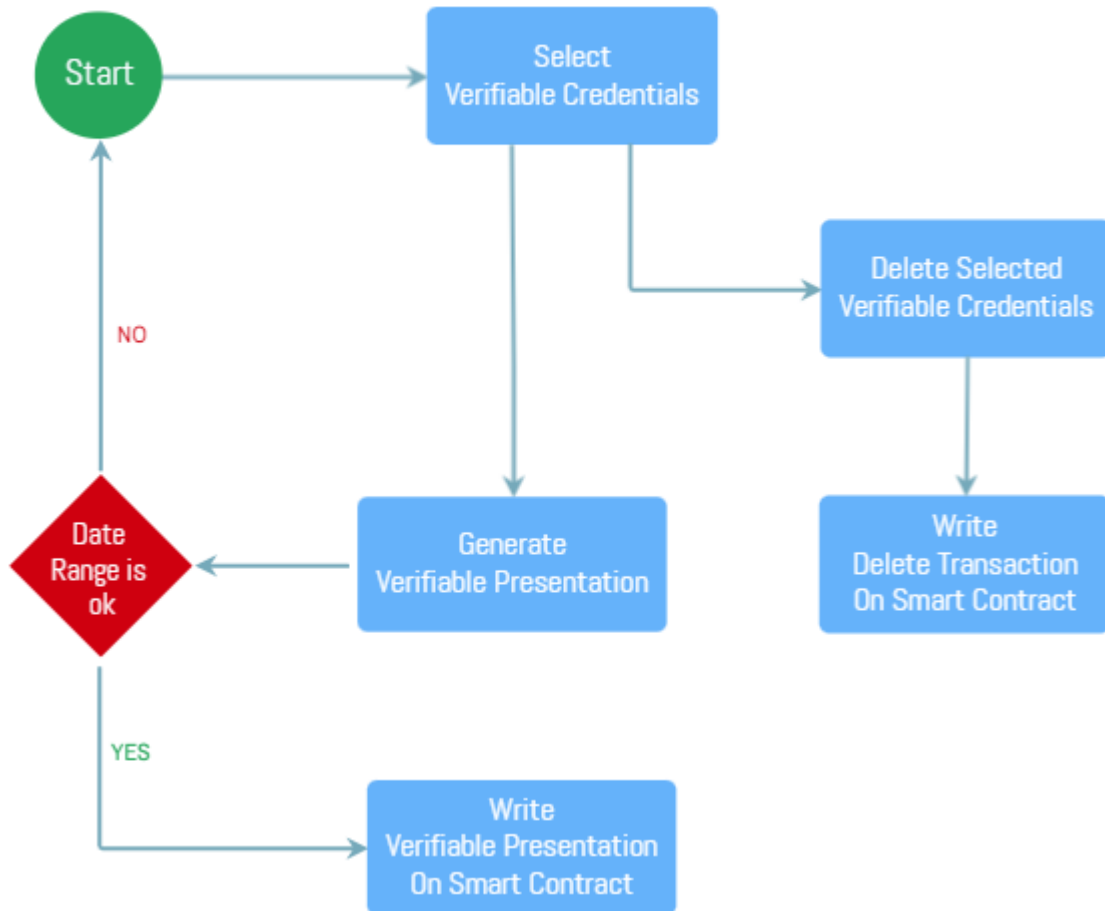
The issuer is probably an organization that can issue new Verifiable Credential (VC) and write it on the smart contract. Then the holder can read all his/her VCs from the smart contract. In addition, the issuer can revoke or (un)suspend specific VCs. For verifying the specific VCs, the holder can generate a Verifiable Presentation (VP) and write it on the smart contract. The holder can also delete arbitrary VCs. The verifier can verify the VPs on the smart contract by having the verify code and holder address.



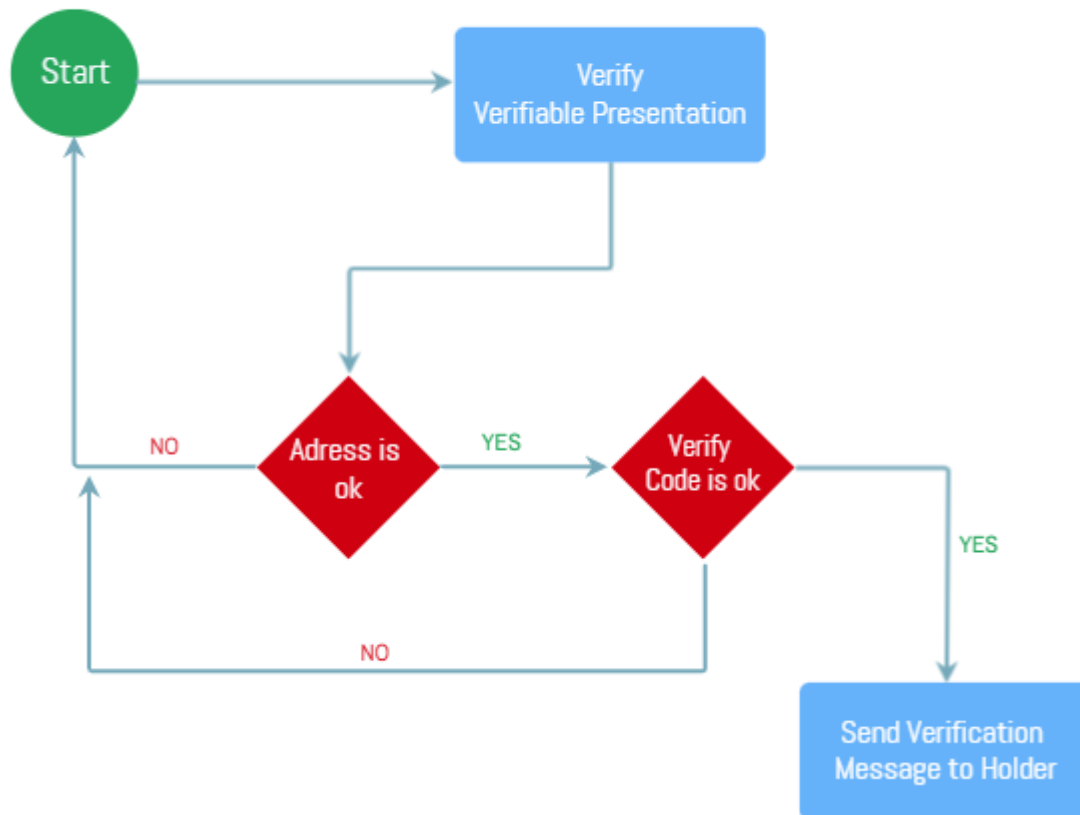
## High-Level Issuer Architecture



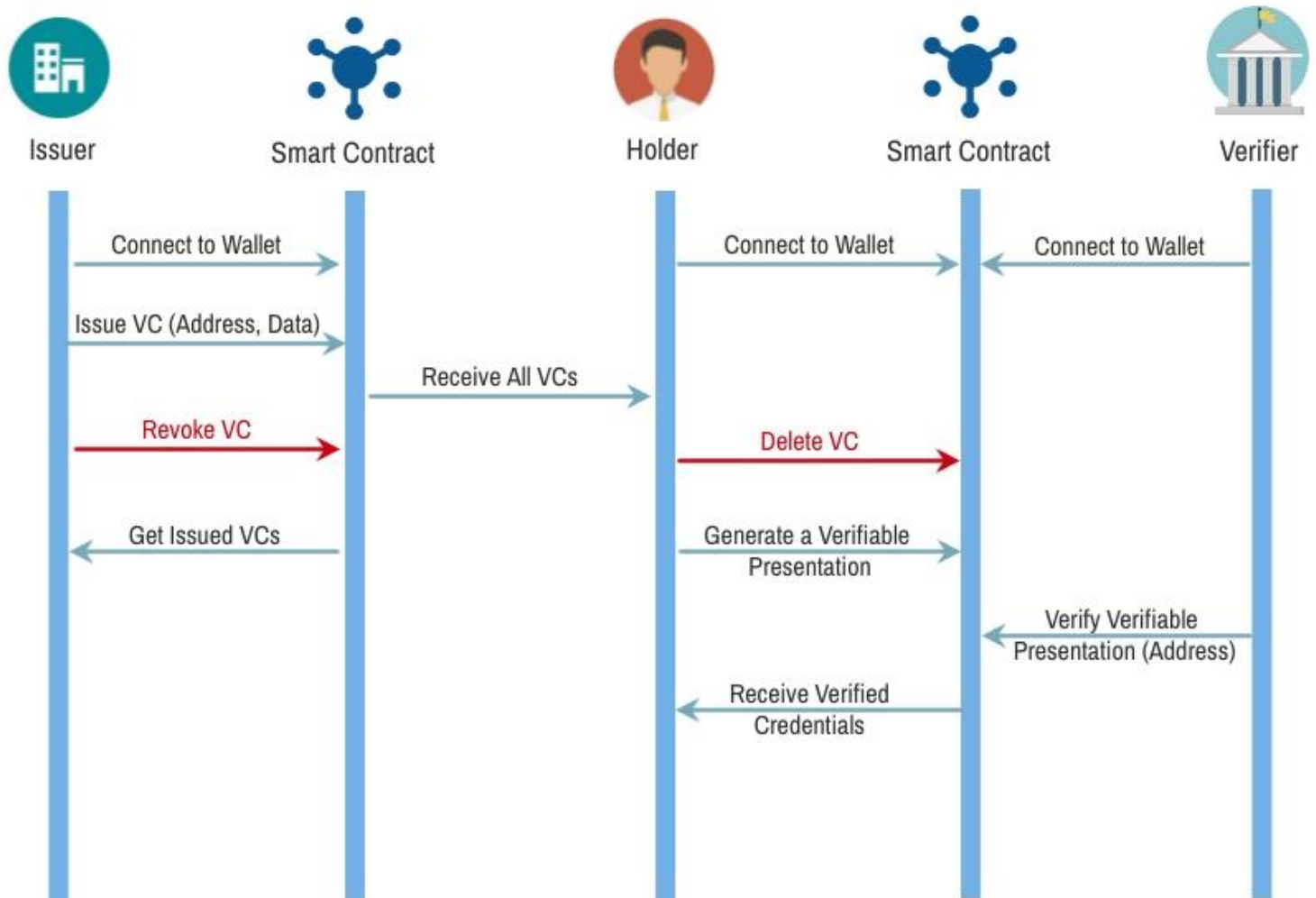
## High-Level Holder Architecture



## High-Level Verifier Architecture



## Issue, Hold and Verify Sequence Diagram



# Implementation

## Back-end Technologies

Unlike the traditional back-end technologies, which are centralized, in this project we had to use decentralized technologies because our project is based on the ethereum blockchain which is decentralized.

Because of the points that have been mentioned, we used smart contract to implement our back-end. Smart contract is a piece of code which is deployed on the Ethereum blockchain network.

Smart contract is written by Solidity Programming Language, Solidity is an object-oriented programming language for writing smart contracts. It is used for implementing smart contracts on various blockchain platforms, most notably, Ethereum.

In order to implement the VCs we had to use non-fungible tokens or NFTs and ERC721 provides a standard for this kind of token. Put simply, being “non-fungible” means that they are unique.

In the development process, we used the Hardhat framework to enhance our development such as compilation, deploying, running our contract on the development network and testing.

For testing, we used TypeScript Programming Language, so that our tests become more reliable and maintainable.

## Front-end Technologies

We used one of the most famous front-end frameworks to enhance our product which is Vue js, and it is a progressive, incrementally-adoptable JavaScript framework for building UI on the web.

As a project bundler, we used Vite for better development experience.

We used Quasar, so that we did not have to write CSS styles from base. Because Quasar provides some useful base components of CSS.

We tried to code in a modular and reusable paradigm.

As a programming language we used typescript, as it is the superset of javascript, a strongly typed language, and scalable.

We used Pinia as a global state manager in our project, State management is simply a way to engender communication and sharing of data across components. It creates a concrete data structure to represent your app's State that you can read and write.



## DevOps

For DevOps we used Netlify as our front-end host ,and it comes along with CI/CD service.

## Testing

Our development Technique is Test Driven Development (TDD). We used Mocha and Chai for automated testing. Mocha is a feature-rich JavaScript test framework running on Node.js, and Chai is a TDD assertion library for node. We divided our test cases into 3 test suits:

- **Issuer tests**

Test Number	Test Summary	Test Procedure	Test Goal	Expected Result	Test Result	Test Environment
1	The balance of addr1 should increment by the number of VCs that are issued	issue 3 VCs for the addr1	Check "issue" function	The balance should be equal to 3	Correct	Web Browser
2	The number of VCs of addr1 should increment by the number of VCs that are issued	issue 3 VCs for the addr1	Check "issue" and "vcsOfHolder" functions	The number of VCs of addr1 should be equal to 3	Correct	Web Browser
3	The address of holder of a VC should be correct	issue a VC for the addr1	Check "ownerOf" function	The address of owner of holder of the created VC should be equal to addr1.address	Correct	Web Browser
4	No one can issue a VC for himself	owner of the contract issues a VC for himself	Check edge case for "issue" function	revert with: "DevDIDS : self issuing is not permitted"	Correct	Web Browser
5	The valid from date of a VC should not be later than its valid to date	issue a VC which its valid from date is later than its valid to date	Check edge case for "issue" function	revert with: "DevDIDS : vc valid from must be greater than valid to "	Correct	Web Browser
6	The balance of addr1 should decrease by the number of VCs that are revoked	issue 3 VCs for the addr1 and revoke one of them	Check "revoke" function	The balance should be equal to 2	Correct	Web Browser
7	Only the issuer	issue 3 VCs	Check edge	revert with: "DevDIDS :	Correct	Web Browser

	of a VC can revoke that VC.	and the address that is not the issuer of them tries to revoke one of them.	case for "revoke" function	you can not revoke a VC that you not issued"		
8	A VC that does not exist can not be revoked	revoke a VC that does not exist	Check edge case for "revoke" function	revert with: "DevDIDS : you can not revoke a VC that you not issued"	Correct	Web Browser

## • Holder tests

Test Number	Test Summary	Test Procedure	Test Goal	Expected Result	Test Result	Test Environment
1	Generating a VP which is valid.	Issue 3 VCs and create a VP with them and assign it to vcs variable.	Check "generateVp" function	The elements of the vcs should be equal to the VCs.	Correct	Web Browser
2	Creating a VP for another Person	Issue 3 VCs for the addr1 and the owner of the contract create VP	Check edge case for "generateVp" function	revert with: " DevDIDS: all of the vcs must belong to you"	Correct	Web Browser
3	The valid from date of a VP should not be later than its valid to date	Issue 3 VCs and create a VP with them and set from date later than valid date.	Check edge case for "generateVp" function	revert with: "DevDIDS : vp valid from must be greater than valid to "	Correct	Web Browser
4	Deleting a vc.	Issue 3 VCs for the addr1 and then addr1 tries to delete one of them	Check "delete" function	The balance of addr1 should be equal to 2	Correct	Web Browser
5	The owner of the VC can delete a VC, and others can not delete it.	Issue 3 VCs for the addr1 and the owner of the contract tries to delete one of them	Check edge case for "delete" function	revert with: "DevDIDS : you can not delete VC that you not hold"	Correct	Web Browser

## • Verifier tests

Test Number	Test Summary	Test Procedure	Test Goal	Expected Result	Test Result	Test Environment
1	Check if a VP is valid or not.	Issue 3 VCs and create a VP with them and assign it to the vp variable.	Check “verify” function	The elements of the vcs should be equal to the VCs.	Correct	Web Browser
2	If the VP is not valid the verifier should understand.	Issue 3 VCs for the addr1 and create VP for them and then check if addr2 has that VP.	Check edge case for “verify” function	revert with: “DevDIDs:holder is not owner of all vcs.”	Correct	Web Browser

By testing each module we had our integration test. Then for the system test (backend) we deployed our code on Ethereum test net, and the system test (frontend) the front code is deployed on netlify. Netlify offers hosting and serverless backend services for web applications and static websites. For the release testing we sent the address of our website to several people, who told us about the problems of our product. We used the inspection test to review our work and find any defects.

We also had user testing, and we sent our product to the product owner. His response was. He approved it in his last [email](#).

# Contact

You can find the production at <http://dev-dids.netlify.app/> and there is a user guide for the production [here](#)

contact us if you had any ideas or suggestions.


[ZahraMohammadPour497@gmail.com](mailto:ZahraMohammadPour497@gmail.com)

[daneshmand.saina@gmail.com](mailto:daneshmand.saina@gmail.com)

[mohsen137853@gmail.com](mailto:mohsen137853@gmail.com)

[amin.bghb7@gmail.com](mailto:amin.bghb7@gmail.com)

# References

- [1]  Project proposal(s) - Hata
- [2] ION
- [3] Azure
- [4] id4d
- [5] Verifiable credentials Microsoft
- [6] W3 Verifiable Credentials Data Model
- [7] Github Front-end Repository
- [8] Github Back-end Repository