

Rokas Bliudzius
Martin Slind Hagen
Diderik Kramer

Decentralized Identity - a mobile wallet and verification platform

Bachelor's thesis in Bachelor of Engineering in Computer Science

Supervisor: Surya Kathayat

May 2022

Rokas Bliudzius
Martin Slind Hagen
Diderik Kramer

Decentralized Identity - a mobile wallet and verification platform

Bachelor's thesis in Bachelor of Engineering in Computer Science
Supervisor: Surya Kathayat
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



NTNU

Kunnskap for en bedre verden

Abstract

This bachelor thesis consists of developing a minimum viable product (MVP) of a digital ID wallet, an application for storing and verifying IDs. The MVP is an interpreted implementation of a recent concept developed by the European Commission (European Commission, 2021) and Thales (Thales, n.d.). The thesis aims to contribute towards research and development of the digital ID field. Currently, ID issuers provide services for a more secure authentication process online compared to username and password authentication, via digital IDs. However, it can provide a bad user experience as they may have contrasting interfaces to their systems. Also, standards for digital claims about entities, and online identifiers respectively called Verifiable Credentials and Decentralized Identifiers, have been defined in recent years. A digital ID wallet would aim to provide an easier way for users to manage their digital identities compared to the existing digital IDs. The resulting MVP uses Verifiable Credentials, Decentralized identifiers and blockchain technology that respectively provide a cryptographical way to prove attributes related to an individual, decentralized and globally unique identifiers, and tamper proof decentralized storage. However, as blockchains can be quite complex the solution also includes a centralized backend serving as a man in the middle between the blockchain and external entities. In total the components used in the ecosystem are a mobile application, blockchain, centralized backend and a mock issuer responsible for simulating the behavior necessary from an issuer to function with the rest of the ecosystem.

The team used Kanban during development, allowing them to work efficiently. The work resulted in the creation of the four aforementioned components that form the ecosystem. The system allows users to fetch and store their digital IDs, as well as using them for identification purposes by creating Verifiable Presentation from an ID and transforming the data to a QR code. To prevent falsification of digital IDs, a blockchain is used to store agreements as proof of existing IDs. The agreements do not store personal information to hinder sensitive data leakages. The system also enables users to verify other users' identity credentials by scanning a QR code and checking if the agreement related to the ID is registered in the blockchain and still valid.

Different technologies have been used when developing the components, to test what would work in such ecosystems. An emphasis was also put on learning new technologies that the team found interesting and wanted to try. The components have been thoroughly developed to meet certain requirements of security, confidentiality, ownership, and control.

Overall, the report provides necessary background theory to understand the problem domain and the solution. It also describes the MVP, discussing the results and examining potential for future work.

Sammendrag

Denne bachelor oppgaven omhandler utviklingen av en digital ID lommebok, som et minste brukbare produkt (MVP). Applikasjon kan lagre og verifisere ID-er, og er en tolket implementasjon av et nytt konsept som er utviklet av European Commission (European Commission, 2021) og Thales (Thales, n.d.). Oppgavens mål er å bidra med forskning og utvikling av en digital ID feltet. I feltet finnes det ID utsteder som tilbyr digitale ID-er, en sikrere autentiseringsmetode sammenlignet med brukernavn og passord. Derimot, kan digitale ID-er gi dårlig brukeropplevelse da forskjellige ID utstedere kan ha forskjellige grensesnitt mot systemene sine. Standarder for digitale påstander angående entiteter, og nettbaserte identifikatorer har i tillegg blitt definert de siste årene. Standardene heter respektivt Verifiable Credentials og Decentralized Identifiers. En digital ID lommebok ville siktet på å tilby en enklere måte for brukere å håndtere sine digitale identiteter sammenlignet med den eksisterende digital ID teknologien. MVP som lagdes bruker Verifiable Credentials, Decentralized Identifier og blokkjede teknologi som respektivt bidrar med en kryptografisk måte å bevise attributter tilknyttet et individ, desentralisert og globalt unike identifikatorer, og sabotasjesikker desentralisert lagring. Samtidig kan blokkjeder være komplekse, derfor har løsningen en sentralisert tjener som kommuniserer mellom blokkjede og eksterne entiteter. Totalt sett består økosystemet av en mobil applikasjon, en blokkjede, sentralisert tjener og en spotteutsteder som simulerer den nødvendige oppførselen til en utsteder for å fungere med økosystemet.

Gruppen brukte Kanban under utvikling for å kunne jobbe effektivt. Arbeidet endte med et økosystem bestående av de fire tidligere nevnte komponentene. Systemet tillater brukere å hente og lagre deres digitale ID-er, i tillegg til å kunne bruke dem for identifiseringsformål ved å lage en Verifiable Presentation fra en ID og transformere dataen til en QR kode. For å forhindre forfalskning av digitale ID-er brukes en blokkjede som lagrer avtaler som bevis på eksisterende ID-er. Disse lagres uten personlig data for å forhindre sensitive datalekkasjer. System gir brukere mulighet til å verifisere andre brukeres ID-er, ved å ha skanne en QR-kode og sjekke om den tilhørende avtalen ligger i blokkjeden samt fortsatt er gyldig.

Forskjellige teknologier har blitt brukt under utvikling av komponentene, for å teste hva som ville ha fungert i et slikt økosystem. Det har også blitt satt fokus på å lære de nye teknologiene som gruppen synes var interessante og ville teste. Komponentene har blitt grundig utviklet med hensyn på visse krav til sikkerhet, konfidensialitet, eierskap og kontroll.

Totalt sett gir rapport nødvendig bakgrunns teori for å kunne forstå problemdomene og løsningen. Den forklarer også MVP-en og diskuterer resultatene samt ser på potensielt fremtidig arbeid.

Preface

As we all have expressed interest in cyber security, cryptography, privacy preserving technology and network programming throughout the course of our degrees, it was natural to select a task involving those aspects of computer science. Before looking through the proposed tasks we knew that we wanted to work together as we had previously done with success. We also decided to send a thesis proposal for creating a password manager, while also staying open minded about other tasks. When looking through the different options, we saw one about researching blockchain technologies which spiked our interest. However, we wanted to create an application, and found the concept of a digital ID wallet enticing, because of its connections to the aforementioned computer science aspects. It also sounded like there was a possibility to apply blockchain technologies to the assignment, which peaked out interest in choosing this task.

The team and the supervisor agreed to identify this thesis as a hybrid of system development and scientific research. Even though the thesis was about developing an MVP, the team found that different scientific questions needed to be answered, along with exploring new technology. Therefore, the report pushes the set limit of 50 pages for a system development task.

We have constantly worked throughout the semester during regular working hours, while sitting together digitally or physically in hopes of achieving great results. The team members have continuously supported and helped each other to streamline the work. The problem domain was new to us, meaning that we gained a lot of useful knowledge and experience in recently growing technological fields. We see this thesis as a great milestone, where the new learnings will hopefully be useful for our future careers.

We would like to thank our supervisor and client Surya Kathayat for providing us with knowledge, wisdom, guidance, and encouragement during our many meetings. He helped us discuss ideas, leading us to make important choices and completing the thesis. We would also thank him for taking the initiative to proofread documents and funding.

We would like to dedicate this thesis to our families for their continuous support and encouragement.

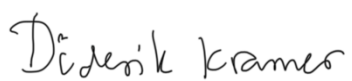
Trondheim May 18, 2022



Rokas Bliudzius



Martin Slind Hagen



Diderik Kramer

Task description

The original task description is still in use (Appendix C). The requirements for the system can be found in the vision document (Appendix F) and the requirement document (Appendix G).

Table of contents

Figures	xii
Abbreviations/symbols	xii
1 Introduction	13
1.1 Thesis/Problem statement	13
1.2 Structure	13
2 Theory	14
2.1 Attack vector	14
2.2 Threat actor	14
2.3 DLT	14
2.4 Blockchain.....	14
2.4.1 Cryptographic Hashes	14
2.4.2 Decentralized	15
2.4.3 Tamper-proof	15
2.4.4 Consensus mechanism	15
2.4.5 Smart contract.....	16
2.4.6 Parachains	16
2.5 Dapp	16
2.6 QR	16
2.7 DIDs	16
2.8 Zero-knowledge proof.....	17
2.9 VCs	17
2.10 VPs	17
2.11 Verifying VC/VP	17
3 Method.....	19
3.1 Research method	19
3.2 Choice of technology.....	20
3.2.1 Solana and Anchor	20
3.2.2 Polkadot and Substrate	21
3.2.3 React Native	21
3.2.4 Next JS.....	21
3.2.5 UI libraries	21
3.2.5.1 React Native Paper	21
3.2.5.2 MUI.....	22
3.2.6 Jest.....	22
3.2.7 Veramo	22

3.2.8	Prisma and PostgreSQL.....	22
3.2.9	Redis.....	22
3.2.10	WebSocket.....	23
3.2.11	Git	23
3.2.12	TS.....	23
3.2.13	React Native Encrypted Storage	23
3.2.14	Supabase.....	24
3.3	Development method.....	24
3.3.1	Kanban.....	24
3.3.2	Experimenting through "Hello world" repositories.....	24
3.3.3	Wireframing in Figma.....	24
3.3.4	Work and role management.....	25
4	Result	26
4.1	Scientific Results	26
4.1.1	Ecosystem architecture	26
4.1.1.1	Aphrodite.....	26
4.1.1.2	Athena	26
4.1.1.3	Iris.....	26
4.1.1.4	Kratos	27
4.1.2	Handling ID.....	27
4.2	Engineering results.....	27
4.2.1	ID storage.....	27
4.2.1.1	Add ID.....	27
4.2.1.2	Remove ID.....	28
4.2.1.3	Refresh ID	29
4.2.1.4	Encryption of ID.....	30
4.2.2	ID viewing	31
4.2.2.1	Card display	31
4.2.2.2	Search and filter IDs.....	31
4.2.2.3	Scroll through the IDs.....	32
4.2.3	Verification.....	32
4.2.3.1	Generate QR-code to prove one's identity	32
4.2.3.2	Verify other identities by scanning other QR-codes.....	32
4.2.3.3	Selection of ID data to be used	33
4.2.4	Authentication	34
4.2.4.1	Unlock wallet with password	34
4.2.5	Provider registration	35

4.2.5.1	Register as ID provider with public key and DID	35
4.3	Administrative results	36
4.3.1	Project schedule.....	36
4.3.2	Time management and activity distribution	36
4.3.3	Agile development.....	36
5	Discussion	37
5.1	Scientific.....	37
5.1.1	Ecosystem architecture	37
5.1.1.1	Centralized backend vs fully decentralized ecosystem.....	37
5.1.1.2	Mock issuer	37
5.1.2	Handling ID.....	37
5.1.2.1	Blockchain agreement model	37
5.2	Engineering	38
5.2.1	ID storage.....	38
5.2.2	ID viewing	39
5.2.3	Verification.....	40
5.2.4	Authentication	41
5.2.5	Provider registration	41
5.3	Administrative	42
5.4	Group reflection	42
5.5	Veramo	43
6	Conclusion and future work	44
6.1	Conclusion	44
6.2	Future work.....	44
	Sustainability	46
	References	48
	Appendix	53

Figures

Figure 4.1: Adding an ID	28
Figure 4.2: Deleting an ID	29
Figure 4.3: Editing an ID card	30
Figure 4.4: ID data with sharing options.....	31
Figure 4.5: Searching in app.....	32
Figure 4.6: Sharing data through a QR-code	32
Figure 4.7: Verifying a user's Passport	33
Figure 4.8: Share data screen, for combining IDs.....	34
Figure 4.9: Wrong PIN code.....	35
Figure 4.10: Issuers register page	35

Abbreviations/symbols

API	Application programming interface
CD	Continuous Development
CI	Continuous Integration
CRUD	Create, Read, Update, Delete
DAPP	Decentralized application
DID	Decentralized Identifier
DLT	Distributed ledger technology
GDPR	General data protection regulation
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JSON Web Token
MUI	Material UI
MVP	Minimal Viable Product
PoH	Proof of History
PoW	Proof of Work
QR	Quick Response
REST	Representational state transfer
SDG	Sustainable Development Goal
TS	TypeScript
VC	Verifiable Credential
VP	Verifiable Presentation

1 Introduction

Services today are dependent on different authentication methods, such as usernames or passwords, or the more secure option digital or physical IDs. Many different ID providers are available for use, making it hard for users to keep track of all their ID data. However, new technologies such as blockchain and VC, in theory, enable a new concept worked on by the European Commission (European Commission, 2021) and Thales (Thales, n.d.).

The concept is a digital ID wallet application, which would create a system for holders to safely store all IDs. The system would ensure that the holders have full ownership of their personal data, and that no central entity will be able to “collect” ID user data. It would act like a physical wallet with multiple holder’s IDs stored in one place. The digital ID wallet, just like a physical wallet, will only belong to the user, giving them full control and freedom to share chosen data with whomever and whenever.

This thesis aims to provide the necessary foundation towards realization of the concept by creating a minimum viable digital ID wallet application. Therefore, some questions need to be answered. Firstly, how the IDs are handled, e.g., where and how to store the IDs? Secondly, how the ecosystem is designed, e.g., what components are needed, what data should they store, would an issuer need to make changes to adapt to the ecosystem?

The thesis also aims to test viable technologies to implement an ecosystem in which a digital ID wallet application can be trustworthy with handling and sharing of identity credentials. It also showcases a solution, discussing issues and potential solutions.

1.1 Thesis/Problem statement

Create a minimum viable digital ID wallet application.

1.2 Structure

The report explains the theory and the relevant literature in chapter 2. The research method, development method and choices of technologies are described in chapter 3. The scientific-, engineering- and administrative results can be found in chapter 4. The discussion of the results is found in chapter 5, with the conclusion and recommendations for future work in chapter 6. Lastly, the “Social impact” chapter discusses the thesis from a holistic system perspective.

2 Theory

2.1 Attack vector

An attack vector is a “specific point of attack (e.g., user input field)” (Morrison).

2.2 Threat actor

A threat actor is an entity that performs cyber-attacks on other entities (Nätt, 2022).

2.3 DLT

DLT is described as technological infrastructure which allows operations and synchronizations of a distributed, immutable, append-only, digital ledger (Sunyaev, 2020). DLTs often use cryptography to justify authenticity of data, creating trust in the underlying technology and removing the need of trusted third parties. Moreover, a distributed ledger consists of a peer-to-peer network of nodes across various locations, creating a decentralized system where decisions are made by all the nodes using an agreed consensus mechanism, meaning that a single node cannot solely control the system, preventing malicious intents towards the system (Moubarak, Chamoun, & Filiol, 2020). A decentralized network of nodes also prevents system downtime since a failure in a single node will not prevent the entire system from working, in the same way that one malicious node will not corrupt the network.

2.4 Blockchain

Blockchain is a type of DLT, providing a digital and decentralized form of data storage. As the name implies, a blockchain is a “chain of blocks” where each block usually contains a hashed batch of data, and a cryptographic hash of the prior block on the chain. One can think of it as a linked list, but instead of pointers pointing to the next block, a blockchain is backlinked using the cryptographic hash of the previous block to reference it. An example of data inside a block can be cryptocurrency transaction information from one decentralized wallet to another. The verification of authenticity, maliciousness, and validity of the data in a block is done by validators or nodes using a specific consensus mechanism (Lie & Øverby, 2022).

2.4.1 Cryptographic Hashes

“A hash function is used to construct a short “fingerprint” of some data; if the data is altered, then the fingerprint will (with high probability) no longer be valid.” (Stinson & Paterson, 2018). A cryptographic hash function is deterministic, using the same input, always results in the same output, and different inputs should give different outputs. Another feature of hash functions is that they are supposed to be one-way, meaning that given a hash, one should not be able to calculate the input. This is dependent on the hash function used. Cryptographic hashes are used in blockchains to generate the cryptographic hash of a block’s data, which will be used in the next block to create a back-link (Stinson & Paterson, 2018).

2.4.2 Decentralized

Blockchain is a decentralized technology, meaning control of decisions is transferred from a centralized entity (individual, groups, or organizations) to a distributed network. This network consists of nodes or “miners” which are computers located anywhere, each having a copy of the blockchain and trying to calculate the next block. For open blockchains, anyone can become a node. Each node is connected to one or more nodes using a peer-to-peer connection. This connection is a simple communication protocol that allows transfer of data between two peers (Amazon Web Services, n.d.).

2.4.3 Tamper-proof

A feature of a distributed network is that it prohibits tampering, as every node in the network has a copy of the blockchain. If a malicious node tries to change or append a malicious block to the blockchain, other non-malicious nodes will not validate or accept the change rendering the attempt useless (IBM, n.d.).

2.4.4 Consensus mechanism

In a peer-to-peer network of nodes, there are no guarantees that a neighboring node is not malicious, creating distrust and insecurity between every node. A preset consensus algorithm (also called protocol or mechanism) creates trust between nodes by providing common “rules” to keep data consistent across the network (Zhao, 2018). An example of its use case is when a node wants to append a new block to the blockchain. The initial node will send the block to other nodes in the network using peer-to-peer connection. The nodes will use an agreed-upon consensus protocol to check the validity and security of the block data, and if the majority of the nodes in the network accept it, the block has been agreed to be a part of the blockchain (IBM, n.d.).

Proof of Work is a consensus mechanism introduced in Bitcoin. The idea is that the nodes are in a “race” to solve a difficult mathematical “puzzle”. The first node that calculates the correct “answer” gets rewarded with cryptocurrency native to the blockchain. This means that PoW uses raw computing power to prevent maliciousness. An attacker would need to control over 50% of the whole network’s computing power to ensure addition of a malicious block to the blockchain (Mingxiao, Xiaofeng, Zhe, Xiangwei, & Qijun, 2017). It is almost impossible to achieve that much computing power, especially for large networks such as Bitcoin (Frankenfield, 2022). Nonetheless, PoW’s reliance on computing power makes the network slow and energy consuming, and therefore expensive, creating a need for more modern consensus solutions (Zhang & Kin, 2020).

Proof of History, the consensus mechanism used by Solana, calculates repeated cryptographic hashes to verify the passage of time between two events. As cryptographic hashes do not have predictable outputs, the hashes need to be calculated to know the output. Proof of History is implemented on this basis. PoH sequence starts with a function cryptographically hashing a random input value. The output hash is then used as input to hash the next value. PoH does this recursively multiple times, hashing the output of previous hash, whilst recording the outputs and the number of iterations the hash function was called. This is all done using a single core, since the sequence is recursive and cryptographic hashes are unpredictable. This ensures that an amount of real time has passed between iterations, while also ensuring that the order of each “record” in the sequence is correct. After the sequence calculation is finished, the sequence data is stored in a block and sent to other nodes in the network. The nodes receiving the block can verify the sequence in parallel by using multiple cores to “cut” the sequence in

multiple parts, each calculating and proving that the hash values are correct. By proving that the hashes are correct, the node also proves that some real time has passed between recorded events creating a cryptographic timestamp. This helps the whole network of nodes with time synchronization and ordering of events, making transaction speed much faster compared to networks with different consensus mechanisms, such as Bitcoin with PoW (Yakovenko, n.d.).

2.4.5 Smart contract

In the context of blockchains, a smart contract can be described as a predefined and automatically executable script stored on a blockchain. When a smart contract has been deployed on a blockchain, it can be triggered by sending a transaction to the contract's on-chain address. A triggered contract will execute likewise on every node in the network, the difference being the output which may vary depending on the data that was sent with the transaction (Christidis & Devetsikiotis, 2016). A byproduct of being stored on a blockchain, is that smart contracts can never be removed, likewise for their content. This means that any actions towards an on-chain smart contract cannot be reverted. Therefore, sending sensitive data to the smart contract's storage might be dangerous if there is no "delete" function predefined in the contract (Ethereum, n.d.-b).

2.4.6 Parachains

Parachains, or "parallelized" chains, is a data-structure introduced in the Polkadot blockchain. A parachain can be seen as an independent chain connected to a "relay chain" that provides it with a common security layer. This differs from other blockchain technologies which have only one coherent blockchain and only allow development of smart contracts and decentralized apps. Parachain development gives the developers full control over the structure and functionality of the chain itself, providing a wider range of optimization (Dr. Wood, n.d.) (Polkadot, n.d.).

2.5 Dapp

Dapps differ from typical apps by not having a centralized backend server. Instead, dapps utilize decentralized peer-to-peer networks e.g., blockchains, to execute backend code. Ethereum, a popular blockchain, defines a dapp as "an application built on a decentralized network that combines a smart contract and a frontend user interface" (Ethereum, n.d.-a).

2.6 QR

QR is a standard for displaying data converted to a series of pixels in a square-shaped grid. Contrary to a barcode, it is two-dimensional allowing for more data to be stored. By scanning the QR-code the stored data can be read. QR-codes usually contain URLs or phone numbers (Hayes, 2021).

2.7 DIDs

DIDs are used to identify subjects. By being globally unique it can be used as an identifier in decentralized networks. Since the controller of a DID can prove their ownership, no trusted entity is needed, which enables decentralized usage. The subject can also be the controller. Each DID has a DID document that describes it, e.g., how to verify the controller and the necessary data to complete the verification process. The format of the document can be anything that may represent the data, such as XML or

JSON. A DID is identified with the format "did:method:method-id", e.g., "did:example:987654321". The different methods specify how the DID is handled for different phases of its lifecycle, such as creation and DID resolution (the process of converting a DID to a DID document). The DID identifier can be extended to a URL with a path (using "/"), query (using "?"), fragment (using "#") or any combination of these. The URL enables retrieving sections of a DID document and other related resources. DIDs are stored on verifiable data registries, such as databases or blockchains. These systems need to be able to transmit the required data for generating a DID document and be able to store the DID (Sporny, et al., 2021).

2.8 Zero-knowledge proof

"A zero-knowledge proof is a cryptographic method where an entity can prove to another entity that they know a certain value without disclosing the actual value" (Sporny, Longley, & Chadwick, 2022d).

2.9 VCs

VCs are a collection of claims regarding one or multiple credential subjects, which can be verified through cryptographic means. This does not mean that the claims are true, but that they have not been tampered with. Such claims are given inside the credential subject property of the VC, and can be made regarding anything, e.g., a university degree. Claims have the model: subject-property-value. E.g., alice-age-35. To identify a subject or value the id property can be used, but it can also be omitted for more privacy in regard(s) to the subject(s). For its value a DID is sufficient, as the id property is meant to be globally unique and requires a URI format. Proving the claims is done via the required proof property, which can be a JWT along with other types of proofs. This requires information about the issuer such as its public key. Therefore, the issuer property must contain a URI and globally unique identifier that can be dereferenced to the required information for verification, such as a DID. By using the expiration date property issuers can issue credentials that are only valid until the expiration date arrives. Verifiable credentials can use any format that is able to represent the data structure, such as XML or JSON as format (Sporny, Longley, & Chadwick, 2022c).

2.10 VPs

VPs can be created and combined from different sources. Firstly, it can be a verifiable subset of data from a VC. E.g., only using the birth date from a VC representing a passport. Secondly, it can be a verifiable combination of multiple VCs or VPs. E.g., combining VCs representing job status and financial status when applying for a bank loan. Lastly, it can be verifiable data derived from a VC. E.g., create a zero-knowledge proof from the subject's birth date in another VC or VP proving that the credential subject is of legal voting age. For proving to the verifier that all VCs used in the VP were issued to the same holder the proof property is recommended. This property also ensures that no unused information from the VCs is exposed to the verifier (Sporny, Longley, & Chadwick, 2022a).

2.11 Verifying VC/VP

VCs and VPs must contain the proof property containing a method for verifying the data. For instance, by using RSA signing, an issuer can send a VC to a wallet with a proof property containing the signed value of the entire VC, the proof method (RSA signing)

and the public key used for decrypting the signed value. The wallet can then generate VPs that consists of one or more VCs, that can be presented to a verifier. The verifier should be able to use the public keys of the issuers contained in the proof property to validate each VC by decrypting the signatures and checking that they equal the content of the VCs (Sporny, Longley, & Chadwick, 2022b).

3 Method

3.1 Research method

The research method or scientific method is a way to systematically uncover knowledge about the world, while also assuring the quality of the discovered knowledge. The reason the scientific method is important is the discovery of quality knowledge, which can be built upon to gain new knowledge or used in inventions that can improve the standard of life for humans (Kjelsberg, 2019).

The team applied the deductive method for developing the conceptual idea of the project. Firstly, a problem was formed from the conceptual thesis task to develop a Digital ID Wallet, to research and discover if it is possible to store digital IDs in a decentralized manner and use them for verification and identification. This idea was motivated by similar concepts from organizations such as the European Commission (European Commission, 2021). Secondly, the team created a conjecture or a confirmable hypothesis for the problem, consisting of an MVP implementation of the ecosystem and a list of result goals and use cases that the project was aiming for (Appendix D). These goals were measurable, and therefore testable, allowing the team to test the end results. In addition to the result goals, manual tests (Appendix I) were formed from the use cases to test the components together and that combinations of the result goals worked as intended. These represented a type of manual integration test for testing that the conceptual idea of developing a digital ID wallet had achieved parts of its functionalities. Along with each use case are steps to reproduce the results, and screenshots acting as the observation for each test. Depending on the results of the goals and tests, they either support or falsify the hypothesis.

The deductive method described above was also used iteratively by forming sub-problems from the main problem of developing a digital ID wallet. A sub-problem could for instance be the communication between two entities in the ecosystem, like the backend (Iris) and mobile application (Aphrodite). The team would test different approaches for solving the sub-problem, and after discovering a functional solution, they would continue the work on another sub-problem. This was done iteratively, until the solutions to the sub-problems could be combined to solve the main problems. When solving sub-problems, new sub-problems could also be discovered and added to the queue of sub-problems needed to be solved in the current or future iterations (Understanding Science, n.d.).

Another scientific method applied is the Design Science Research method. Its purpose is "achieving knowledge and understanding of a problem domain by building an application of a designed artifact." (Hevner, March, Park, & Ram, 2004). As mentioned in the deductive method description, the main goal of the project was to use conceptual ideas to develop an MVP application. By building a working application, the team achieved a deeper understanding and knowledge of the problem within the task. For example, the concept of VC was used to create verifiable proof of a document. By adapting the concept to the Digital ID Wallet ecosystem, the team gained knowledge about the cryptographic methods that can secure digital IDs, and how digital IDs could function in a system without centralized authorities.

One scientific strategy used was research on the conceptual idea, before designing an implementation and developing it. Research consisted of reading documentation on the necessary practical and theoretical functionalities in the ecosystem. Examples of these would be DIDs, VC, blockchain and smart contracts using standards, documentation, and articles. Many technological concepts were new to the team and essential to understand before being implemented. For example, understanding the connection between VC and VP and how each part (issuer, holder, verifier etc.) of the ecosystem utilizes them. These concepts are relatively new, released in 2019, and not finalized as they have been continuously updated (World Wide Web Consortium, 2022). Although prior research was important to learn about the new technologies, practical building of the application itself also proved to be important for the team's understanding of them. Additionally, there was no finalized standard for implementing a digital ID wallet, therefore parts of the ecosystem were built using the team's prior data science knowledge.

After the research, the team began designing an implementation. It involved writing diagrams on how the components (e.g., blockchain) in the ecosystem would interact with each other, what the interactions do, and in what order they execute. The diagrams were to be displayed to the supervisor to be able to gain feedback and discuss further improvements. Thereafter, development of the implementations began.

A weakness of the team's scientific method choices was the lack of empirical user data. Design Science Research method, as mentioned above, does not require any tests, since the understanding of the problem domain is achieved through the app development. The deductive method does require testing to confirm the hypothesis, but the tests are not necessarily empirical user tests. The decision to exclude user testing was made by the team as it was not necessary for completing the main use cases. For instance, improved UI would not help in reaching the main goals of an MVP, however, if the application was to be deployed publicly and used by user, the team would have put more emphasis on user testing.

3.2 Choice of technology

3.2.1 Solana and Anchor

From chapter 3.4 "Summary of user needs" in the vision document (Appendix F), the Digital ID Wallet should provide users the ability to "always be able to verify themselves", by "Using decentralized blockchain technology". "Solana is a decentralized blockchain built to enable scalable, user-friendly apps for the world." (Solana, n.d.-b). No members had any prior knowledge of using a blockchain, therefore the selection was based on the blockchains' development popularity and the member's interests. The smart contract library for Solana is complex, leaving most options and implementations to the developers. As the needed functionalities for the project are adding and updating data, the team decided to use the Anchor library, which simplifies the syntax and implementation for creating a smart contract (Anchor, n.d.). Even with less documentation, it had descriptive examples, and with the simple structure it allowed for expanding the examples to integrate the necessary functionality. Additionally, Solana is described as having high transaction speed and low transaction costs compared to other blockchains (Solana, n.d.-b), which was also one of the factors in the decision-making process. Solana uses accounts to store data between transactions (Solana, n.d.-a).

3.2.2 Polkadot and Substrate

The Polkadot blockchain using Substrate development environment (Wood, n.d.) was considered along with other options, before choosing Anchor and Solana. Polkadot has parachains, which allow developers to customize the underlying blockchain to fit their needs, for instance, edit block generation speed, select what data types that can be stored directly on the chain and select the consensus protocol. Additionally, the documentation is well written, and has simple tutorials with good examples for creating a Polkadot parachain (Substrate, n.d.). The reasons for not choosing Polkadot was issues with using the JavaScript library Polkadot provides, issues deploying a parachain to a test network, and because deploying a parachain to the Polkadot main network would require either financing or community support (Parachains.info, n.d.).

3.2.3 React Native

React Native is a JavaScript library for creating native apps for Android and iOS using React (React Native, n.d.). None of the members had any previous experience with creating a mobile application, therefore never used, or experimented with any of the frameworks. Selecting the framework consisted of choosing a popular one, that had a familiar syntax. The options boiled down to Flutter and React Native, and as most members have had previous experience with React, the team decided to use React Native because of their similar syntax.

3.2.4 Next JS

Next JS is a React framework for creating REST APIs and React frontend applications. It has a simple structure and supports Typescript (Next.js, n.d.). During the planning phase, the team knew the backend and mock issuer would require both a simple UI and a non-complex API. The backend's main feature would be to communicate with the smart contract, and the mock issuer would only need CRUD-operations for handling users. For simplicity and API support, the team decided to use Next JS for the backend and the mock issuer. Moreover, the team wanted to experiment with new technologies, and this was the main reason for not choosing Spring Boot, a framework all members have had experience with.

3.2.5 UI libraries

From chapter 3.4 "Summary of user needs" in the vision document (Appendix F), it is mentioned that the Digital ID Wallet application should be "universally accessible". The project has three different "applications" which a user can interact with, a frontend application, a backend application, and another backend application for mocking an issuer. Since the main part of the project was the frontend mobile application, the team put more focus towards its accessibility development, therefore doing more manual styling in addition to using a UI library designed for mobile applications. For the backend applications, the team decided to choose a UI library for web applications, which was efficient for development purposes, but also provided good accessibility.

3.2.5.1 React Native Paper

Paper is a React Native library that follows Google's guidelines of Material Design. The library was chosen for the frontend application for its rich collection of customizable components, which are well documented and support accessibility standards. Additionally, the library is free, open source and compatible with both Android and iOS (React Native Paper, n.d.).

3.2.5.2 MUI

MUI or Material UI is CSS library that exports styled React components and UI tools (MUI, n.d.). It allows developers to focus less on styling and more on logic. As the main goal of the project was to create the Digital ID Wallet ecosystem, removing the necessity of styling, thereby saving time which could be spent elsewhere. The team had previous experience using MUI, which removed the need for learning it. These arguments describe why the team decided to use MUI in the UI for both the backend and mock issuer.

3.2.6 Jest

Jest is a testing framework for JS and TS that focuses on simplicity (Jest, n.d.). It is a widely used and popular testing framework, and it is already imported when generating a Next JS project. Furthermore, it provides most of the necessary functionality like mocking and assertions for writing tests. However, it had not implemented dependency injection, allowing developers to, for instance, replace database or blockchain connections when running integration tests, which meant the team could not create certain unit tests.

3.2.7 Veramo

The Veramo website describes Veramo as follows:

Veramo is a JavaScript framework that makes it easy for anyone to use cryptographically verifiable data in their applications. It was designed to make it easy for developers who want to use DIDs, verifiable credentials, and data-centric protocols to bring next-generation features to their users (Veramo, n.d.-a).

An essential part of the project was creating and handling DIDs, VCs and VPs. Veramo was recommended by the team's supervisor, and after experimentation and documentation reading, the team understood that the library had flaws. However, the other options were creating a library, this would probably require a lot of time, or using Affinidi (Affinidi, n.d.) that was less developer friendly as it required the application to connect to their ecosystem. This would also defeat the purpose of the assignment, as the digital ID wallet ecosystem would have already been created. As Veramo has implemented solutions for the project's necessary use cases, the team decided to use it.

3.2.8 Prisma and PostgreSQL

Prisma is an open-source and free ORM library for Node and TS (Prisma, n.d.). The library was chosen to integrate with a PostgreSQL database (PostgreSQL, n.d.-b) for saving Solana account keys. The keys had to be stored in large quantities and for long periods of time, which the database is great for since it can handle data of various sizes and quantities (PostgreSQL, n.d.-a). Prisma also integrates well into the backend since it creates types and interfaces when using TS. Additionally, the team had previous knowledge using PostgreSQL.

3.2.9 Redis

Redis is an in-memory data store, which has a lot of use cases, but mainly used as a database or a cache (Redis, n.d.). It was chosen for the project to store blacklisted JWT tokens. The team decided to use Redis instead of the already implemented Prisma library because they wanted to experiment with new technologies. Another reason was that Redis is supposed to store smaller amounts of data, and as a result has optimized speed for completing CRUD operations (Redis, n.d.). Additionally, it has simple and well documented libraries for implementation on both the server-side and the client-side. For

the reasons above, the team decided it was a good fit for storing the JWT tokens that are blacklisted after being used.

3.2.10 WebSocket

Current implementations use a WebSocket connection to send the VC from the issuer to the mobile application of the subject. The team therefore had to select a WebSocket library. Both Socket.io and ws was experimented with for the server side. Socket.io is a communication library using a protocol built on top of the WebSocket protocol with HTTP polling (Singh, n.d.) and reconnecting features (Socket.IO, n.d.). However, some of the functionality was not necessary for the application's purpose, like creating socket rooms and parts of handling client sockets. Socket.io also had issues when closing the server as all clients needed to disconnect before it could close. It did not disconnect client sockets itself. Meanwhile, ws had a simple interface for initializing the server and handling sockets, and closing the server was a simple function call. On the client-side browser WebSocket client was selected even though Socket.io-client has more features for connecting and reconnecting to a server, removing the necessity for a timer in the React component for trying to connect to the server every two seconds.

3.2.11 Git

The team decided to use Git for version control because every member has had substantial experience with the system. For hosting the remote repository Gitlab (GitLab, n.d.) was chosen by the supervisor, and because all members have had previous practice with the platform in previous projects.

3.2.12 TS

TS is an extension of JS which adds typing to the language. It is therefore able to detect logical errors in the code and handle them by throwing explicit errors, which in turn hinder unexpected bugs (TypeScript, n.d.). The addition of typing makes TS easier to structure than JS. Throwing explicit errors eases the developer experience and makes the code more secure as many potential bugs are erased before the code is ever run. This in turn makes development more efficient. An additional benefit is when external libraries implement TS, due to the API being typed which leads to simplifying the process of understanding the API.

3.2.13 React Native Encrypted Storage

From chapter 2.2 "Summary of product" in the vision document (Appendix F), the Digital ID wallet should be able to "safely store different identity documents in one place". The team and supervisor decided on storing the IDs on the mobile device, as that means only the user has access to the issued ID, instead of some entity located somewhere else. Encryption was also needed in case someone got hold of the device. To be able to store the IDs encrypted on device React Native Encrypted Storage was selected. Also, as it is a module that enables encrypted storing of entities and supports TS (GitHub, 2021).

Three other libraries were considered, but they had issues. The library react-native-keychain did not work as expected as can be seen by the 145 issues it has (Oblador, n.d.). Another library react-native-secure-info had documentation that differed from the code and some flaws, such as inability to reset the secure store (Andrade, 2021). Using Expo Secure storage required redoing the React Native projects infrastructure as it ran natively and not through Expo (Expo, n.d.). However, React Native Encrypted Storage simply worked as expected, and was therefore chosen.

3.2.14 Supabase

Since the team decided to create a mock issuer, a solution for storing data was needed. The mock issuer is not the main product of the thesis. Therefore, it needed to be as simple as possible, making Supabase a natural choice. Supabase provides multiple services like file storage and authentication (Supabase, n.d.-c). Nevertheless, only the database (Supabase, n.d.-b) configured to minimum security and the automatically generated API (Supabase, n.d.-a) was used. This enabled easy viewing and altering of the data and schemas in the database through the web editor, and easily perform CRUD operations on it through their API.

3.3 Development method

3.3.1 Kanban

Since there were many components involved in the project and uncertainty regarding what needed to be implemented, the team wanted an agile development method, to gain enough flexibility to iteratively adapt to changes. Scrum was discussed but scrapped due to its team size of 7-8 members, as well as having to create extra documentation such as sprint reviews and backlogs. Instead, Kanban was chosen due to it being lighter than Scrum, meaning no restrictions on team sizes, and no required documentation other than the Kanban board. To facilitate the development, a document called "Kanban board description" (Appendix J) was created, which contained descriptions about each column in the board, and the limit for number of items at the column simultaneously. The limit helped make sure that tasks were completed by forcing focus on bottlenecks instead of adding more tasks. Therefore, the max limit to a column (excluding open, done, and closed with no limit) was twice the number of team members. Since the team consists of three members the max limit was six. As the project was hosted in a group in Gitlab, the integrated issue board was used for all the repositories inside the group. It also provided a functioning solution on the same platform as the repositories were hosted on and therefore lowered the complexity in managing the project.

As the team's supervisor and client were the same person, both the supervisor and client could be addressed during the same meeting. It was therefore decided to aim for weekly meetings to replace the sprint reviews, where both guidance about the thesis could be gained, while also updating the client about the product, as well as gaining feedback about it through demos. The team members aimed to sit together at most work hours either via online video conferences or physically, and could therefore always communicate with each other, which neglected the need for scrum's daily 15 minutes stand up meeting.

3.3.2 Experimenting through "Hello world" repositories

As some of the technologies used were new to the team, the supervisor recommended the creation of "Hello world" repositories for those technologies. There were four of them made, one for React Native, one for NextJS, one for Solana and one for Substrate. Thereby, the team could get familiar with the technologies, experiment and make crucial decisions at the start of the project instead of waiting until it is too late. E.g., testing the repositories helped the team to choose Solana over Polkadot (Substrate).

3.3.3 Wireframing in Figma

For the development of the frontend application, the team decided to create a wireframe and showcase the wireframe to the client before developing it. This way, there was room

for experimentation and visualization with a lower time cost than actual development would have created. A wireframe is also useful for the team to be able to refer to as a consensus during development. Every time a new feature was to be added to the mobile application it was to be wireframed first and showcased to the client, as well as discussed, before being implemented.

Figma was chosen as the wireframing tools for multiple reasons. One is its realistic design, meaning that Figma wireframes can look like a finished product. Another is the ability to create interactive demos through connecting screens, meaning that a press of a button on one screen can lead to another screen. Figma can also run the demos on different devices, this makes the demo more realistic, which provides more learning about the design. Lastly, Figma allows sharing of prototypes which enables a team to work on it simultaneously (Figma, n.d.).

3.3.4 Work and role management

It was decided early on that Martin had the responsibility of leading the team and conducting meetings, Rokas being responsible for writing meeting summaries and checking the quality of all documents, and Diderik being responsible for taking backups of all documents. The reasoning behind the role distribution was the team's previous experience working together, which has proved that this role dynamic has worked well. Regarding the repositories, the team decided to distribute the team members to different repositories, due to it being regarded as more efficient, since a member would become an expert on its assigned repositories. Rokas would mainly work on Iris and Kratos, Diderik would focus on Aphrodite and Iris, and Martin would develop on Aphrodite and Athena. Additionally, Rokas would be responsible for deployment, Diderik would be writing automated testing and Martin would complete manual testing. All the responsibilities had wiggle room, as the team thought that may hinder bottlenecks originating from the responsible team member being preoccupied with other tasks.

4 Result

4.1 Scientific Results

4.1.1 Ecosystem architecture

As the wallet alone would have minimal functionality, it was necessary to construct the entire ecosystem that consists of 4 components, the mock issuer (Athena), the wallet (Aphrodite), the backend server (Iris) and the smart contract (Kratos). All components in the ecosystem communicate with each other, as described in chapter 2 "Architecture" in the system documentation (Appendix H). This section describes the resulting functionalities of each component in the ecosystem.

4.1.1.1 Aphrodite

The wallet communicates with two entities, the issuer(s) (Athena) from where it will fetch its VC and a centralized backend (Iris) used to communicate with the smart contract (Kratos). After receiving a VC from an issuer, the wallet can create and display a VP, which a verifier should be able to verify. The VCs are stored in an encrypted storage. Additionally, the hash and salt for verifying the wallet owner locally, and the holder DID, used for identifying the wallet owner globally and sign VPs, are stored in the encrypted storage. The wallet is also able to serve as a verifier through the means of scanning QR-codes.

4.1.1.2 Athena

The mock issuer (Athena) is used to demonstrate what is required of an issuer to be able to integrate with the ecosystem. Firstly, it needs to be able to generate its own DID used for signing VCs. Secondly, it needs to be able to create VCs. Thirdly, it needs to send a REST call to the backend (Iris) for creating agreements. Fourthly, it needs to be able to collect the DID from a wallet during the issuing of a VC, e.g., through a query param. Lastly, it needs to keep track of the document DIDs of all issued VCs so it can revoke the VCs belonging to a user. Other attributes needed that are expected to already be implemented in an issuers business logic, are for example user logic and a login web UI able to be used in an embedded web view. Athena stores personal data about all its users to be used in VCs issued to them, such as forename. This data can vary with different issuers. Also, Athena stores data needed for authentication such as hash and salt. Additionally, Athena stores its DID used for signing VCs and registering as an issuer to the blockchain.

4.1.1.3 Iris

The centralized backend (Iris) is used as a communicator with the smart contract (Kratos). It utilizes a PostgreSQL database to store pairs, where each contain a public key and a DID. The public keys belong to the Solana accounts. The accounts are created whenever a new issuer or a new ID agreement is added to the blockchain. Each account creation requires the creation of a new keypair. The public key of the account is then stored in the backend server together with the corresponding DID. Later, public keys can be used in RPC calls to the smart contract (Kratos) to execute functions defined in the

contract regarding the account. Furthermore, Iris allows issuers to register themselves by adding them to the blockchain and storing previously named pairs to the database.

4.1.1.4 Kratos

The smart contract (Kratos) has been programmed to execute predefined functions. It is deployed on the blockchain, meaning that when a function is called, it will be executed automatically. The current smart contract MVP implementation has simple functions of creating an ID agreement, creating an issuer agreement, and invalidating an ID agreement, in addition to fetching the agreements. An ID agreement stores the holder DID, the issuer DID, the document DID, the expiration date and a valid binary. Issuer agreements contain the issuer DID, the URL to the issuer's web page for retrieving a VC (used by Aphrodite through embedded web view when adding an ID), name and a valid binary. All information stored on the blockchain is non-sensitive data.

4.1.2 Handling ID

Agreements stored on the blockchain, VCs and VPs are used for handling the IDs. The issuers in the ecosystem issue an ID as a VC to a wallet and create an agreement on the blockchain using the backend. Both the issued VC and the agreement contain the issuer DID, the wallet/subject DID and the unique DID for the ID (created when the VC is created). In addition to the DIDs, the agreement contains an expiration date and a valid argument. The valid argument can be set to false by the issuer or the user, to mark the ID as invalid. The wallet can at any time create a VPs using one or more VCs. The presentation is used to verify the VCs by first using the proof method which ensures the verifier that both the VP and embedded VCs has not been tampered with. The second part of the verification is validating all the VCs used in the VPs with the corresponding agreements stored on the blockchain. For the VP to be valid, all agreements need to be valid, not expired and the DIDs need to correspond.

4.2 Engineering results

This section describes the goals of the functional properties for the MVP, which can be found in the vision document (Appendix F) chapter 5 "Functional properties of the product". The goals were defined in the beginning of the project in consensus with the team members and the client. A few of the goals were not implemented in the MVP and will not be mentioned in this section, but some of them are discussed in the "Discussion" chapter and "Future work" subchapter. The source code for the MVP is attached to the report (Appendix K).

4.2.1 ID storage

4.2.1.1 Add ID

The goal was to create a system for adding a digital ID to the wallet. This can be linked to the security goals from vision document's (Appendix F) chapter 6 "Non-functional properties and other needs". They declare that all personal data should be stored on-device, all personal data should be encrypted, and that transactions from providers to users should forever be stored on blockchain. The technical goal of adding an ID was fulfilled, including the security goals. The digital ID is stored on-device using encrypted local storage within a VC. On the Solana blockchain, the smart contract Kratos, creates a new account for storing the agreement between the issuer and the holder. Additionally, the Solana account public key of the newly created ID is stored in a PostgreSQL database

in the backend (Iris) together with the corresponding document DID. The public key can later be used by anyone to view and verify the document, or to revoke an ID.

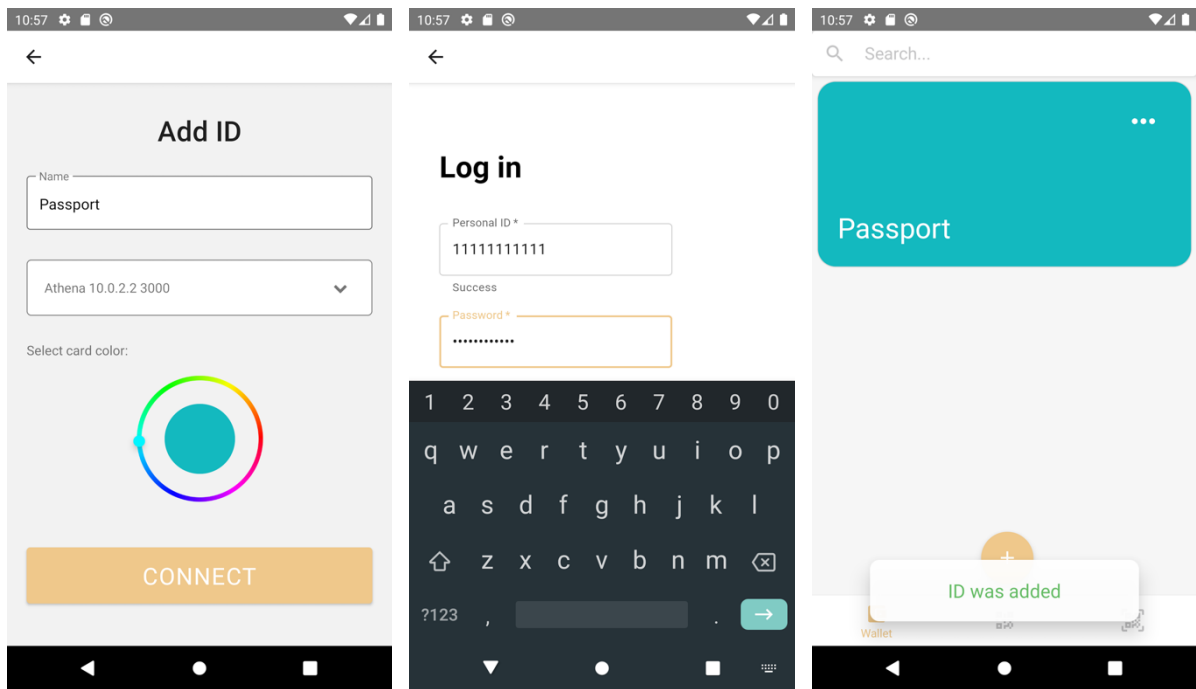


Figure 4.1: Three screenshots displaying the process of adding an ID as seen by the user

4.2.1.2 Remove ID

The goal for removing an ID was completed after the “add ID” function was implemented. The remove ID function, also called “revoke ID”, can be invoked by the ID holder or the issuer. To initiate the “revoke ID” action, the issuer or the holder must call to the backend (Iris) with a REST API call containing a document DID of the ID that is being revoked. The backend server (Iris) then uses the document DID to fetch the corresponding public key of the document’s Solana account from the PostgreSQL database. The backend server (Iris) then initiates revocation through the smart contract (Kratos) using an RPC call. The smart contract’s “revoke ID” function is automatically run, which sets the agreement’s “valid” binary to “false”. The ID card will also be removed from the encrypted local storage in the application (Aphrodite).

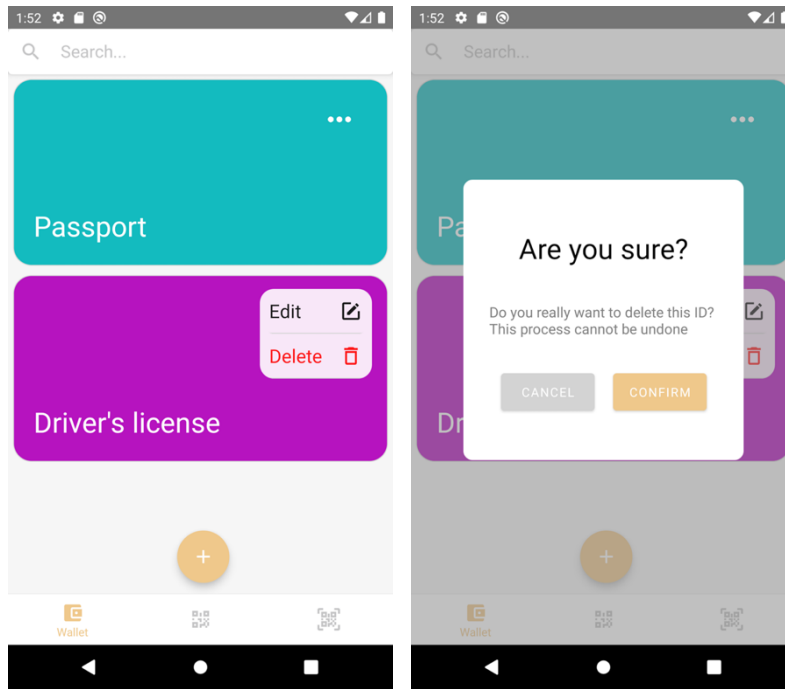


Figure 4.2: Two screenshots displaying the process of deleting an ID as seen by the user

4.2.1.3 Refresh ID

This functionality has been partially implemented. The initial idea was to let users or issuers renew or “refresh” an ID. For example, a user might have changed their name leading to them needing to renew the passport. This application feature would let users refresh their documents in their wallets by fetching renewed data from the issuer without them needing to create a new document entirely. Additionally, users could also edit their ID cards within the application, such as giving them new colors or renaming them. Only the latter functionality of this goal has been achieved, leaving the former functionality for future work. The user logic in Athena also allows editing user data.



Figure 4.3: Five screenshots displaying the process of editing an ID card from a user's perspective

4.2.1.4 Encryption of ID

The goal of encrypting ID data is important for security. This is linked together with the security goal from vision document's (Appendix F) chapter 6 "Non-functional properties and other needs", which says that all personal data should be encrypted. This was achieved by using an encrypted local storage in the Digital ID Wallet frontend application (Aphrodite), since the IDs are only saved there. There is more information about the library in chapter "Choice of Technology", subchapter "React Native Encrypted Storage".

4.2.2 ID viewing

4.2.2.1 Card display

This goal consisted of creating a view for an ID that is stored in the wallet. It was completed early in the project and can be accessed by pressing an ID in the wallet window in the mobile application. It should be displayed as the example below.

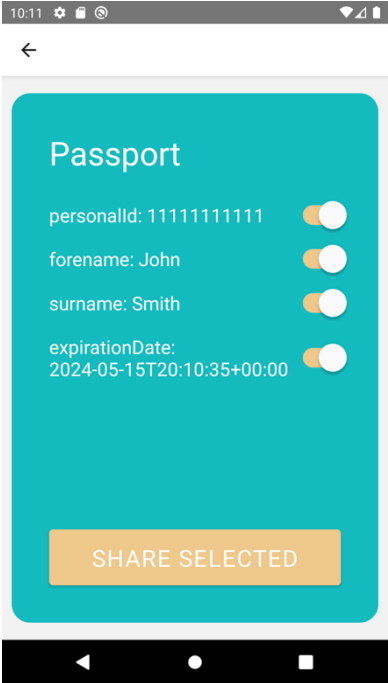


Figure 4.4: The screen displaying the ID information along with sharing options

4.2.2.2 Search and filter IDs

The application allows users to filter and search IDs based on the card name, by using the search bar. This process is depicted in the figure below.

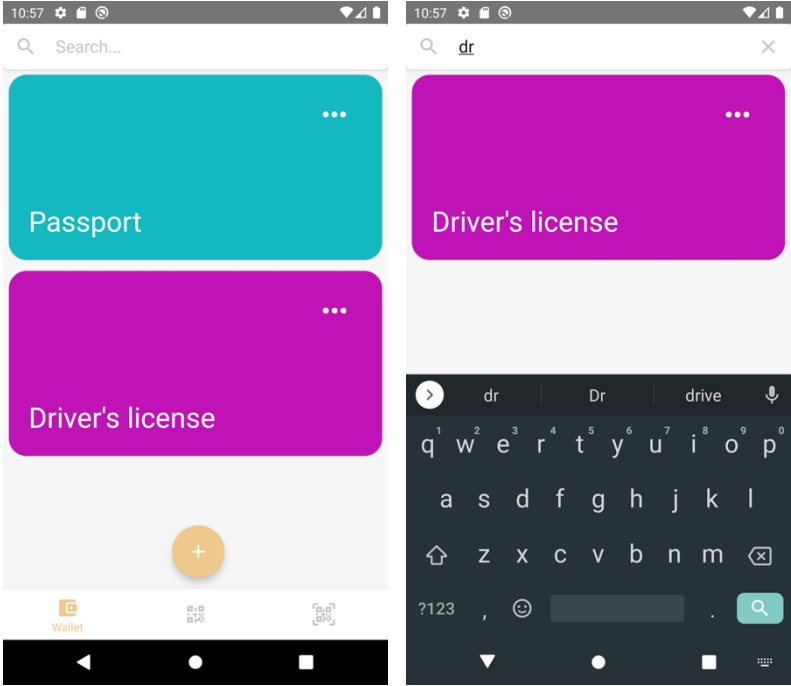


Figure 4.5: Searching with the term "dr" filters the Passport away

4.2.2.3 Scroll through the IDs

This was implemented using a Scroll View component that enables vertical scrolling if the number of IDs exceed the mobile frame.

4.2.3 Verification

4.2.3.1 Generate QR-code to prove one's identity

The feature allows users to provide a non-tampered ID to the verifier using a QR-code. This can be linked to the availability goal from vision document's (Appendix F) subchapter 3.4 "Summary of user needs". It mentions that a user should always be able to verify themselves even if the issuer provider is experiencing downtime. This goal was completed using VCs and VPs. As VCs are stored on-device, VPs can be created at any time using one or more of them. The Veramo library uses a JWT proof method that creates a signed JWT token containing the information of all VCs in the presentation. Afterwards, a user can generate a QR-code from the JWT proof token, which will be presented to a verifier.

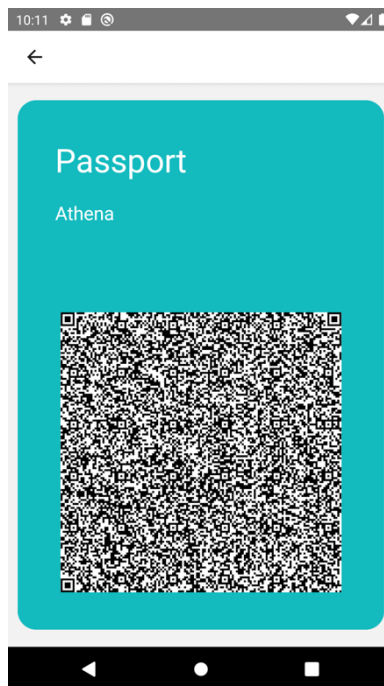


Figure 4.6: Sharing the data of a Passport through a QR-code

4.2.3.2 Verify other identities by scanning other QR-codes

This feature allows verifiers to scan other users' generated QR-codes. This can be linked to the availability goal from vision document's (Appendix F) subchapter 3.4 "Summary of user needs". It mentions that a user should always be able to verify themselves even if the issuer provider is experiencing downtime. This goal was achieved using VP and blockchain technology. After a user has generated a QR-code containing a JWT proof token, a verifier is able to scan it using the QR-scanner in the Verify window. After the verifier scans and receives the token, it begins the validation explained in the Handling ID subchapter. For the proof method, Veramo uses a signed JWT token, as explained in the previous subchapter, and validates it by verifying that the signature has not been

tampered with. Afterwards, it decodes the proof token, generating an object containing the VCs, VP and DIDs.

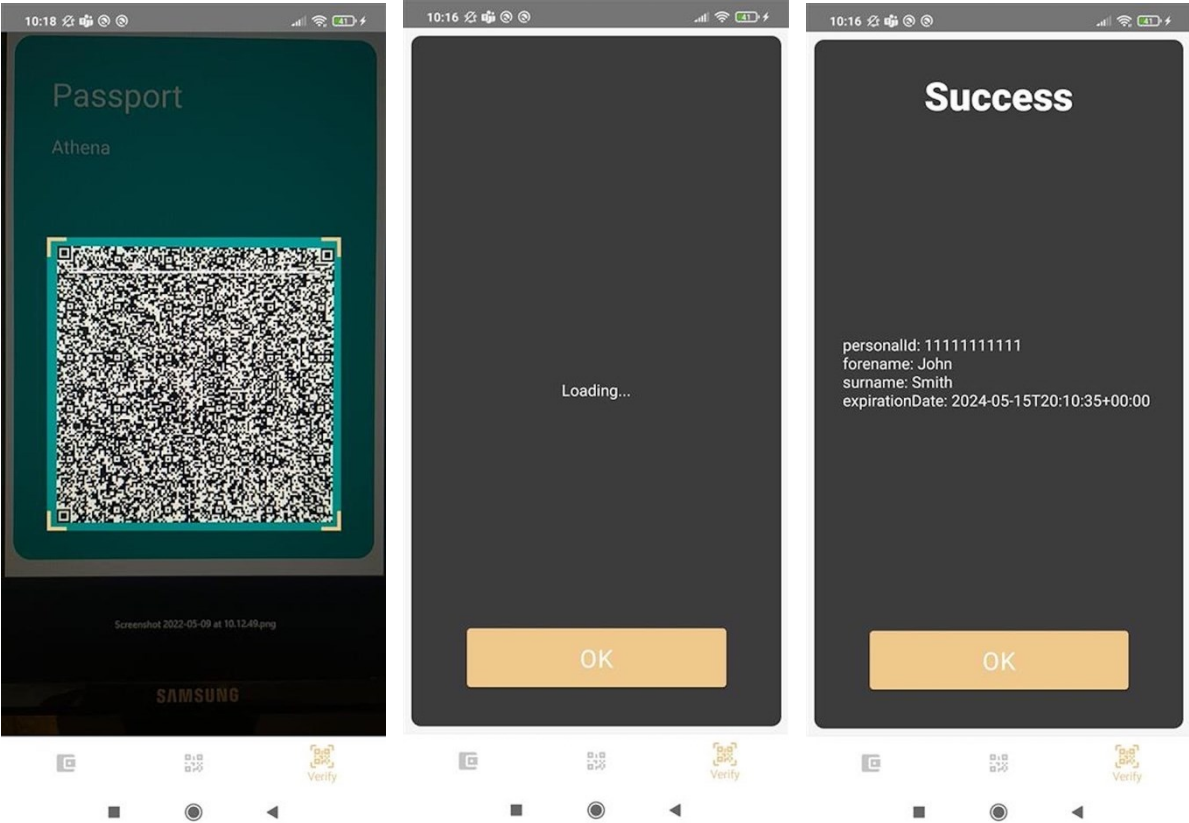


Figure 4.7: The screenshots demonstrate the process of verifying a user's Passport

4.2.3.3 Selection of ID data to be used

The idea of this goal was selecting specific data of an ID which the user wants to be shared. This goal was partially completed. When viewing an ID card, all the data belonging to the ID is shown. There are switches next to each of the data fields, which can be selected or deselected. In the current MVP, they do not have any functionality; all ID data will be used to create a VP when sharing an ID with QR code (see previous subchapters).

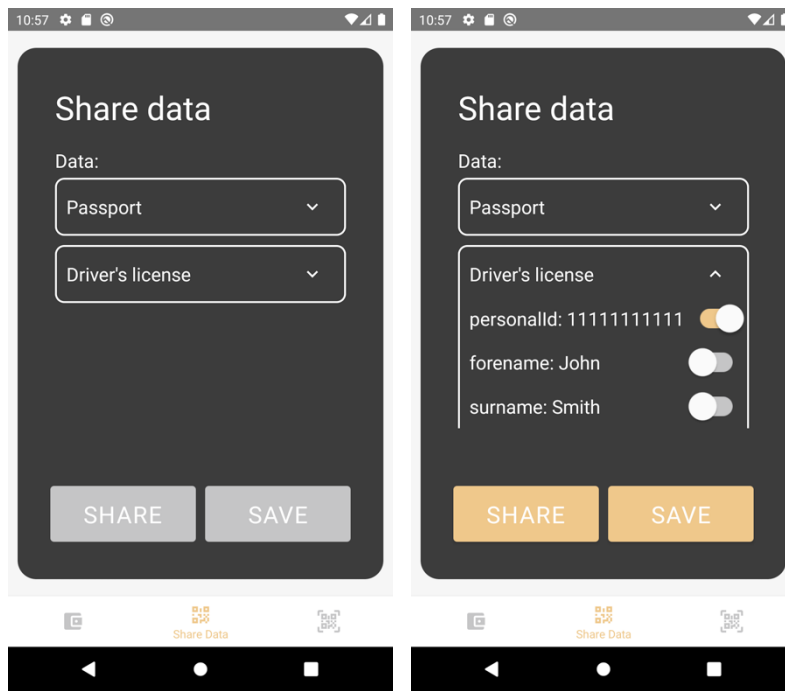


Figure 4.8: The share data screen, for combining data from different IDs

4.2.4 Authentication

4.2.4.1 Unlock wallet with password

To ensure security in the application (Aphrodite), it was planned to protect the application with a password lock. This was changed to a PIN code lock during the development, since similar applications (e.g., Vipps) use PIN codes. When a user creates a new wallet and a new PIN code, the code is hashed with salt and stored in the encrypted storage (see Appendix H). For more information about the encrypted storage see chapter "Choice of Technology", subchapter "React Native Encrypted Storage". When the Digital ID Wallet application is locked, the user must type the PIN code that was created at the start. The application is locked every time it is restarted, or when the user is inactive for a certain period (5 minutes in the current MVP).

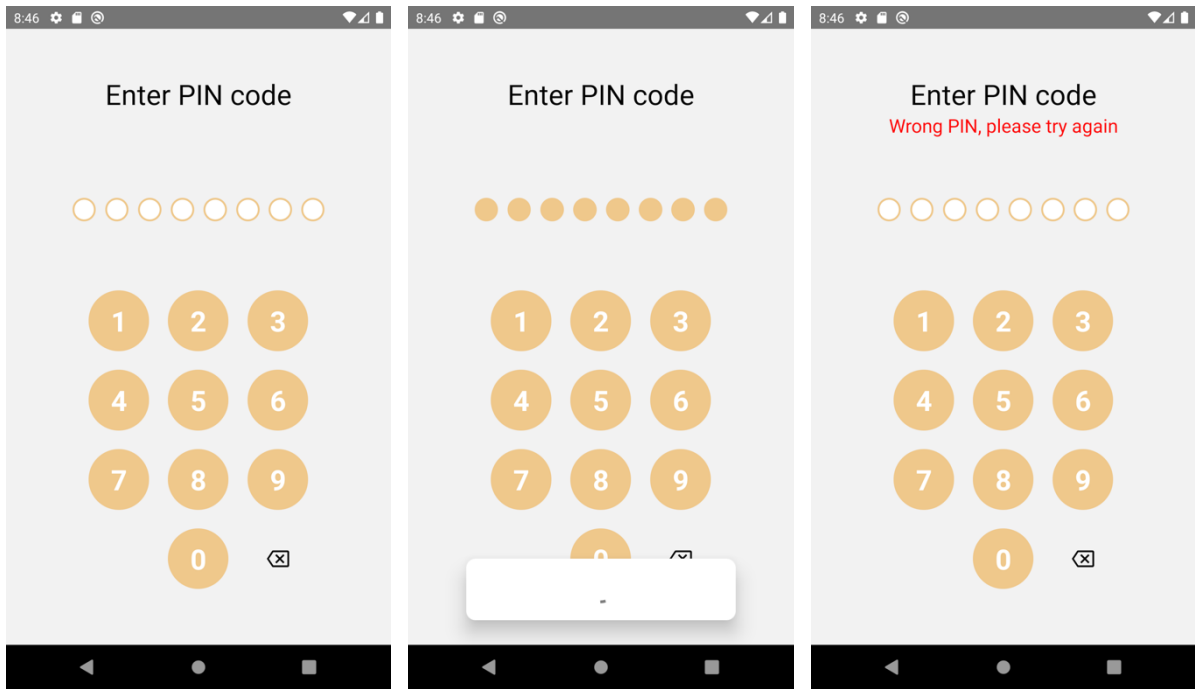


Figure 4.9: The process of entering a wrong PIN code

4.2.5 Provider registration

4.2.5.1 Register as ID provider with public key and DID

This feature allows issuers to register themselves to the ecosystem. This feature was completed by creating both a UI and an endpoint for issuers that is protected by requiring an API token. More details can be read in chapter 5 "Server services" and chapter 6 "Security" in the system documentation (Appendix H).

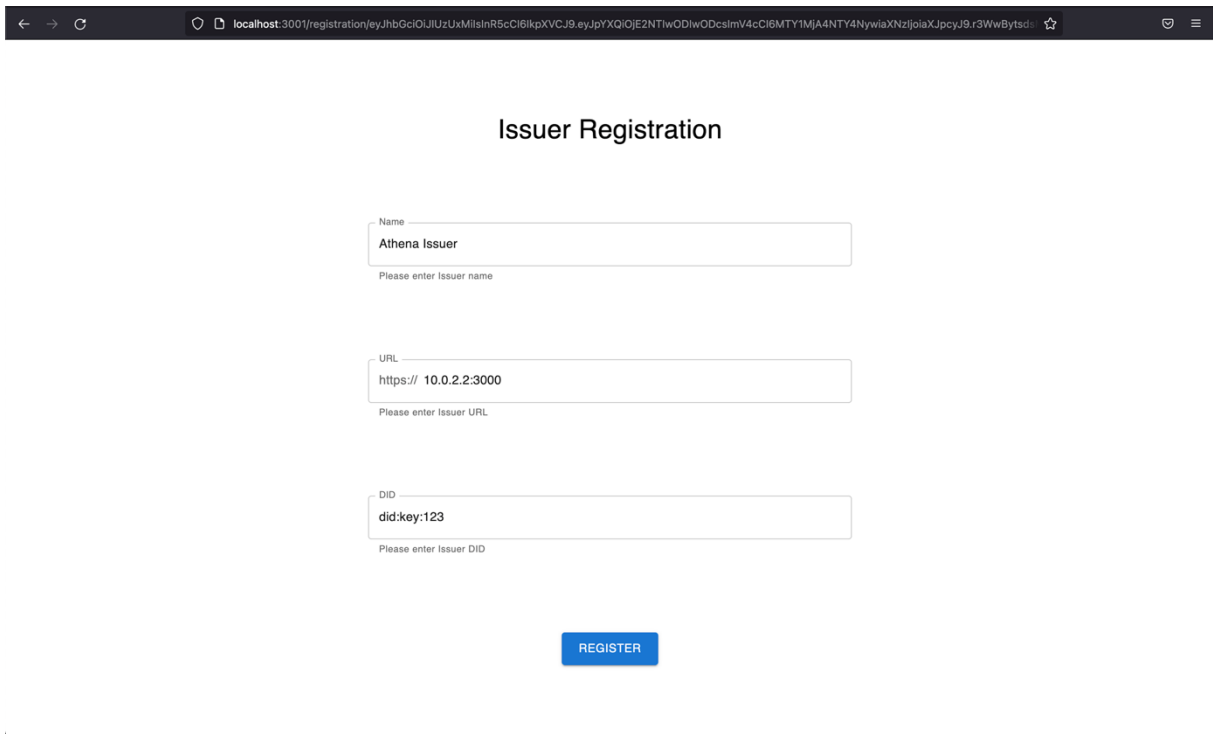


Figure 4.10: The page where issuers can register themselves to the ecosystem

4.3 Administrative results

4.3.1 Project schedule

The project tasks and use cases were scheduled using Gantt diagrams that are displayed in the project handbook (Appendix E). They were formed early in the project and continuously updated when finishing and progressing tasks. As displayed in the diagrams, a few use cases and tasks were never completed or started. Additionally, in most cases the scheduled timelines did not match the actual timelines, because they were either started too early/late, or finished too early/late.

4.3.2 Time management and activity distribution

Time management and activity distribution were documented daily and summarized weekly in the status reports that can be viewed in the project handbook (the project handbook (Appendix E). The status reports explain what each member worked on and completed that week, together with a time sheet of the activity distribution. Two pie charts were generated from the final time sheet to display the total activity distribution, and total work hours distribution of each member (Appendix A). As seen in the work distribution chart, most of the time was used on development, report writing and research. Furthermore, displayed in the work distribution per team member chart, all members worked approximately the equivalent number of hours.

4.3.3 Agile development

As the team used Kanban during the development, it was created an issue board on Gitlab (Appendix B). After the MVP was complete, it contained a total of 85 issues where 7 are still open, 70 completed and implemented, and 8 closed.

5 Discussion

5.1 Scientific

5.1.1 Ecosystem architecture

5.1.1.1 Centralized backend vs fully decentralized ecosystem

Regarding the system being sustainable against security breaches the centralized backend may serve as a weak link, which can be utilized as an attack vector for potential threat actors, since hijacking the backend would mean having full control over the agreements being created. Being able to make the system fully decentralized would mitigate this issue, by changing the most valuable asset for a threat actor from the centralized backend to an issuer, meaning that only the agreements of the hijacked issuer could be manipulated instead of all agreements. However, the backend does provide value regarding the ability to offload the overhead from managing keypairs when communicating with the blockchain. Nevertheless, security should always be a priority therefore a decentralized ecosystem should probably be pursued, making the result of the project discouraged from being deployed in its current state.

5.1.1.2 Mock issuer

In the result section it is described how the mock issuer Athena has been implemented for the ecosystem to function. As the issuers in this potential ecosystem already exist, and may have varying systems, the implementation poses a potential source of error regarding the result. To clarify, the result is based under the assumption that different issuers can adapt their systems to the ecosystem, and that it is worth it for them. These changes are showcased in the chapter Result, subchapter Athena. If either of those predicates fail the result is invalid, as there will be no available issuers in the system.

5.1.2 Handling ID

5.1.2.1 Blockchain agreement model

The agreements stored on the blockchain were used to prove that an issuer has agreed and given a VC to a user. The agreement therefore contains both the issuer's DID and the user's DID. To specify the agreement terms, the VC's DID is added to confirm what is agreed upon, which is the issued credential. When a verifier fetches the agreement, it checks that the received VC has an agreement stored on blockchain and verifies it. An existing security vulnerability in the ecosystem allows everyone to add an agreement to the blockchain. This allows counterfeit agreements, where users can issue their own VC containing false information and add an agreement on the blockchain with the corresponding DIDs. The verifier would not know if the agreement was added by a malicious issuer or not. A solution to this vulnerability would be to add authorization, which would remove the possibility of users adding malicious agreements, because only registered issuers are able to authorize. Another solution could be to register an issuer with a public key, and store agreements, signed by the issuer's private key, on the blockchain. On verification, the verifier would fetch the public key from the registered issuer and decrypt the agreement. This would remove the vulnerability by confirming

both the issuer has been registered, and that the agreement was signed and created by the issuer.

5.2 Engineering

5.2.1 ID storage

One of the most important features of a digital ID wallet is the ability to store IDs in a secure and trusted yet easily accessible way. As mentioned in the "Result" chapter, the resulting product stores IDs in an on-device encrypted local storage, while the agreements between issuers and holders are stored on the Solana blockchain. By storing a digital ID on the holder's device, the system gives the holder full ownership and control over the ID. However, malicious holders could tamper with the ID data without the issuer's approval. VC technology was used to prevent tampering. This technological choice resulted in creating trust in the holders without any other party controlling the holder's data. Issuers can only control the agreements' validities between themselves and the holders, which are stored on the blockchain. By not storing IDs on the blockchain, the system does not expose personal data, ensuring confidentiality, as all data on the blockchain is public. The agreements are only used as "proof" that a holder's VC is valid.

By utilizing DLT, the Digital ID Wallet would ideally provide a constant service with no downtime. Meaning that an ID holder can use an ID for verification even if the issuer's services are unavailable. Since all the agreements are stored on the blockchain, a verifier can find the "proof" by fetching data from the blockchain. Nonetheless, the current product has a centralized backend server (Iris) which is used to connect holders, issuers, verifiers and the blockchain. Ideally, the system could be a fully decentralized ecosystem (with no centralized backend server). However, there will currently be downtime if the backend server experiences issues.

The DLT also ensures that the data stored on it is tamper proof. This can be viewed as a necessity for the system to function properly, since this hinders forging of agreements, e.g., extending the expiration date of an agreement, or making it valid again after it has been revoked. One could also change the holder DID thereby changing who owns the agreement. Regarding the issuers stored on the blockchain tampering could change the public keys for a known issuer. These public keys are used for verifying a VC, meaning that a threat actor could change it to its own keypair and therefore sign VCs, while proving they are issued from a known and trusted issuer. The URL for the issuer could also be changed, prompting the users to a malicious web site when adding a new ID, with the purpose of stealing the user's authentication data, e.g., username and password. All these potential pitfalls demonstrate the importance of the tamper proof ability of the blockchain.

The centralized backend server has several important use cases. Firstly, it offers an interface and API endpoints for ID issuers to register themselves on the ecosystem. This creates a common pattern in which the issuers can "sign up" to the ecosystem. Secondly, the Solana account public keys are stored in a centralized PostgreSQL database. The keys can then be used by anyone in the ecosystem to fetch agreements from the blockchain. However, the public keys can also be used to invalidate or "revoke" an ID. This is done by calling the smart contract with a Solana account's public key, which in turn will set an agreement's "valid" binary to "false", making the agreement invalid. This means that anyone can invalidate anyone's ID agreements on the blockchain by calling

the correct API endpoint with a correct document DID. This is not an intended feature, which can be exploited by malicious attackers.

Another problem with the result was that the refresh ID feature was not fully implemented. The user can update the appearance of the ID card. The ID data can, however, not be refreshed. During the development process, this feature was less prioritized because of time constraints and prioritization of more important features.

Transferring a VC from the issuer was done using a WebSocket connection. The WebSocket server opens after the issuer has created a VC and added an agreement to the blockchain. The receiving wallet will then connect to the server, collect the VC, and emit a "received" message. After the server receives the message, it closes. This should also happen on subsequent requests, however, after closing the server on the first request it cannot reopen. This means the entire Next JS server must restart for another wallet to receive their VC. The server closes because the function for initializing the server uses the VC as an argument, since it is to be transferred to the first client socket that connects to the server. This feature has security vulnerabilities, by allowing the first client socket to get the VC, anyone trying to connect to the WebSocket endpoint can receive it. However, an improvement would be an endpoint known only by the server and the user, for instance using the holder's DID in the URL path or authorization header. Malicious sockets would then not know the WebSocket address and therefore not be able to steal VC. A second option would be to change the protocol, as WebSocket is mainly used for multiclient-server communication and not peer-to-peer. A peer-to-peer connection would also remove the security vulnerability explained above, by creating a secure connection only between the server and client. Regardless, WebSocket had multiple JavaScript libraries for implementing the functionality and as the team had previous experience using the protocol it was selected contrary to another protocol.

5.2.2 ID viewing

To make the application accessible, the user interface had to be easy and convenient to use. Simplistic ID card displays represent a physical wallet with physical cards. This provides the user with a sense of familiarity, making the transition from a physical wallet to a digital ID wallet easier. Scrolling and filtering of ID cards are also simple mechanisms that ease user experience and are familiar concepts from other applications.

There is a subgoal from the vision document (Appendix F) section 5 "create access history for each ID" which was not implemented. During the planning process, this subgoal seemed achievable and useful, but while developing and researching, it was not prioritized. Firstly, it could be space consuming to save the access history of an ID. There would be necessary to use a separate database or another storage space for saving timestamps and other metadata for an ID's usage. In fact, since the IDs are only stored in the holder's on-device storage, and since mobile devices often have a very limited storage space, the access history could relatively use up a lot of storage space if saved locally. Secondly, saving access history would defeat the purpose of confidentiality. The solution is built on the concept of a decentralized system, where there is no centralized entity controlling the system and collecting user data. By implementing a feature to save access history, the verifiers could potentially exploit it by requesting ID holders to share ID access history, thus creating a way of collecting confidential user data. Lastly, the feature would not provide much information to the user as they already are responsible for initiating the verification process, and therefore already know the data, time and

verifier involved in the process. These arguments show why the subgoal has not been prioritized in the MVP development.

5.2.3 Verification

QR-codes were simple to implement because the team decided to use a React library for generating and displaying them. However, a few of the libraries were unoptimized and used extensive time to generate the codes. This was also caused by the large proof token generated by the VP. By the end the team found an effective and working library, that generated the QR-code in a user-friendly time.

The VC standard was utilized for displaying the IDs in a verifiable manner. The Veramo library used for this purpose might, however, have contained a security breach. It uses a JWT method in the proof property, which should be signed and verified by a secret, however, Veramo never documents what secret it uses to sign and verify the JWT token. If it uses a public secret (likely), anyone can tamper with the proof property by decoding it, adjusting the data, and signing it again. However, by using the DID resolver, Veramo could be recreating the DID's public key. This would allow the verification of the JWT token if it is signed by the DID's private key. Moreover, the proof property provided by Veramo does not contain a public key, meaning if a different private key than the DID's private key was used to sign the JWT token, there would be no way of decrypting the signature. The team decided not to extend the asymmetric signing functionality of Veramo to ensure that the proof property works as there are only a few encryption libraries implemented for React Native and it would probably have wasted resources and time. A proposed solution would be to sign a JWT token with the DID's private key that Veramo generates when the DID is created. In the proof of the VC/VP add the JWT token with the DID's public key. This would allow the verifier to verify the signature and decode the token to check that no tampering has happened with the credentials.

An error occurs when trying to display a JWT proof token that is larger than the allowed limit for QR-codes. Hence, the Share Data window in the application is useless until a better transfer method is implemented. Therefore, a transfer protocol that allows for larger amount of data is needed to allow users to combine VCs in a VP. Additionally, QR has multiple JS libraries for implementing it, which reduced the time spent developing the physical verification. The selection was also based on not knowing what the QR-code would contain as the VC ecosystem has multiple proof methods, for instance, RSA signature, which perhaps would require less data to be transferred from the user to the verifier. Therefore, solutions would either be attempting a different proof method when generating a VP, choosing a different transfer protocol or compressing the proof token.

The selection of ID data to be used in a VP, was, as mentioned in the result chapter, not fully implemented. Veramo had implemented Selective Disclosure Request, however, researching and reading source code uncovered that it did not use any cryptographic operations or zero-knowledge proof to generate the VP. Instead, it only linked to the VC, removing the purpose of Selective Disclosure, which is combining data from different credentials without exposing the entire credential. Therefore, to implement it into the ecosystem would require extending the Veramo library. This would require adding cryptographic operations for zero knowledge proof which manually would have required time and resources, or use a cryptographic library, however in React Native none of them had the necessary functionality. Therefore, the team decided to use the time and resources on other implementations.

The feature “Websites and apps can request identification from the wallet” defined in the vision document (Appendix F), would allow other mobile applications or websites to use the Digital ID Wallet for identification. The feature would resemble using for instance Vipps as a payment method when shopping online, however instead of payment, the Digital ID Wallet would be presented as an identification option. This feature had a low priority, resulting in it not being implemented in the MVP.

The last subgoal “Users must authenticate themselves before they can share their IDs” in the vision document (Appendix F), was a feature that was supposed to be added to increase the security of the mobile application and decrease possibilities for identity theft. For instance, if a user leaves their device logged into the application, the thief should not be able to share the application’s IDs without passing the PIN code. The feature would resemble how Vipps secures transactions using biometrics or PIN code. Even though the PIN code was created as a React component with the necessary functionality, it was never implemented to function before each time the user tries to share ID data.

5.2.4 Authentication

An open source React library was first used to create the PIN code authentication scene for the Digital ID Wallet. The library turned out to be very slow and unresponsive, pushing the team to create their own implementation of the PIN code authentication system. As mentioned in the “Result” chapter, the system stores the hashed PIN code safely and securely in the encrypted local storage. Moreover, the PIN code interface itself is responsive and user friendly. The typical PIN code systems in other applications have 4-digit PIN codes, whereas the team chose to create 8-digit PIN code. This allows for 10000 times as many PIN combinations. Since the Digital ID Wallet application operates with digital IDs containing personal user data, it is important that the system is highly secure. It is also important that the security measurements do not negatively affect the user experience. An 8-digit PIN code increases security against e.g., brute force attacks by increasing possible combination amount, while retaining a relatively quick authentication process, contributing to a positive user experience.

To increase security even further, the Digital ID Wallet application has a time limit for inactivity. If a user is inactive for 5 minutes, the application will be “locked”, prompting the user to the authentication scene. The idea behind this mechanism is to possibly prevent malicious users from accessing the wallet in case the holder of the wallet leaves the application unlocked. Additionally, if a potential attacker tries to access the wallet, but types the incorrect PIN code multiple times, the application will be inaccessible for a period. This mechanism was also implemented to reduce chances of brute force attacks.

It was planned for the Digital ID Wallet to be password and biometric data protected instead of PIN code. The initial goal was to create a password system, since passwords are most often more secure than PIN codes (because of length variety and character quantity). Additionally, a biometric authentication mechanism was also thought to be an alternative to password authentication. During the development process, the team decided to switch to PIN code authentication, since similar applications such as Vipps already use a PIN code system and they are simpler to use and remember. However, biometric data would probably be the most secure and user-friendly option.

5.2.5 Provider registration

The API key was implemented to ensure only valid and not malicious issuer could be registered to the system. Therefore, administrators are the only ones authorized to fetch

and distribute an API key for an issuer that has requested to be a part of the ecosystem. To access the API key generation endpoint, the requester must use the administrator username and password in the authorization header using Basic Authentication. The reason for only allowing valid issuers is removing the opportunity for users to register their own issuer and issuing counterfeit VCs that could contain false information. However, as discussed under the chapter "Discussion", subchapter "Verification", everyone can add agreements on the blockchain and therefore one can still counterfeit VC.

5.3 Administrative

In retrospect Kanban was a good choice for development method as it allowed the team to focus mostly on programming the ecosystem. This focus was needed as the team barely managed to get a functioning MVP together before their set deadline. Having other development methods that required more documentation would have likely meant that the team would not have been able to complete an MVP. By having weekly meetings, the team was able to iteratively adapt to new wishes from the client and guidance from the supervisor. These meetings were also a valuable tool regarding understanding the system, as the team had to explain the current implementation to the client and thereby reflect on what has been created, and if it can be improved.

By experimenting with Hello World repositories for the new technologies the team was able to uncover fundamental errors in the technology to be used, early in the development process. E.g., Polkadot and Substrate not working as mentioned in the method chapter. After the switch to Solana and Anchor, the team was also able to confirm that the different repositories would be able to communicate with each other through the Hello World repositories, this along with gaining some experience with the new technologies, probably brought needed stability and familiarity when the team started implementing the product, which may have increased the productivity.

By Wireframing in Figma the team was able to easily explore different designs without development overhead and were also able to create demonstrations of the application's proposed flows to the client, thereby gaining feedback before starting the heavier process of implementing the proposed features into the application. This certainly spared the team time that would have been wasted on implementing features or flows in the application that would have been scrapped.

The decision to use diagrams helped the team and supervisor reach consensus regarding the ecosystem's different processes. Often there would be confusion during the meetings, but the diagrams would help clear the confusion. When developing, the diagrams were utilized for giving the team an overview of the ecosystem, which probably sped up the process as one could easily see the connection between the general overview and the detailed implementations.

5.4 Group reflection

As mentioned in the development method subchapter the team aimed to sit together most of the time, and this was accomplished. Having other team members available to help during most work hours probably raised effectiveness as one could get an answer without having to do research. This also allowed the team to plan and discuss solutions together, in turn creating better solutions or improving existing ones. Along with the Kanban board this gave everyone a better overview of what every team member was

working on at a given time. One issue was that the team neglected to plan code structure too much, which caused some confusion due to the assigned repositories having different coding styles. Another contributor to this may have been the time pressure felt by the team due to MVP being huge, meaning that the team were not too thorough with the code in merge requests if it worked. The team having fixed work hours probably helped the team with working the necessary amount required by the project, while also giving all team members a feeling of a fair work distribution, thereby increasing team morale.

5.5 Veramo

To manage VCs, VPs and DID's, the team decided to use the Veramo library as mentioned in the "Theory" chapter. Even though it was selected above implementing a library and using Affinidi, the team still experienced problems. Firstly, Veramo is a library in public beta, resulting in some of the functionality being absent or containing errors. The first problem that occurred was understanding and using the documentation that Veramo provides developers. They divide the library in multiple plugins that add functionality to the implementation. As a result, finding documentation for the desired functionality was cumbersome. An example was setting up the Veramo Agent, which they explain in a subchapter on their website, however it never mentions how to create it programmatically (Veramo, n.d.-c). Additionally, after finding a tutorial on how to set it up, it never showcases how to extend its features or how to add plugins to it, for instance, how to add VCs functionality. On the other hand, the documentation explains the expected arguments and their types, which simplifies the use of the functions. On the contrary, the documentation fails again to explain what it returns and what happens, which resulted in the team having to read the source code. Secondly, the library has a confusing API for verifying VCs and VPs. It uses messages and message handlers, for validating and decoding proof tokens (Veramo, n.d.-b) instead of providing developers simple functions for validation and decoding of JWT proof tokens and validation of VCs/VPs. Furthermore, it returns an object that resembles a message which provides unnecessary information. Regardless, the handlers do provide the necessary functionality and work as expected. Lastly, the community is small because VC is a new concept, therefore, the library was not developed enough for its size. Consequently, the library contains errors and does not provide developers with the ability to adjust functionality by providing them with options. An example was Veramo not providing the opportunity to use the already implemented storage functionality in the mobile application for storing DID's and keys. The team had to implement the storage, as the provided storage classes only used memory or database storage. Moreover, this problem could also have been solved, had Veramo given options for returning the DID's private key, which was never provided to the developer. Additionally, because of the small community, the problems the team confronted had never been seen before and therefore, to solve them, the team had to debug the library's source code to find alternatives. To summarize, Veramo wasted time that could have been spent on resolving bugs and implementing other functionality.

6 Conclusion and future work

6.1 Conclusion

The problem statement from chapter 1 was to create an MVP for a digital ID wallet application. It was an open task, meaning that the team was free to choose the technology themselves to develop the application. Based on the conceptual ideas described by the European Union, the team decided to create a decentralized solution using blockchain. The team was driven by the thought of creating something new, while also learning about new technologies and concepts. Since the team did not have prior experience with blockchains, it was also decided to create a centralized backend server to help with the development.

Scientifically, the MVP was developed as an ecosystem that consists of issuers, mobile application, a backend server, and a smart contract (blockchain) that each store different data and work together to provide users the ability to fetch, use and display their IDs, and verify other's IDs. To ensure that the IDs are not tampered with between entities, the VC standard and blockchain technology were used. When implemented, issuers will transfer VCs to holders which present them to verifiers by generating VPs. In addition, an agreement is stored on the blockchain containing the DIDs from the issuer, holder, and VC, which removes the vulnerability of counterfeiting IDs. The standard prohibits tampering by using signatures, while the blockchain is inherently immutable.

Implementing and developing the MVP consisted of fulfilling the functional properties described in the vision document and adding the functionality for the use cases defined in the requirement document. Some examples of functionality consist of adding an ID to the wallet, sharing an ID, or invalidating it. These functions had many underlying technicalities that also had to be implemented e.g., creating VCs, generating QR codes consisting of VPs, or fetching blockchain accounts by contacting the smart contract. For users, the implementations are visualized by a mobile application that communicates with the blockchain through the backend, and stores the IDs encrypted on device. In the report, it was discussed how the MVP can be updated in the future by removing the backend, since ideal solution would be to create a fully decentralized ecosystem.

Administratively, the team used Kanban for agile development. A less agile development method would have slowed down the development process, contributing to a less functional MVP. Weekly meetings with the supervisor, wireframing and diagrams helped the team shape a viable product and adapt the process to changes and difficulties.

6.2 Future work

To allow users to create VPs by selecting claims from different VCs without referencing them, Selective Disclosure must be implemented. This can be done by building a library with the necessary functionality or finding an alternative library to Veramo. This will consist of implementing zero knowledge proof. Otherwise, waiting for Veramo to further develop their library and Selective Disclosure plugin is necessary.

The current implementation only allows one VC to be sent before restarting the Next JS server and it contains security flaws as explained in the discussion. Therefore, changing

the protocol for transferring the VC from the issuer to a wallet should be done. The proposed solutions are either improving the WebSocket implementation by allowing multiple VCs to be sent without restarting the Next JS server and ensuring that the credential is sent to the requester. Another option is using a separate protocol that is intended for peer-to-peer connections.

Combining two VCs creates a VP with a proof token larger than the maximum limit of a QR-code. This prohibits users from combining VCs. A proposed solution could be selecting a different communication protocol like Bluetooth or Apple's Airdrop for transferring the proof token. Also, it could be possible to compress the proof token or use a different proof method that allows for less data being sent.

The current application only allows for physical identification, and therefore there is no integration with websites and other mobile applications. Implementing an integration would allow external companies to use the Digital ID Wallet's functionality for identification in their applications.

The current MVP of the Digital ID Wallet is dependent on the backend, which handles connections and communications between holders, verifiers, issuers, and the smart contract. A feature goal could be to fully decentralize the system by removing the backend. To achieve this, the issuer and the wallet applications can be made into dapps. This would most likely also require a new smart contract or upgrades to the current one, which would run the backend code. Then the user's Digital ID Wallet dapp could communicate directly with the smart contract, eliminating the need for a centralized backend. Issuers would need a dapp for registering themselves to the blockchain, and to integrate their services with the blockchain. The Digital ID Wallet system itself would not have a centralized backend, creating a fully decentralized ecosystem.

It was described in the chapter "Discussion", subchapter "ID storage" that Solana account public keys are stored in a database in the backend, and that they can be used to "fetch" and "revoke" agreements from the blockchain accounts. It was also mentioned that the feature of revoking an agreement with a public key should not be possible. Regarding the security risks linked to this bug, fixing it should be a priority in the future. This can be solved in several ways. The first solution could be to add roles based on DIDs, which then could be used as an authentication method on the smart contract before revoking an agreement. The second solution could be to privately distribute the account keypairs to both the issuers and the ID holders after agreement creation. This means that only the issuer and the holder of a specific document would hold the private key to the specific account, giving them full access to revoke an agreement. Both solutions require that the smart contract is updated to perform checks and verifications of the party that is trying to revoke an agreement. Only the issuers and the holders of an ID should be able to revoke (or perform any other changes to) the agreement of the ID.

Biometric authentication could be implemented into the application in addition to the PIN code. Biometrics are safer than PIN codes since biometric data is unique for each person. Moreover, authentication should happen when accessing or sharing ID data, and not only when opening the application. Since the Digital ID Wallet works with sensitive user data, the application must identify if the user accessing or sharing the data is a non-malicious user. This can be achieved by making the user authenticate with PIN code or biometrics before each access or share of the ID data.

Social impact

The IEEE Computer Society code of ethics principle 1.03 states that one should only approve software if one has a “well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment” (IEEE, 1999). There are some aspects to be discussed regarding the application and privacy. Since the new components of the ecosystem only store personal data on the owner's device it should be compliant with ethics regarding privacy. However, as the ecosystem has not been thoroughly tested for security holes, personal data could potentially be leaked which would violate principle 1.03.

The result of this thesis is an MVP for a digital ID wallet application. The scientific process creating the MVP did not include user testing. Therefore, one needs to be cautious when drawing conclusions from the results. The results suggest that a similar ecosystem can be created and that it works, however, it does not imply anything regarding the usability or demand for such a system, and therefore nothing about whether the system should be implemented.

To ensure that the results are scientifically correct, the results should be falsifiable. The manual tests (Appendix I) contribute towards proving that the results are scientific, by providing step-by-step manuals for reproducing results and falsifying them. Such proof mechanism is a strength regarding the scientific process of this thesis.

Another strength of the scientific process is passing the results on to others. Since the result is an MVP, someone will have to pick up the knowledge that was gained throughout the thesis to create a finished product. The different system-, requirement- and project documents aim to communicate the information to the readers necessary to understand the entire ecosystem that has been created. Additionally, “readme” documents, installation manuals and manual tests document aim to describe closely what each part of the system does, and how they can be set up and tested. Lastly, this report also contributes to communication, providing the necessary theory, results, discussion, and suggestions for future work.

Sustainability

The United Nations’ SDG 9 is about industry, innovation, and infrastructure. One need it specifies is improving information technology (United Nations, n.d.-c). By outsourcing identity management to designated ID providers, services can reallocate resources used for authentication. Meanwhile, designated ID providers create more secure authentication than service providers due to it being their source of income. Digital IDs also have the benefit of allowing holders to prove their identity digitally, enabling distance verification, thereby adding functionality to the digital infrastructure, and streamlining the overall infrastructure of a society. E.g., verifying age when online shopping. A service needing to integrate with multiple ID providers having different systems, however, can negate the benefits of digital IDs. Nevertheless, a digital ID wallet ecosystem might solve this problem, by providing a common interface for services to integrate with all available ID providers. Additionally, it will hopefully create a better user experience which encourages users to adopt the more secure system. The potential of the ecosystem to create a more secure and functional digital infrastructure as mentioned above, might be able to contribute to the United Nation’s SDG 9.

During the COVID-19 pandemic, health has promoted the use of COVID-19 certificates, demonstrating that the holder has been vaccinated, and thereby allowing access to services. By being able to digitally distribute the certificates, governments have been able to hinder mass gathering of people collecting physical certificates. This has displayed the potential of digital IDs in the health sector, where they can be used for future pandemics or health related claims about individuals. As mentioned in the paragraph above, the Digital ID Wallet ecosystem might help popularize digital IDs by easing the use for users and service providers. The United Nation's SDG 3 mentions a need for improved health services, both nationally and globally (FN, 2022). Since the ecosystem might popularize digital IDs which can be utilized to streamline health services, the ecosystem may benefit the SDG 3.

The Digital ID Wallet solution utilizes blockchain technologies to make the ecosystem decentralized. As mentioned in the theory chapter, a blockchain uses a consensus mechanism. Depending on which blockchain and consensus mechanism is utilized to implement a digital ID wallet solution, it could have negative environmental consequences. For example, Bitcoin with PoW. This is mentioned as energy consuming in the theory chapter which with the domination of non-renewable energy in the world can contribute to environmental catastrophes. An article from "Nature Climate Change" from 2019 mentions that "the computer processing power needed for the Bitcoin network alone could result in a global temperature rise of 2 °C by 2050" (Howson, 2019). This counteracts the United Nations' SDG 13 of hindering climate change (United Nations, n.d.-a). On the other hand, there are other blockchain technologies with more economic and environmentally friendly features. For example, the "proof of stake" consensus mechanism uses less than 1% of the total energy used by PoW (Howson, 2019). Another example is PoH, which as described in the theory chapter is much faster than PoW, thus also being more energy efficient.

"Conflict, insecurity, weak institutions and limited access to justice remain a great threat to sustainable development" is used to describe the United Nations' SDG 16 (United Nations, n.d.-d). One kind of injustice humans face is identity theft. By collecting all identities in one place, the mobile application helps its users keep track of their identities, thereby minimizing the risk of losing one. Also, the application protects identities with a PIN code, which lowers the risk of an identity thief exploiting the ids from a stolen device. Additionally, a user can always call an issuer which can digitally revoke stolen IDs on behalf of the user, making them worthless. All these features of the ecosystem help prevent identity theft, which contributes to SDG 16.

The United Nations' SDG 17 states that "A successful development agenda requires inclusive partnerships – at the global, regional, national and local levels" (United Nations, n.d.-b). Digital IDs can potentially benefit international collaboration, since an international digital ID could be created, allowing for more fluid transfer between countries, and thereby streamlining communication and exchange of goods and services. As mentioned regarding SDG 9, the Digital ID Wallet ecosystem might help ease the use of digital IDs for both ID holders and service providers, and thereby help promote the use of digital IDs, in turn allowing a potential increase in global partnerships, thus helping realize the SDG 17.

References

- Affinidi. (n.d.). *Welcome to Affinidi*. Retrieved May 2, 2022, from Affinidi:
<https://www.affinidi.com/>
- Amazon Web Services. (n.d.). *What is Decentralization in Blockchain?* Retrieved April 29, 2022, from Amazon Web Services:
<https://aws.amazon.com/blockchain/decentralization-in-blockchain/>
- Anchor. (n.d.). *Introduction - The Anchor Book v0.24.0*. Retrieved May 18, 2022, from Anchor: <https://book.anchor-lang.com/>
- Andrade, M. (2021, June 28). *Overview*. Retrieved May 4, 2022, from React-Native-Sensitive-Info: <https://mcodex.dev/react-native-sensitive-info/docs/>
- Christidis, K., & Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access, IV*, 2292-2303.
doi:10.1109/ACCESS.2016.2566339
- Dr. Wood, G. (n.d.). *Polkadot Whitepaper*. Retrieved May 2, 2022, from <https://polkadot.network/PolkaDotPaper.pdf>
- Ethereum. (n.d.-a). *Introduction to dapps*. Retrieved May 13, 2022, from Ethereum: <https://ethereum.org/en/developers/docs/dapps/>
- Ethereum. (n.d.-b). *Introduction to smart contracts*. Retrieved May 2, 2022, from Ethereum: <https://ethereum.org/en/developers/docs/smart-contracts/>
- European Commission. (2021, June 3). *Commission proposes a trusted and secure Digital Identity for all Europeans*. Retrieved May 17, 2022, from European Commission: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_2663
- Expo. (n.d.). *Expo secure store*. Retrieved May 4, 2022, from GitHub: <https://github.com/expo/expo/tree/main/packages/expo-secure-store>
- Figma. (n.d.). *Free Prototyping Tool to Create Clickable Prototypes*. Retrieved May 4, 2022, from Figma: <https://www.figma.com/prototyping/>
- FN. (2022). *God helse og livskvalitet*. Retrieved May 16, 2022, from FNs bærekraftsmål: <https://www.fn.no/om-fn/fns-baerekraftsmaal/god-helse-og-livskvalitet>
- Frankenfield, J. (2022, April 27). *51% Attack Definition*. Retrieved May 15, 2022, from Investopedia: <https://www.investopedia.com/terms/1/51-attack.asp#toc-how-likely-is-a-51-attack-against-bitcoin>
- GitHub. (2021). *React Native Encrypted Storage*. Retrieved May 4, 2022, from GitHub: <https://github.com/emeraldsanto/react-native-encrypted-storage#readme>
- GitLab. (n.d.). *The One DevOps Platform*. Retrieved May 18, 2022, from GitLab: <https://about.gitlab.com/>

- Hayes, A. (2021, March 29). *Quick Response (QR) Code Definition*. Retrieved May 2, 2022, from Investopedia: <https://www.investopedia.com/terms/q/quick-response-qr-code.asp>
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004, March). Design Science in Information Systems Research. *MIS Quarterly*, XXVIII(1), 75-105. doi:10.2307/25148625
- Howson, P. (2019, September). Tackling climate change with blockchain. *Nature Climate Change*, IX, 644-645. doi:10.1038/s41558-019-0567-9
- IBM. (n.d.). *What is blockchain security?* Retrieved May 16, 2022, from IBM: <https://www.ibm.com/topics/blockchain-security>
- IEEE. (1999). *Code of Ethics*. Retrieved May 19, 2022, from IEEE Computer Society: <https://www.computer.org/education/code-of-ethics>
- Jest. (n.d.). *Jest*. Retrieved May 4, 2022, from <https://jestjs.io/>
- Kjelsberg, R. (2019). *Teknologi og vitenskap* (2nd ed.). Universitetsforlaget AS. Retrieved May 18, 2022
- Lie, K. S., & Øverby, H. (2022, February 3). *blokkjede*. Retrieved April 29, 2022, from Store Norske Leksikon: <https://snl.no/blokkjede>
- Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). *A Review on Consensus Algorithm of Blockchain*. IEEE. doi:10.1109/SMC.2017.8123011
- Morrison, D. (n.d.). Common software vulnerabilities (1/2). IDATT2503 Sikkerhet i programvare og kryptografi. Trondheim, Trondelag, Norway. Retrieved May 12, 2022, from https://ntnu.blackboard.com/ultra/courses/_27108_1/cl/outline
- Moubarak, J., Chamoun, M., & Filiol, E. (2020, July 1). On distributed ledgers security and illegal uses. *Future Generation Computer Systems*, 183-195. doi:<https://doi.org/10.1016/j.future.2020.06.044>
- MUI. (n.d.). *Material UI*. Retrieved May 3, 2022, from <https://mui.com/>
- Nätt, T. H. (2022, February 21). *trusselaktør*. Retrieved from Store Norske Leksikon: <https://snl.no/trusselakt%C3%B8r>
- Next.js. (n.d.). *Next.js by Vercel - The React Framework*. Retrieved May 3, 2022, from Next.js: <https://nextjs.org/>
- Oblador. (n.d.). *Keychain/Keystore Access for React Native*. Retrieved May 3, 2022, from GitHub: <https://github.com/oblador/react-native-keychain>
- Parachains.info. (n.d.). *Auctions and crowdloans on Polkadot & Kusama networks*. Retrieved May 4, 2022, from Parachains.info: <https://parachains.info/auctions>
- Polkadot. (n.d.). *Advanced, Next-Generation Blockchain | Polkadot*. Retrieved May 2, 2022, from Polkadot: <https://polkadot.network/parachains/>
- PostgreSQL. (n.d.-a). *About*. Retrieved May 18, 2022, from PostgreSQL: <https://www.postgresql.org/about/>

- PostgreSQL. (n.d.-b). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Retrieved May 18, 2022, from PostgreSQL: <https://www.postgresql.org/>
- Prisma. (n.d.). *What is Prisma?* Retrieved May 4, 2022, from Prisma: <https://www.prisma.io/docs/concepts/overview/what-is-prisma>
- React Native Paper. (n.d.). *React Native Paper*. Retrieved May 5, 2022, from <https://reactnativepaper.com/>
- React Native. (n.d.). *React Native - Learn once, write anywhere*. Retrieved May 3, 2022, from React Native: <https://reactnative.dev/>
- Redis. (n.d.). *Redis*. Retrieved May 3, 2022, from <https://redis.io/>
- Singh, S. (n.d.). *What is HTTP Long Polling?* Retrieved May 4, 2022, from Educative: <https://www.educative.io/edpresso/what-is-http-long-polling>
- Socket.IO. (n.d.). *Introduction*. Retrieved May 4, 2022, from Socket.IO: <https://socket.io/docs/v4/>
- Solana. (n.d.-b). *Solana*. Retrieved May 3, 2022, from Solana: <https://solana.com/>
- Solana. (n.d-a). *Accounts*. Retrieved May 18, 2022, from Solana Documentation: <https://docs.solana.com/developing/programming-model/accounts>
- Sporny, M., Longley, D., & Chadwick, D. (2022a). *Presentations*. Retrieved May 3, 2022, from Verifiable Credentials Data Model v1.1: <https://www.w3.org/TR/vc-data-model/#presentations>
- Sporny, M., Longley, D., & Chadwick, D. (2022b). *Proofs (Signatures)*. Retrieved May 11, 2022, from Verifiable Credentials Data Model v1.1: <https://www.w3.org/TR/vc-data-model/#proofs-signatures>
- Sporny, M., Longley, D., & Chadwick, D. (2022c). *Verifiable Credentials Data Model v1.1*. World Wide Web Consortium. Retrieved May 2, 2022, from <https://www.w3.org/TR/vc-data-model/>
- Sporny, M., Longley, D., & Chadwick, D. (2022d). *Zero-Knowledge Proofs*. Retrieved May 3, 2022, from Verifiable Credentials Data Model v1.1: <https://www.w3.org/TR/vc-data-model/#zero-knowledge-proofs>
- Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., & Allen, C. (2021). *Decentralized Identifiers (DIDs) v1.0*. World Wide Web Consortium. Retrieved May 2, 2022, from <https://www.w3.org/TR/did-core/>
- Stinson, D. R., & Paterson, M. (2018). *Cryptography: Theory and Practice* (4th ed.). Taylor & Francis Group. Retrieved May 18, 2022
- Substrate. (n.d.). *Substrate How-to Guides*. Retrieved May 4, 2022, from Substrate: <https://docs.substrate.io/how-to-guides/v3/>
- Sunyaev, A. (2020). Distributed Ledger Technology. In A. Sunyaev, *Internet Computing* (pp. 265-299). Cham: Springer. doi:10.1007/978-3-030-34957-8_9
- Supabase. (n.d.-a). *APIs*. Retrieved May 4, 2022, from Supabase: <https://supabase.com/docs/guides/api>

- Supabase. (n.d.-b). *Database*. Retrieved May 4, 2022, from Supabase: <https://supabase.com/docs/guides/database>
- Supabase. (n.d.-c). *Introduction*. Retrieved May 4, 2022, from Supabase: <https://supabase.com/docs>
- Thales. (n.d.). *Digital ID Wallet - Your credentials at hand (Mobile ID Services)*. Retrieved May 18, 2022, from Thales: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/identity/digital-identity-services/digital-id-wallet>
- TypeScript. (n.d.). *TypeScript for JavaScript Programmers*. Retrieved May 3, 2022, from TypeScript: <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>
- Understanding Science. (n.d.). *The real process of science*. Retrieved May 16, 2022, from Understanding Science: https://undsci.berkeley.edu/article/0_0_0/howscienceworks_02
- United Nations. (n.d.-a). *Climate Change*. Retrieved May 15, 2022, from United Nations Sustainable Development: <https://www.un.org/sustainabledevelopment/climate-change/>
- United Nations. (n.d.-b). *Global Partnerships*. Retrieved May 14, 2022, from United Nations Sustainable Development: <https://www.un.org/sustainabledevelopment/globalpartnerships/>
- United Nations. (n.d.-c). *Infrastructure and Industrialization*. Retrieved May 15, 2022, from United Nations Sustainable Development: <https://www.un.org/sustainabledevelopment/infrastructure-industrialization/>
- United Nations. (n.d.-d). *Peace, justice and strong institutions*. Retrieved May 14, 2022, from United Nations Sustainable Development: <https://www.un.org/sustainabledevelopment/peace-justice/>
- Veramo. (n.d.-a). *Introduction*. Retrieved May 4, 2022, from Veramo: <https://veramo.io/docs/basics/introduction/>
- Veramo. (n.d.-b). *Message Handlers*. Retrieved May 11, 2022, from Veramo: https://veramo.io/docs/veramo_agent/message_handlers/
- Veramo. (n.d.-c). *Veramo Agent*. Retrieved May 11, 2022, from Veramo: https://veramo.io/docs/veramo_agent/introduction/
- Wood, G. (n.d.). *Substrate and Polkadot*. Retrieved May 4, 2022, from Substrate: <https://substrate.io/vision/substrate-and-polkadot/>
- World Wide Web Consortium. (2022). *Verifiable Credentials Data Model v1.1 Publication History*. Retrieved May 6, 2022, from W3C: <https://www.w3.org/standards/history/vc-data-model>
- Yakovenko, A. (n.d.). *Solana: A new architecture for a high performance blockchain v0.8.13*. Retrieved May 4, 2022, from <https://solana.com/solana-whitepaper.pdf>

Zhang, R., & Kin, W. (2020). Evaluation of Energy Consumption in Block-Chains with Proof of Work and Proof of Stake. *Journal of Physics: Conference Series*, 1584. doi:10.1088/1742-6596/1584/1/012023

Zhao, Y. (2018). *Research on the Consensus Mechanisms of Blockchain Technology*. doi:10.25236/icefbd.18.063

Appendix

Appendix A: Work distribution charts

Appendix B: Issue board

Appendix C: Task description

Appendix D: Pre-project plan

Appendix E: Project handbook

Appendix F: Vision document

Appendix G: Requirement document

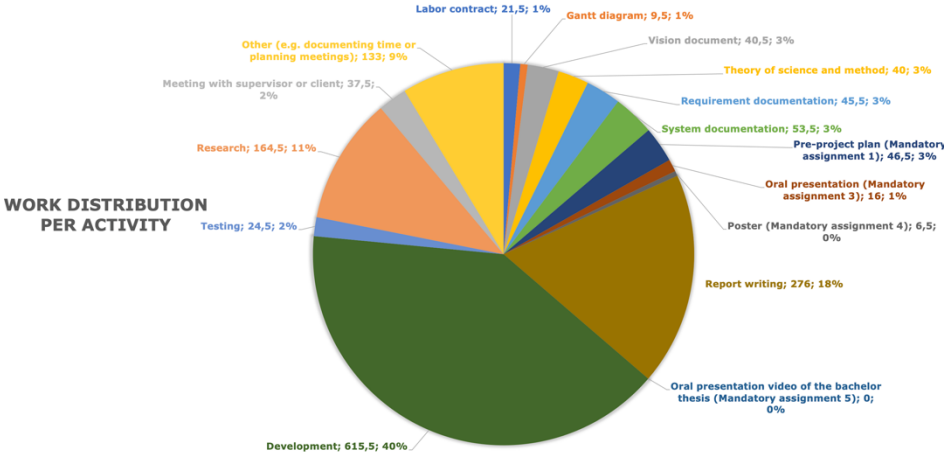
Appendix H: System documentation

Appendix I: Manual testing

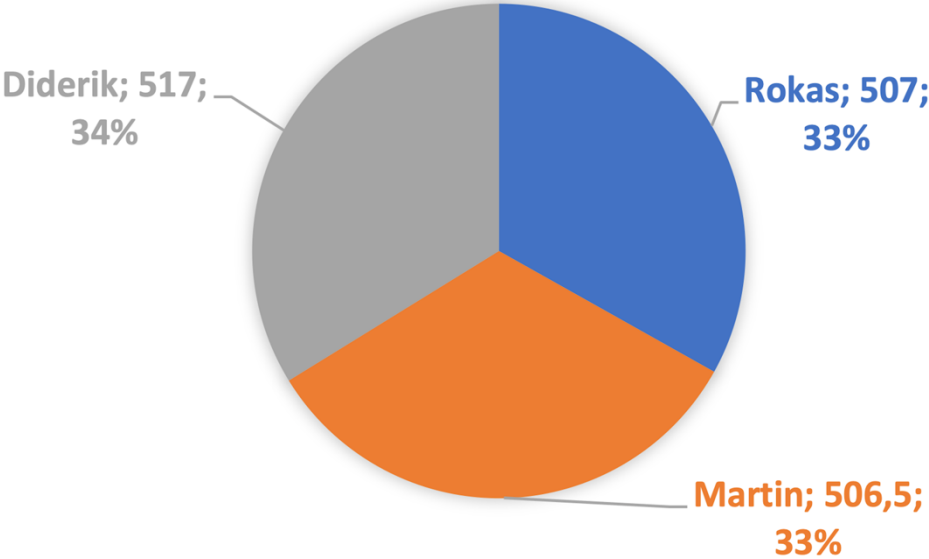
Appendix J: Kanban board description

Appendix K: Source code

Appendix A: Work distribution charts



WORK DISTRIBUTION PER TEAM MEMBER



Appendix B: Issue board

The screenshot displays a Jira issue board with three columns: Open, Done, and Closed. The board is organized into three vertical lanes, each with a filter bar at the top. The 'Open' column (7 issues) contains tasks such as 'Fix: Await response for bulk revoking ids', 'Edit id connect to iris', and 'Add selective disclosure impl'. The 'Done' column (70 issues) lists completed tasks like 'Document code and general clean up', 'Update README.md', and 'Add environment variables'. The 'Closed' column (8 issues) includes items like 'Logic for returning necessary data for creating did document' and 'Add zero knowledge proof'. Each issue card shows a title, a unique ID, and a small icon representing the issue type.

Column	Issue Title	Issue ID	
Open (7)	Fix: Await response for bulk revoking ids (see description)	ntnu-id thesis 22_077 iris#20	
	Edit id connect to iris	ntnu-id thesis 22_077 athena#15	
	Add selective disclosure impl	ntnu-id thesis 22_077 aphrodite#33	
	Fix NextJS MUI bug (Info in description)	ntnu-id thesis 22_077 athena#10	
	Observer pattern for global error handling with snackbar	ntnu-id thesis 22_077 aphrodite#27	
	Add ranges in share data page	ntnu-id thesis 22_077 aphrodite#26	
	Path not found error handling	ntnu-id thesis 22_077 iris#5	
Done (70)	Document code and general clean up	ntnu-id thesis 22_077 iris#22	
	Document code and general clean up	ntnu-id thesis 22_077 athena#18	
	Document code and general clean up	ntnu-id thesis 22_077 aphrodite#38	
	Update README.md	ntnu-id thesis 22_077 aphrodite#36	
	Update README.md	ntnu-id thesis 22_077 athena#17	
	Add environment variables	ntnu-id thesis 22_077 aphrodite#37	
	Update README.md	ntnu-id thesis 22_077 kratos#10	
	Logic for verifying a did (physical verification)	ntnu-id thesis 22_077 aphrodite#25	
	Send id details to requester through Log in UI	ntnu-id thesis 22_077 athena#4	
	Check backend for existing agreement during verification	ntnu-id thesis 22_077 aphrodite#35	
	Update README.md		
	Closed (8)	Logic for returning necessary data for creating did document from did	ntnu-id thesis 22_077 iris#14
		Endpoint for accepting did (and relevant data for constructing did document)	ntnu-id thesis 22_077 iris#11
Endpoint for returning necessary data for creating did document from did		ntnu-id thesis 22_077 iris#13	
Send did document to backend when creating user		ntnu-id thesis 22_077 aphrodite#24	
Edit ID endpoint		ntnu-id thesis 22_077 iris#9	
Edit ID implementation		ntnu-id thesis 22_077 kratos#5	
Add zero knowledge proof		ntnu-id thesis 22_077 aphrodite#16	
Add Zero knowledge proof		ntnu-id thesis 22_077 iris#7	

Appendix C: Task description

Arbeidstittel:

Digital ID Wallet App (+ backend)

Hensikten med oppgaven:

These days, every private and public service is either digitalized, or in digitalization phase. That is good. However, there are underlying fundamental problems to be solved. Variety of services depend on array of physical and/or digital identities of end-users, and task of end-users to prove their identity with secure trusted credentials is getting cumbersome. With the aim of simplifying user journey of proofing their identity, the European Union has been working on the concept of Digital ID wallet. It is conceptually like a physical wallet which holds user's multiple identities at one place and let the users (the wallet holders) control over their data, with the freedom to decide exactly what information to share, with whom, and when. This project is about building minimum viable "Digital Identity Wallet" App - Android or iOS or Web.

Kort beskrivelse av oppgaveforslag:

There will be two main parts:

- A wallet like Mobile (and/or web App). This app can either be used to initiate verification process (for verifier role) or let end-users to verify their identities from various identity sources depending on the use-case.
- An orchestration platform at the backend. This will coordinate the integration with identity sources (identity providers like BankID++) and offer APIs for the front-end mobile or web app.

Some potential scenarios that can be implemented for the MVP are the following.

- Mobile wallet app to be used for age verification purposes at bar entrance or for shopping alcohol at shops.
- Mobile wallet app to be used for driver's license verification by police or by vehicle leasing companies
- Web/Mobile app to be used for creating bank accounts online or shopping prescribed drugs at pharmacy that has requirement for ID with photo
- Mobile app to be used for passport, visa or residence permit verification purposes at airports or police offices

The project can also potentially be carried out by two smaller groups either independently or via collaboration. One can focus on the front-end side and while the other on the backend side. To have smoother progress, either group can simplify the other group's part by mocking their corresponding interfaces. For example, front-end group can use simplified backend by mocking third-party integration part or the backend group can use simplified front-end in the form of raw API calls. Towards the end of the project, the two groups can potentially (if feasible) integrate towards each other's work.

NB: Industrial collaboration might be possible.

References:

- https://ec.europa.eu/commission/presscorner/detail/en/ip_21_2663
- https://www.finextra.com/blogposting/21140/cryptocurrency-wallet-development--a-quick-guide-for-crypto-wallet-creation?utm_medium=weeklycommunity&utm_source=2021-11-1

Oppgaven passer for (kryss av de(t) som passer og skriv evt. en kommentar til oss):

- Bacheloroppgave
- Masteroppgave, Digital samhandling

Hvilket studieprogram og emne passer oppgaven til? (spesifiseres ved bacheloroppgaver)

IDATT2900 - Bacheloroppgave Dataingeniør

Oppgaven passer best for, antall studenter:

- 2
- 3

Opplysninger om oppgavestiller

Er du fra en bedrift/virksomhet eller er du student med en egendefinert/selvlaget

oppgave?	- Bedrift/virksomhet
Navn på bedrift/organisasjon/student:	NTNU
Adresse	IDI/AIT
Postnummer	7491
Poststed	Trondheim
Navn på kontaktperson/veileder:	Surya Kathayat
Telefon:	46392873
Epost:	surya.b.kathayat@ntnu.no

Appendix D: Pre-project plan

IDATT2900-077

**Digital ID Wallet App
Preliminary project plan**

Version <1.0>

IDATT2900-077

Revision history

Date	Version	Description	Author
11.01.2022	0.1	Initial revision, translated into English	Diderik, Martin, Rokas
19.01.2022	0.2	Wrote on chapters 1, 2, 3, 4, 6	Diderik, Martin, Rokas
20.01.2022	0.3	Wrote on chapters 1, 3, 4, 5, 6	Diderik, Martin, Rokas
27.01.2022	0.4	Changes after review	Diderik, Martin, Rokas
28.01.2022	1.0	Final changes	Diderik, Martin, Rokas

Table of contents

1. Goals and conditions	4
1.1 Orientation	4
1.2 Thesis question, project description and performance targets	4
1.3 Effect goals	5
1.4 Conditions	5
2. Organizing	5
3. Implementation	6
3.1. Main activities	6
3.2. Milestones	10
4. Supervision and quality assurance	10
4.1 Quality assurance	10
4.2 Reporting	11
5. Risk assessment	11
6. Appendix	12
6.1 Schedule	12
6.2 Address list	12
6.3 Contractual documents	13
6.3.1 Labor agreement for bachelor-group	13

1. Goals and conditions

1.1 Orientation

This was one of the predefined tasks available to select. The team wanted to create an application, while being able to focus on security and privacy. Based on the interests of the team the task was convenient. The goal was to create something new which potentially could contribute to identity digitalization and web decentralization. This meant that the team would also learn something new in addition to using knowledge the team has gathered throughout the studies.

1.2 Thesis question, project description and performance targets

The main goal of the project is to create a Digital ID Wallet application. The wallet can store various IDs and be used to verify other IDs. The application will be an Android-, iOS- and/or web app, with a partially decentralized backend storage system based on blockchain technology. An issuer/ID provider will be able to register to a centralized part of the system, and then send identity documents to users using peer-to-peer connections. The transactions will be safely stored on a blockchain, which can then be used by different verifiers to check if the documents of different users are valid.

There are many brilliant technologies in this field of digital identity and decentralization. Throughout the project, the team will describe the different choices and approaches of technology use. What technologies can be used? How will user identities be represented and stored? Will there be a need to use a centralized database and/or a decentralized blockchain? How can digital identities be used for verification using mobile devices? Can a user share certain data of the ID without exploiting unnecessary information?

Various performance targets are set to help achieve the goal:

- Application follows OWASP top 10.
- Frontend can run on iOS and Android.
- Backend test coverage of at least 50%.
- Every API endpoint at backend is documented (Swagger).
- The application can be used to identify oneself.
- The application grants users control over their data.
- The application can be used for age verification.
- The application will be free.
- The application will store all identity documents encrypted.
- The application will offer protection of identity documents through pin code or biometrics, that when successful will trigger the decryption of the identity documents.
- The application will be RESTful.
- The application can create a QR-code for an ID.
- The application can verify another user's QR-code.
- The application can display all registered ID providers.
- The application can add, remove, and refresh an ID from a registered ID provider.

IDATT2900-077

- ID providers can register themselves to the application with DID and public key.
- The application will be based on decentralization, where all ID transactions between ID providers and users will permanently be saved as blocks in a blockchain.
- Verifiers will be able to check validity of IDs by accessing the public blockchain and checking if the provider and owner combinations exist. If it exists, then checking if the ID has been revoked is also possible.
- The application will allow web services the opportunity to ask the app to verify a user instead of having to create and maintain a user system.
- The theory section of the main report will contain descriptions of all technology used that are more advanced than a sophomore Computer science student would know. The section will refer to general technology and not implementations e.g., writing about a programming language instead of Java.

1.3 Effect goals

Since society today is moving towards a digitalized world, identity digitalization has also become an important topic. Therefore, the goal of this project is to simplify user identification procedures by creating a minimum viable digital ID wallet application (product). The product would contribute towards:

- Development of the European Digital Identity Wallet concept proposed by the European Union.
- Replacement of society's dependency of physical identification documents, contributing towards a more efficient and secure society, where the users have full authority of their data.
- Increased security. Physical identification documents are prone to falsification, theft, and loss. A digital ID wallet can resolve these issues.
- Sustainability. Digital IDs would mean less use of plastic and paper for creation of ID cards.

1.4 Conditions

There is currently no need for money or equipment. If the application is to be deployed on a blockchain in the future, money to invest in a valid cryptocurrency for paying fees may be needed. However, the application will be using test blockchains during development to avoid transaction fees and other expenses. The duration of the bachelor thesis is around 18 weeks (about 4 months). This means that the conditions of the project will be set by the time available.

2. Organizing

NTNU/IDI is the sole stakeholder in the project.

3. Implementation

3.1. Main activities

This subchapter describes the main activities that will be done by the team during this project. For each activity, the subpoints describe what is done, why it is done, how it is done, when it should be done, and if there are any prerequisites. All team members will contribute to every activity.

- Labor contract
 - This consists of writing a document containing goals to be pursued and rules to be followed during the project, resulting in a labor agreement.
 - The contract is the first thing that is written in the project.
 - The labor contract's purpose is to serve as a guideline for the agreed upon goals and rules.
 - To complete the activity the team will sit together and draft the document.
 - There are no prerequisites to completing the document other than having reached consensus regarding the interpretation of the task.
 - The result of the labor contract is a document containing agreed upon goals and rules for the project.
- Gantt diagram
 - The activity consists of scheduling all activities.
 - This is created for the team to get an overview of the project schedule and set deadlines for the various activities.
 - To complete the activity, the team sits together and plans the schedule.
 - This is done early in the project.
 - There are no prerequisites for this activity.
 - The activity results in a Gantt diagram displaying the start and duration of each activity.
- Vision document
 - This consists of writing a document explaining the vision for the product, by describing what properties the product is envisioned to have.
 - This is done to serve as a guideline for the vision for the product, its uses, and features.
 - Vision document is written using a template provided from the university.
 - The document is written early in the project.
 - There are no prerequisites for writing the document.
 - The activity results in a vision document explaining the vision for the project.
- Pre-project plan
 - Writing this document consists of describing the plan for the project. This means description of topics such as conditions, implementation, milestones, supervision, and risk assessment.
 - This document is written to serve as a guideline for the project's process and goals.
 - Pre-project plan is carefully written together in the team so that all team members are included in project planning, and that all come to a mutual understanding and agreement of the project's process.
 - The plan is written early in the project, during weeks 2-4.

IDATT2900-077

- Before writing the plan, the labor contract must be signed by all parties included. The Gantt diagram must also be completed.
- The activity will result in a plan that can be followed by all parties and referred to during the project.
- Requirement documentation
 - Writing the requirement document consists of writing a document containing requirements for the final product.
 - The activity is done to be able to easily check that all requirements have been accomplished later, as well as quickly providing information about the product capabilities to someone unfamiliar with it.
 - To complete the activity, firstly, the team drafts the document together. Thereafter, if needed, the document is updated throughout the project based on feedback from the supervisor and the client.
 - The first draft of the document should be done during weeks 3 and 4.
 - Before starting the activity, a mutual understanding about the goals of the project needs to be reached with the supervisor and client through completing the vision document.
 - The resulting documentation created from this activity will help guide the development process by listing the requirements for the application, while also serving as a summary of the applications capabilities.
- System documentation
 - This activity consists of describing the system (product) that was created throughout the project.
 - The system documentation is necessary to understand the details of the system. Such documentation is useful for everyone who wishes to utilize or understand the system.
 - This is achieved through describing the structure, security, installation, and other necessary topics. Different models and diagrams are also utilized to better describe the system that has been created.
 - This document should be written towards the end of the project.
 - The prerequisite is that the product (the system) has taken shape, which means that some parts of the system are finished so that they can be described in the system documentation.
 - The result of this activity is a document describing the completed product.
- Oral presentation during project
 - This activity consists of presenting the project and its progress to another team, their supervisor, and this team's supervisor.
 - This activity gives the team an opportunity to display the progress made. Moreover, the other team members can learn something from this project, or even give some feedback. Likewise, this team can learn something from the other team's presentation. This is also an opportunity to discuss progress with both supervisors.
 - This activity is done together with another team, where posters (described in the next point in the activity list) are used to display the project's progress.
 - The presentation is being held in week 13.

IDATT2900-077

- The prerequisites are that the team worked with the project before week 13, such that the team has some progress to present and share with the other team. A presentation with a poster must also be created beforehand.
- The activity will result in knowledge gained about how other teams are working on the thesis and being able to display the progressions made.
- Poster
 - The activity consists of creating a poster containing information and images of the product's current state.
 - The poster activity is an opportunity for the team to present the current implementations of product for others and the supervisor. The team will have to explain their concepts and theory to others and argue for their decisions.
 - The team will design a poster containing product images and information.
 - Making the poster will happen from week 12 to week 13.
 - Before the poster can be created, the development of the product must be underway.
 - The activity will result in a poster about the project's state.
- Oral presentation video of the bachelor thesis
 - This consists of creating a video of the team's oral presentation.
 - The oral presentation should contain simplifications of each topic from the report, resulting in a summary of the project. This is to show others what the team has created throughout the project, and how the team was working to achieve it (process). The oral presentation also creates an opportunity for all team members to show that they understand their work by being able to freely talk about the project.
 - The team will create slides consisting of relevant topics from the project. The team will then present the slides and record it.
 - The video will be created at the end of the project, during weeks 20-21.
 - The product and main report need to be finished before creating the video.
 - The finished video will contain a presentation of the project.
- Theory of science and method
 - The activity consists of both live and pre-recorded lectures, films and reading material.
 - The activity is necessary for the team members to gather necessary knowledge for authoring a scientific main report.
 - By attending the lectures and reading the assigned material the team will complete this activity.
 - The activity will be completed during the first two weeks of the project.
 - There are no prerequisites for this activity.
 - The results of the activity will be reflected in the scientific aspects of the report.
- Report writing
 - The activity consists of creating a report fulfilling all the requirements given in the main report template.
 - This must be done because it will provide the necessary knowledge for others to replicate the process along with learning from the results of the project.

IDATT2900-077

- To complete the activity the report will be continuously updated while developing the application. The theory section, however, may be started on before development. During the writing of the report feedback will be gathered from the supervisor to continuously improve the report.
- Report writing will start in week 4 and last to 20.
- To begin with authoring the report the labor contract, requirement document, Gantt diagram, theory of science and method, and the pre-project plan all must be completed.
- This activity will result in a 30-50 page long main report containing theory, method, result, observations, conclusions and more.
- Development
 - This consists of developing the planned system.
 - The purpose of this activity is to develop a product.
 - Development will be done through the Kanban method.
 - Development will start in week 4 and end week 18, since the last weeks will be spent solely on the report.
 - Before development can begin, the requirement document needs to be completed along with potential research and planning.
 - The development process will result in the completed product as well as knowledge to be utilized in the main report.
- User testing
 - This consists of collecting feedback from potential users about the product.
 - The activity should be done at least twice, first with the wireframe and afterwards with a usable version of the product.
 - The user tests start at the earliest when the first draft of the wireframe is completed. As a usable version of a product is created more rounds of user tests can be completed.
 - A round of user tests should always occur early enough to make potential changes before the thesis deadline.
 - An interview plan/schema needs to be created before testing can start.
 - After a round of interviews have been completed and the schemas have been filled, they will be stored in the “User testing” folder in Microsoft Teams.
 - The feedback from the interviews will hopefully contribute to improving the product.
- Research
 - The research activity is the process of gathering necessary knowledge to complete the project.
 - Research is needed for developing the product and being able to make the best decisions possible regarding the product by understanding it.
 - To conduct the best research possible peer-reviewed articles will be used as sources of knowledge, alongside interviews with professionals.
 - The research process will span the entirety of the project except for the last weeks, when the focus will be on finalizing the report.

- To be able to conduct necessary research the team needs an understanding of what knowledge is necessary, therefore an agreement on technology to be used and an overall architecture of the product must be reached with the client and supervisor.
- The resulting knowledge gathered from the research will be distributed in the theory section of the main report as well as reflected by the implementation of the final product.

3.2. Milestones

- 28.01.2022: Preliminary project plan delivery deadline.
- 28.03.2022: Delivery deadline for poster.
- 28.03.2022 - 03.04.2022: English presentation of project.
- 20.05.2022: Delivery deadline for thesis and process documentation.
- 27.05.2022: Delivery deadline for digital presentation.

4. Supervision and quality assurance

4.1 Quality assurance

Document (including videos, posters, and more) quality assurance policy:

- All members must ensure the quality of all documents.
- All documents should be continuously reviewed.
- All documents must follow the templates provided.
- Ask supervisor for advice if something is uncertain.

Development quality assurance policy:

- Code should document itself if possible.
 - All variables should have descriptive and self-explanatory names.
 - Functional cohesion should be strived for, meaning each function only completes one action.
- If possible, immutable objects should be used to hinder unexpected side effects.
- All complex code needs to be documented.
- Team members must give constructive feedback to each other's work.
- At least one team member needs to approve a pull/merge request.
- All backend code containing logic (e.g., no data classes) must have automated unit tests at 50% code coverage minimum.
- All API endpoints must be tested using integration testing.
- Continuous integration
 - All tests should be run during a pull/merge request.
 - A linter and formatter should run on all code possible.

4.2 Reporting

The team reports to Surya Kathayat, who is both the client and the supervisor for this project, during meetings every Friday after 13:00.

Other necessary documents and products will be submitted to NTNU (see “*Milestones*” subchapter 3.2).

5. Risk assessment

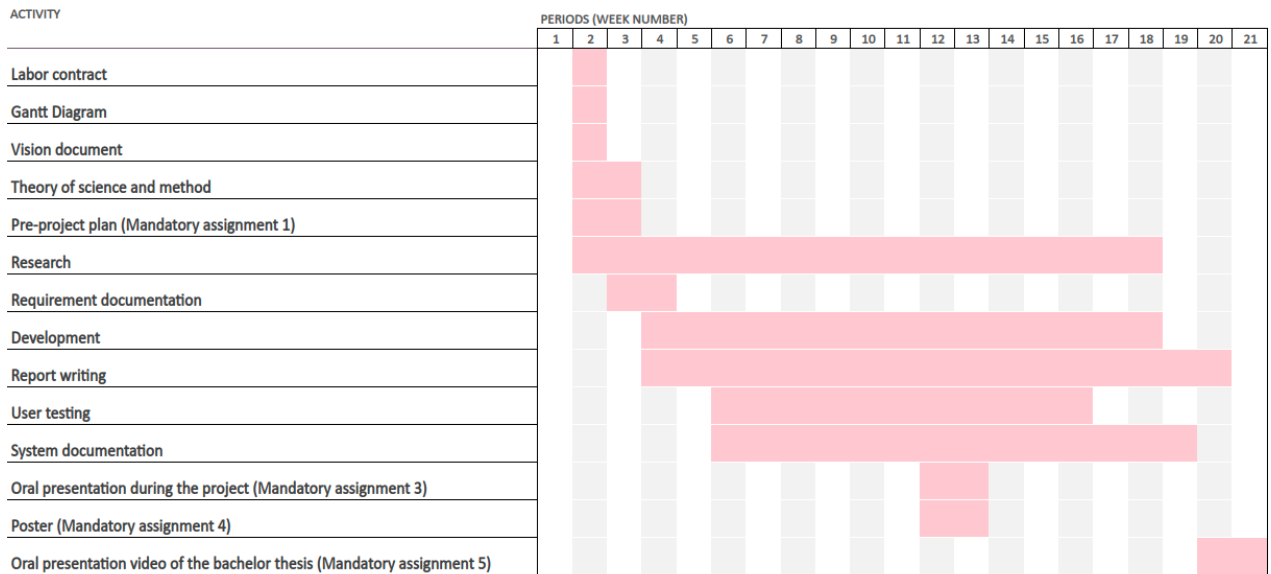
The risk assessment table below shows different risk actions that can occur during the project, and their corresponding probabilities and impacts (measured on a scale from 1-10, 10 being the highest probability or the biggest impact). The significance column corresponds to the product of the action’s probability and impact.

Action	Probability	Impact	Significance	Measures
Obligatory military service	2	9	18	If possible, the person in question must work digitally. If this is not a solution, then the person and the team must discuss this with the project supervisor or study advisor.
Illness	8	4	32	An ill person must either rest or work from home through digital tools until they are healthy, depending on the level of illness.
Disagreement	8	2	16	The disagreement will be handled like defined in the Labour Agreement under Interaction, subpoint D “ <i>Disagreements, breach of contract.</i> ”
Conflict between members	1	6	6	The conflict will be handled like defined in the Labour Agreement under Interaction, subpoint D “ <i>Disagreements, breach of contract.</i> ”
Loss of documents, code, or any other necessary data	1	10	10	To prevent this, everything must be saved online. Additionally, the team members are responsible for making weekly back-ups on hardware. If this action does happen, the team must discuss further actions with the supervisor.
Social restrictions	6	1	6	Work from home.

6. Appendix

6.1 Schedule

The following Gantt diagram visualizes the timeline of all activities that are necessary for the completion of this project. Each period represents a week-number, and the week columns marked with red color represent the planned time needed to complete the corresponding activity.



6.2 Address list

Name	Firm	Telephone number	Email address	Address	Postal code	City
Bliudzius, Rokas	None	+47 967 24 930	rokasb@stud.ntnu.no	Jakobslivegen 49B	7059	Trondheim
Kathayat, Surya	NTNU	+47 463 92 873	surya.b.kathayat@ntnu.no	IDI/AIT	7491	Trondheim
Hagen, Martin Slind	None	+47 462 12 453	martsha@stud.ntnu.no	Odd Brochmanns veg 53	7051	Trondheim
Kramer, Diderik	None	+47 948 93 643	diderikk@stud.ntnu.no	Vegamot 1	7049	Trondheim

6.3 Contractual documents

6.3.1 Labor agreement for bachelor-group

Labor agreement for IDATT2900-077

Members: Bliudzius Rokas, Hagen Martin Slind, Kramer Diderik

Introduction

This labor agreement uses a set of typical goals, task distributions, procedures and guidelines for student work interactions. The labor agreement is filled with perceptions of what is meant with these and how to achieve them.

Items are added or removed depending on evaluation of adaptation to the task.

Goals

Effect goals

Since society today is moving towards a digitalized world, identity digitalization has also become an important topic. Therefore, the goal of this project is to simplify user identification procedures by creating a minimum viable digital ID wallet application (product). The product would contribute towards:

- Development of the European Digital Identity Wallet concept proposed by the European Union.
- Replacement of society's dependency of physical identification documents, contributing towards a more efficient and secure society, where the users have full authority of their data.
- Increased security. Physical identification documents are prone to falsification, theft, and loss. A digital ID wallet can resolve these issues.
- Sustainability. Digital IDs would mean less use of plastic and paper for creation of ID cards.

Performance targets

- Application follows OWASP top 10.
- Frontend can run on iOS and Android.
- Backend test coverage of at least 50%.
- Every API endpoint at backend is documented (Swagger).
- The application can be used to identify oneself.
- The application grants users control over their data.
- The application can be used for age verification.
- The application will be free.
- The application will store all identity documents encrypted.
- The application will offer protection of identity documents through pin code or biometrics, that when successful will trigger the decryption of the identity documents.
- The application will be RESTful.

IDATT2900-077

- The application can create a QR-code for an ID.
- The application can verify another user's QR-code.
- The application can display all registered ID providers.
- The application can add, remove, and refresh an ID from a registered ID provider.
- ID providers can register themselves to the application with DID and public key.
- The application will be based on decentralization, where all ID transactions between ID providers and users will permanently be saved as blocks in a blockchain.
- Verifiers will be able to check validity of IDs by accessing the public blockchain and checking if the provider and owner combinations exist. If it exists, then checking if the ID has been revoked is also possible.
- The application will allow web services the opportunity to ask the app to verify a user instead of having to create and maintain a user system.
- The theory section of the main report will contain descriptions of all technology used that are more advanced than a sophomore Computer science student would know. The section will refer to general technology and not implementations e.g., writing about a programming language instead of Java.

Roles and task management

Not following the responsibilities for one's role results in one violation of the contract.

- A. *Team leader (Martin)*
Responsible for leading the team. This includes looking after the team and resolving disagreements if there are any.
- B. *Document manager (Diderik)*
Responsible for handling all documents, document version control, needs to save a copy of all documents every Friday.
- C. *Quality assurance manager (Rokas)*
Responsible for reading through all documents and making sure that the quality is as expected, and all conditions are met. Responsible for delivering all documents on time.
- D. *Meeting conductor (Martin)*
Responsible for the meeting agenda, conducting the meetings and inviting all parties to meetings via meeting requests.
- E. *Meeting summary writer (Rokas)*
Responsible for taking notes during meetings and recording them.

Procedures

- A. *Meeting invitation*
Schedule meetings through Microsoft Teams. Store each meeting invitation, use the form from the project handbook.
- B. *Meeting summary*
Use the form from the project handbook.
- C. *Notification when absence or incidents*
Applies only to mandatory attendance e.g., meetings or working sessions. Write a message in the Microsoft Teams group chat.
- D. *Document management*
Documents are saved and shared between everyone in Microsoft Teams. Additionally, the Document Manager is responsible for back-up saving the documents on personal hardware as an extra safety measure.
- E. *Submission of group work*
Everyone in the group should proofread the final version of the document(s) in question at least twice. Submission done by quality assurance manager.
- F. *Microsoft Teams*
Strictly for professional use. Used to hold meetings as well as to store documents.
- G. *Time sheets*
At the end of the day each team member records what they have done at each point in time in the "Time sheet team_member_name".docx file.
- H. *Kanban board*
A number cap should be added to each column to hinder bottlenecks. Tasks should be pulled from left to right when the number cap is not reached in the destination column. Tasks are ranked by priority based on their vertical ordering in each column (highest ranking as most important).

Interaction

- A. *Attendance and preparation*
Expected workhours are 8 per weekday with a core time from 09:00-15:00 with room for change and exceptions, such as lectures/work in other courses, training or matches. In the case of training or matches it is expected that the work hours missed are later replaced by working overtime. Always aim to be 5 minutes early to meetings. 30-minute lunches are allowed. Breaks are allowed if the necessary work gets done in time. All relevant procedures

IDATT2900-077

for an event are expected to be completed before the event starts. When all work is done the expected workhours are no longer mandatory and the team members can therefore take time off.

B. *Presence and commitment*

Using a PC for entertainment is only allowed during breaks. When other team members depend on communication to get work done, giving them the required attention is mandatory. Meaning taking a break or listening to music instead of them is not permitted. Members are expected to be committed to the project during workhours. When the expected work is not completed due to a lack of commitment, then it is expected that the team member uses their spare time to catch up.

C. *How to support each other*

Always show up to work with a positive attitude. Make sure feedback is constructive and not negative. Make sure to complement team members on task related achievements. Always be open and available to help others.

D. *Disagreements, breach of contract*

- a. If a disagreement that is not listed in the contract occurs, then each team member should make a case for their opinion and listen to all others' opinion, thereafter, voting shall precede. Everyone must vote for one option. If no option wins, then the team leader makes the final decision by carefully considering all options. In cases where the team leader is unable to decide, or if the team leader is one of the two parties in the disagreement, then the supervisor is contacted for a final decision.
- b. A breach of contract is registered in the violations document. Every 3 violations result in buying the rest of the team at least 2 pizza slices each. After 10 violations talk to the team member about contract breaches. After 15 violations talk to the course coordinator about excluding the team-member from the project.

Signatures

Bliudzius, Rokas

Rokas Bliudzius

Hagen Martin, Slind

Martin Slind Hagen

Kramer, Diderik

Diderik Kramer

Appendix E: Project handbook

The project handbook is not attached to the main report but is delivered privately to the supervisors as "Project handbook.pdf" file.

Appendix F: Vision document

IDATT2900-077

**Digital ID Wallet App
Vision document**

Version <1.1>

Revision history

Dato	Version	Description	Author
11.01.2022	0.1	Initial changes	Diderik, Martin, Rokas
12.01.2022	0.2	Translated to English, started writing chapter 1, 2, 3, 5, 7	Diderik, Martin, Rokas
13.01.2022	0.3	Finished writing chapters 1, 2, 3, 5, 7, wrote chapters 4, 6.	Diderik, Martin, Rokas
18.01.2022	0.4	Added more information in chapters 1, 2, 3, 4, 5, 6	Diderik, Martin, Rokas
19.01.2022	0.5	Added some subpoints and figures in chapter 4	Diderik, Martin, Rokas
27.01.2022	1.0	Small changes after review. Final draft.	Rokas
20.05.2022	1.1	Needed to change picture DPI, added figure table, changed reference style	Rokas, Martin

Table of content

1.	Introduction	5
2.	Summary of problem and product	5
2.1	Summary of problem	5
2.2	Summary of product	5
3.	Overall description of stakeholders and users	6
3.1	Summary of stakeholders	6
3.2	Summary of users	6
3.3	User environment	6
3.4	Summary of user needs	7
3.5	Alternatives to our product	10
4.	Product synopsis	10
4.1	The product's role in the user environment	10
4.2	Prerequisites and dependencies	12
5.	Functional properties of the product	12
6.	Non-functional properties and other needs	13
7.	References	13

Figures

Figure 4.1: Examples of user environments where the product can be used by a common application user	11
Figure 4.3: Examples of user environments for a verifier.	12
Figure 4.4: Examples of user environments for an identifier.....	12

1. Introduction

This document explains the ideas behind a Digital ID Wallet application. The vision for this project is to create a minimal viable product for such an application, which should be available on Android, iOS and/or web.

2. Summary of problem and product

2.1 Summary of problem

Problem with	<i>having to carry different physical IDs for different purposes or relying too much on centralized digital ID providers and ID issuers</i>
concerns	<i>everyone with an ID</i>
resulting in	<i>forgetting the required ID, identity theft through forgery and stealing, unwanted ID-data being leaked, being tracked by ID providers</i>
a successful solution will	<i>store all different IDs well arranged in one place, be unforgeable, give the user full control of their data, and overview of the shared IDs.</i>

2.2 Summary of product

For	<i>users</i>
needing	<i>to keep track of their IDs, keep them theft proof, keep their usage private from ID providers and access their IDs without downtime</i>
Digital ID wallet	<i>is a mobile application</i>
able to	<i>safely store different identity documents in one place such that a user may easily find and access an ID at any time even if the provider is having downtime. Additionally, it should be able to hinder identity theft through protecting the user's identity documents with passwords or biometrics, while being cryptographically secure hindering data exposures that could lead to forgery. As well as empowering user privacy through giving the user full ownership of the registered IDs.</i>
contrary to	<i>having one physical item for every ID that can be stolen or falsified and easily distributed to unwanted parties, or being tracked by a digital ID provider</i>
our product offers	<i>on device safekeeping of digital IDs, the possibility to use the app for verification even during ID provider downtime, and an overview for the user to control ID access.</i>

3. Overall description of stakeholders and users

3.1 Summary of stakeholders

Name	Elaborate description	Role during development
<i>NTNU/IDI</i>	Supervisor and client	Continuously provides and updates conditions for the project such as time, money, and features. Gives feedback on current implementations.

3.2 Summary of users

Name	Elaborate description	Role during development	Represented by
<i>User/Owner</i>	Person with digital identification document(s) stored in the application.	Be a part of user testing.	Self
Issuer/ID provider	A provider of an ID a user can add to their wallet.	None (will be mocked during testing).	Self
Verifier/Service provider	An entity that provides a service given that it can verify the necessary data about the user. E.g., verify valid age.	None (will be mocked during testing).	Self

3.3 User environment

The project solution can solve problems in many different user environments for different types of users (see 3.2 Summary of users):

User/Owner:

- Circumstances where different identification documents are needed for different verification purposes. This may be relevant in scenarios such as:
 - Airport check-ins
 - Police station visits
 - Liquor and drug store checkouts
 - Driving (driver's license)

- Bar entrances
- Online identification while logging in or other online activities such as shopping.
- Identity theft when carrying physical ID.
- Forgetting physical ID which can cause inconveniences.
- Unwanted sharing of physical IDs or digital copies of them.
- Accessing/using identification document while ID provider services are down.
- Using digital identities while not being tracked by ID provider.

Issuer/ID provider:

- Providing ID for online verification to improve on the current username and password system.

Verifier/Service provider:

- Verify another person's ID, e.g.:
 - Cashier selling alcohol
 - Bouncer checking entree's age
 - Employer checking employees' IDs

3.4 Summary of user needs

Needs	Priority	Concerns	Current solutions	Proposed solutions
-------	----------	----------	-------------------	--------------------

Free	High	Users should not need to pay for the services.	None	Government investments for paid services, such that the application can be free for a common user.
Accessible	High	All users that are willing to use the service should be able to do that easily.	None	Available on all platforms and follow WCAG guidelines to make it universally accessible.
Secure and trusted	High	The user should be the only party that is able to access and manage its own account.	None	Follow OWASP top ten, implement secure standards e.g., OAuth 2.0.
Confidentiality	High	The user should be able to control access to each ID separately.	None	Store documents encrypted on device. Decryption(access) should only occur after biometrics or password has been correctly given.
Wide range of uses	Medium	Users can store various kinds of IDs and documents that can be used in a wide range of scenarios.	None	Create a general-purpose application that can be applied to any form of identification, thus storing a wide range of different document types.
Log	Low	Users can see a log of all usage of their IDs.	None	On the device, store all usage of IDs, and display them.

Availability	High	The user should always be able to verify themselves even if the ID provider has experienced downtime.	None	Using decentralized blockchain technology to store necessary information about ID providers and storing necessary ID documents on device.
Divisibility of data	Medium	A user should be able to decide what part of an ID should be shared.	None	None currently.
Manage IDs	High	A user should be able to add, refresh and remove IDs.	None	Store every ID from issuers on device, users can add new IDs by connecting with identity providers, refresh by connecting again, and remove by removing them from the device.

An issuer registering availability	High	An issuer should be able to register themselves as an available provider.	None	Create an endpoint for issuers to register themselves with a DID and a public key and store them using decentralized blockchain technology.
In-app verification	High	A user should be able to verify the ID of other users in physical situations.	None	Application generates a QR code for a verifier to scan resulting in them checking if the signature of the ID is valid.
Distance verification	High	An online service should be able to request verification	None	Frontend app offers a way or online services to request proof of ID from it.

3.5 Alternatives to our product

- A mobile application called Digital iD (Post u.d.) that encrypts and stores personal information. It was developed by Australia Post and can be used as a passport in Australia, driver license and more.
- Another mobile application called Yoti (Yoti u.d.), stores staff ID cards, student ID cards and professional certifications. The application uses the picture of a user’s passport to determine their digital ID. Additionally, Yoti has the option to store passwords. It uses QR code for validation.
- Another environment where a user stores its own digital identity is called Digidentity (Digidentity u.d.). It makes it easy and safe to log in to governments and companies, with a possibility to also link various products to the environment.

4. Product synopsis

4.1 The product's role in the user environment

User/Owner:

- Users can access their data through a mobile or web application.
- The product should be able to store and manage different identification documents for different purposes on a single mobile application. The user can then use these identifications when needed.
- Identification documents stored on the application can be used for online and physical verification.
- Users can choose what ID data they want to share.

Issuer/ID provider:

- An issuer is always able to register themselves as an ID provider.
- An issuer can experience downtime without affecting the application.

Verifier/Service provider:

- The application and its verification use case should always be available.
- The application can be used to verify other users' IDs both digitally and physically.

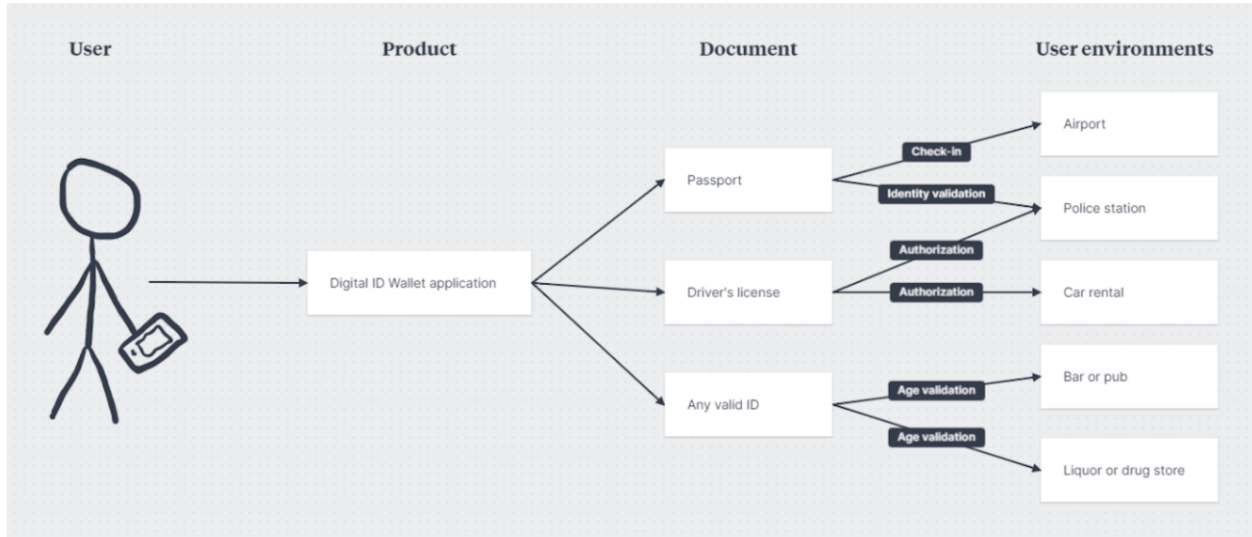


Figure 4.1: Examples of user environments where the product can be used by a common application user

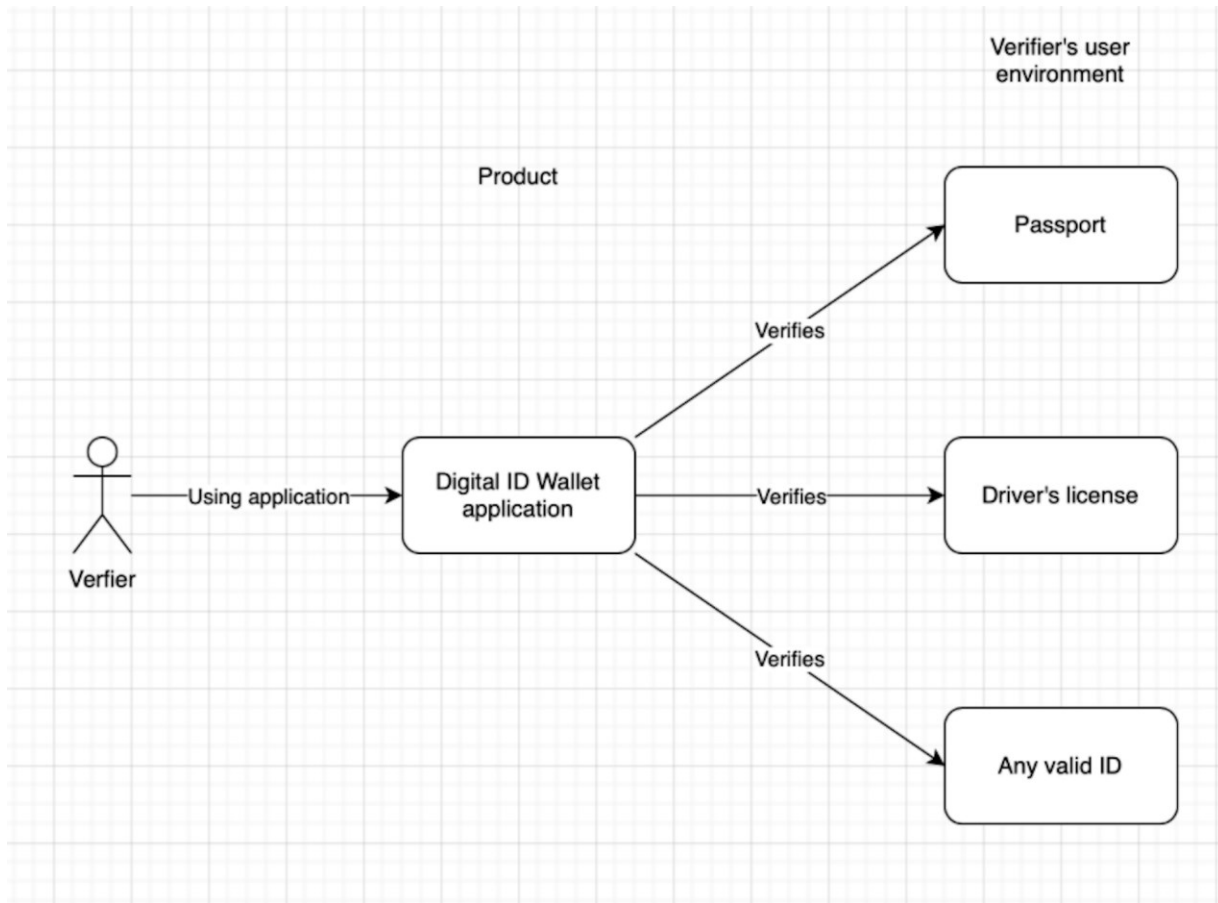


Figure 4.2: Examples of user environments for a verifier.

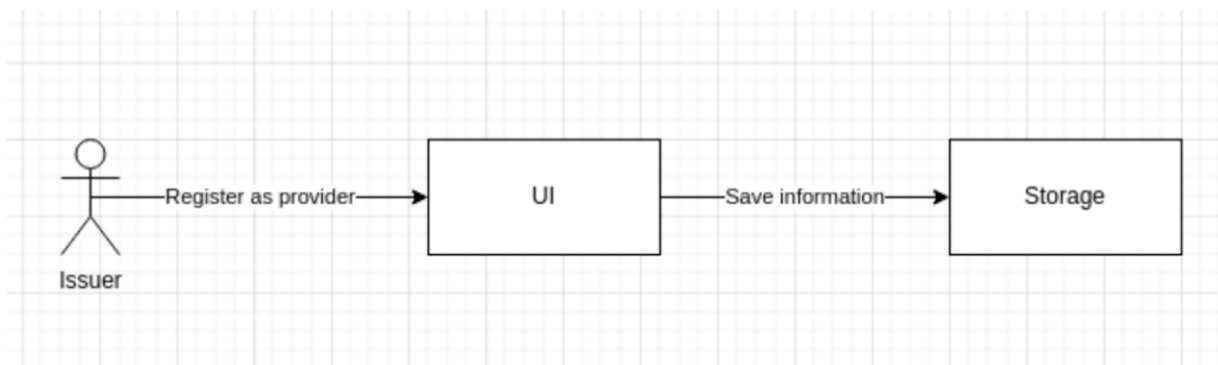


Figure 4.3: Examples of user environments for an identifier.

4.2 Prerequisites and dependencies

- None

5. Functional properties of the product

- ID storage

- Add ID
- Remove ID
- Refresh ID
- Encryption of ID
- ID viewing
 - Card display
 - Create access history for each ID
 - Search and filter IDs
 - Scroll through the IDs
- Verification
 - Generate QR-code to prove one's identity
 - Verify other identities by scanning other QR-codes
 - Websites and apps can request identification from the wallet
 - Selection of ID data to be used
 - Users must authenticate themselves before they can share their IDs
- Authentication
 - Unlock wallet with biometrics
 - Unlock wallet with password
- Provider registration
 - Register as ID provider with public key and DID.

6. Non-functional properties and other needs

- Secure
 - All personal data is stored on-device.
 - All personal data is encrypted.
 - ID transactions from providers to users are forever stored in a blockchain.
- Reliable
 - Users should always be able to fully utilize the application.
- Performance
 - Users can access their data efficiently.
- Privacy
 - Application should act as a protecting layer hindering ID providers from tracking users.

7. References

Digidentity. n.d. *Digidentity*. Accessed January 12, 2022. <https://www.digidentity.eu/>.
 Post, Australia. n.d. *DigitalID*. Accessed January 12, 2022. <https://www.digitalid.com>.
 Yoti. n.d. *Yoti*. Accessed January 12, 2022. <https://www.yoti.com/>.

Appendix G: Requirement document

**Digital ID Wallet App
Requirement documentation**

Version <1.9>

Revision history

Date	Version	Description	Author
13.01.2022	1.1	Initial revision and translate to English	Diderik, Rokas, Martin
21.01.2022	1.2	User stories and sequence diagram added	Diderik, Rokas
03.02.2022	1.3	User stories, domain model, sequence diagrams, prototypes	Rokas
10.02.2022	1.4	Enumerated user stories	Rokas
22.04.2022	1.5	Updated user stories	Rokas
05.05.2022	1.6	Updated domain model and table of contents	Martin
06.05.2022	1.7	Updated page numbers for table of contents	Martin
11.05.2022	1.8	Updated the DPI for the images included to 310, also updated the sequence diagrams, added captions to figures. Added subheadings and structured the table of contents	Martin
18.05.2022	1.9	Added page numbers	Rokas

Table of contents

Figures	iv
1. Introduction	5
2. User Stories	5
2.1 User/Owner	5
2.1.1 Case 1:	5
2.1.2 Case 2:	5
2.1.3 Case 3:	5
2.1.4 Case 4:	5
2.1.5 Case 5:	5
2.1.6 Case 6:	6
2.1.7 Case 7:	6
2.1.8 Case 8:	6
2.1.9 Case 9:	6
2.2 Verifier	6
2.2.1 Case 10:	6
2.3 Issuer	7
2.3.1 Case 11:	7
2.3.2 Case 12:	7
2.4 Developer	7
2.4.1 Case 13:	7
2.4.2 Case 14:	7
3. Domain model	8
3.1 Sequence diagrams	8
4. Prototypes	11
4.1 Wireframes	11
4.2 Storyboards	11

Figures

Figure 3.1: Domain model describing the problem domain	8
Figure 3.2: Sequence diagram of physical verification.....	9
Figure 3.3: Sequence diagram of issuing an ID	10
Figure 4.1: Storyboard describing applications use case during interview	11
Figure 4.2: Storyboard displaying physical verification.....	12

1. Introduction

This document contains technical requirements that describe how the final product will be. The document is used as a guidance during development and is consistently updated if any changes in technical requirements occur. The changes to the document can be seen in the revision history. To describe the technical requirements, the document contains user stories, as well as different models and prototypes.

2. User Stories

2.1 User/Owner

2.1.1 Case 1:

As a user

I wish to create a wallet

So that I can use the application and store my IDs.

Requirements:

- Must create an 8-digit code.
- Device is assigned a unique holder DID.
- If an error occurs, the user will be noted.
- User sent to home page when wallet creation was successful.

2.1.2 Case 2:

As a user

I wish to authenticate

So that I can use the application.

Requirements:

- The user has already created a wallet.
- Use the digit code authentication method.
- If an error occurs or bad credentials were sent, the user will be notified.

2.1.3 Case 3:

As a user

I wish to lock the application

So that I must authenticate myself again when entering the application.

Requirements:

- The user must authenticate next time they use the application.
- The application should lock itself after a certain amount of time of inactivity.

2.1.4 Case 4:

As a user

I wish to change my PIN-code

So that I can use another PIN-code

Requirements:

- The user must know the previous PIN-code.
- The user must select a new 8-digit code.
- Application notifies the user of failure or success.

2.1.5 Case 5:

As a user

I wish to add a new ID

So that I can use the document later for various purposes using the application.

Requirements:

- The user has access to this document through an ID provider.

- The ID provider sends the document to the user/application to be stored on device.
- A smart contract is stored on blockchain for later verification.
- Application notifies user of failure or success.

2.1.6 Case 6:

As a user

I wish to use one of my IDs

So that I can verify my identity to a third party physically.

Requirements:

- The user must authenticate themselves in the application to access/submit the document.
- The user can choose the ID data they want to share with the third party.
- The application can generate a QR-code that can be scanned or submitted to verify the document's integrity and legality.

2.1.7 Case 7:

As a user

I wish to use one of my IDs

So that I can verify my identity to a third party through the internet.

Requirements:

- The user is prompted to the application.
- The user must authenticate themselves in the application to access/submit the document.
- The user can choose the ID data they want to share with the third party.
- The verifier receives information about verification success.

2.1.8 Case 8:

As a user

I wish to remove an existing ID document from the application

So that the application does not have access to a document that is no longer valid.

Requirements:

- The user must authenticate themselves.
- The user must confirm removal.
- Application notifies the user of failure or success.

2.1.9 Case 9:

As a user

I wish to refresh my ID

So that the previous version of the ID is updated with new data.

Requirements:

- The user must have the ID already stored in the application.
- The new data must update existing ID data.
- Application notifies the user of failure or success.

2.2 Verifier

2.2.1 Case 10:

As a verifier

I wish to perform verification of ID from another user

So that I can confirm the user's identity.

Requirements:

- The verifier must open a scanner or code verifier on their application.
- The user must open an identity verification code on their application for the specific ID.

- The verifier can scan the user's verification code to ensure that the document is verified.
- The verifier then receives some identity information about the user being verified.

2.3 Issuer

2.3.1 Case 11:

As an issuer

I wish to register myself as an ID provider

So that users of the application can add my ID.

Requirements:

- The issuer uses an API endpoint or UI in the application.
- The issuer submits their DID.
- Application notifies the issuer of failure or success.

2.3.2 Case 12:

As an issuer

I wish to revoke issued IDs

So that users of the application can no longer use the ID.

Requirements:

- The issuer uses an API endpoint or UI in the application.
- Blockchain invalidates IDs.

2.4 Developer

2.4.1 Case 13:

As a developer

I wish to have sufficient test coverage

So that future changes do not break the application.

Requirements:

- Tests consist of both integration and unit tests.
- All public methods must be unit tested to give consistent responses to dummy input.
- All API-endpoints must be tested to behave correctly.

2.4.2 Case 14:

As a developer

I wish the application has implemented Continuous Integration and Delivery

So that the changes that are delivered to production are tested to ensure the application works.

Requirements:

- Application has sufficient test coverage.
- Only push changes to production if tests pass.

3. Domain model

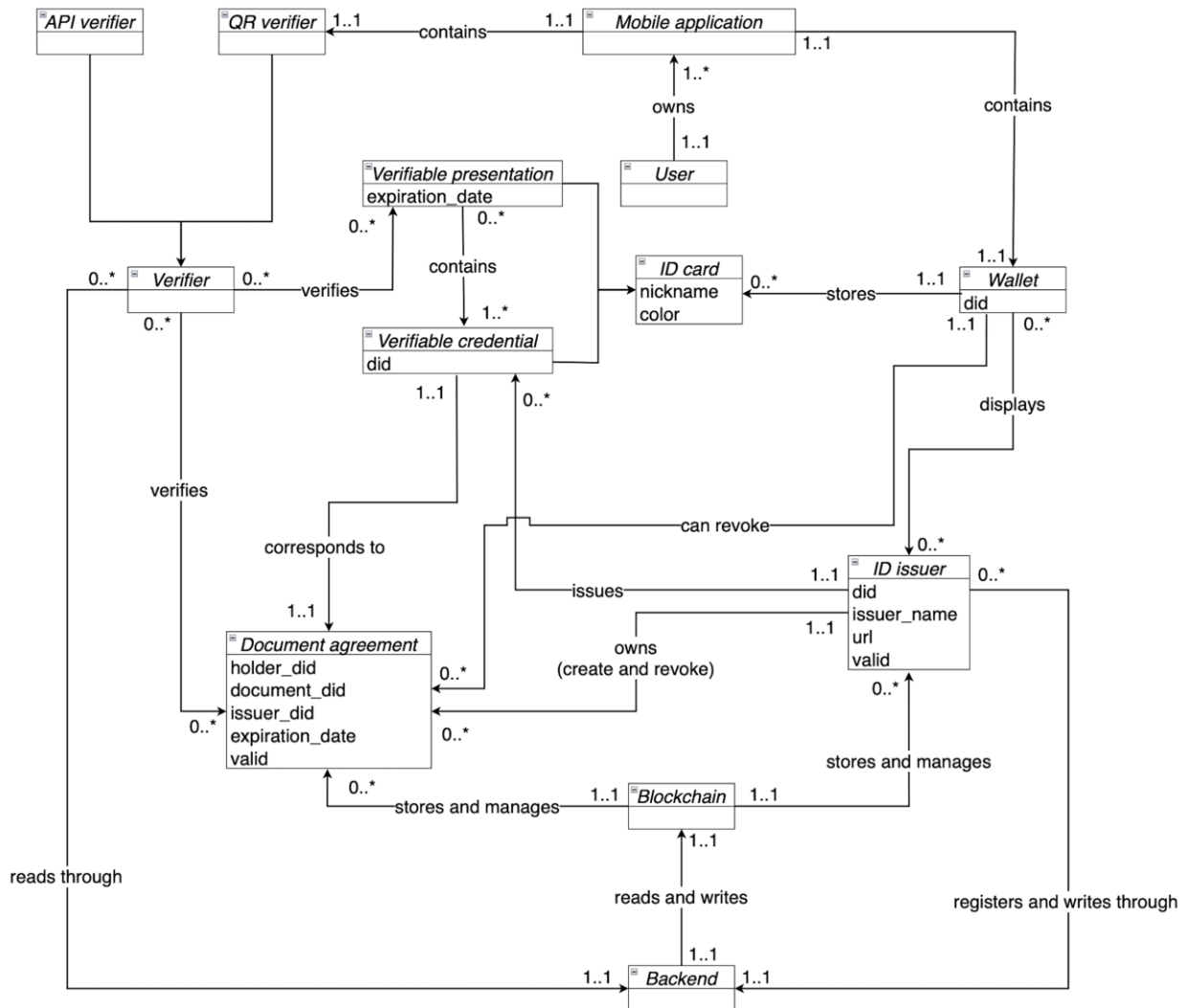


Figure 3.1: Domain model describing the problem domain

3.1 Sequence diagrams

Underneath is a scenario describing the action of physically verification. The verifier requests the whole identification document e.g., security at the airport requesting passport of travelers. The requested ID is sent through a QR code. Afterwards, the signature must be checked to ensure the integrity of the document. Then the verifier can check the documents validity in the DLT (blockchain).

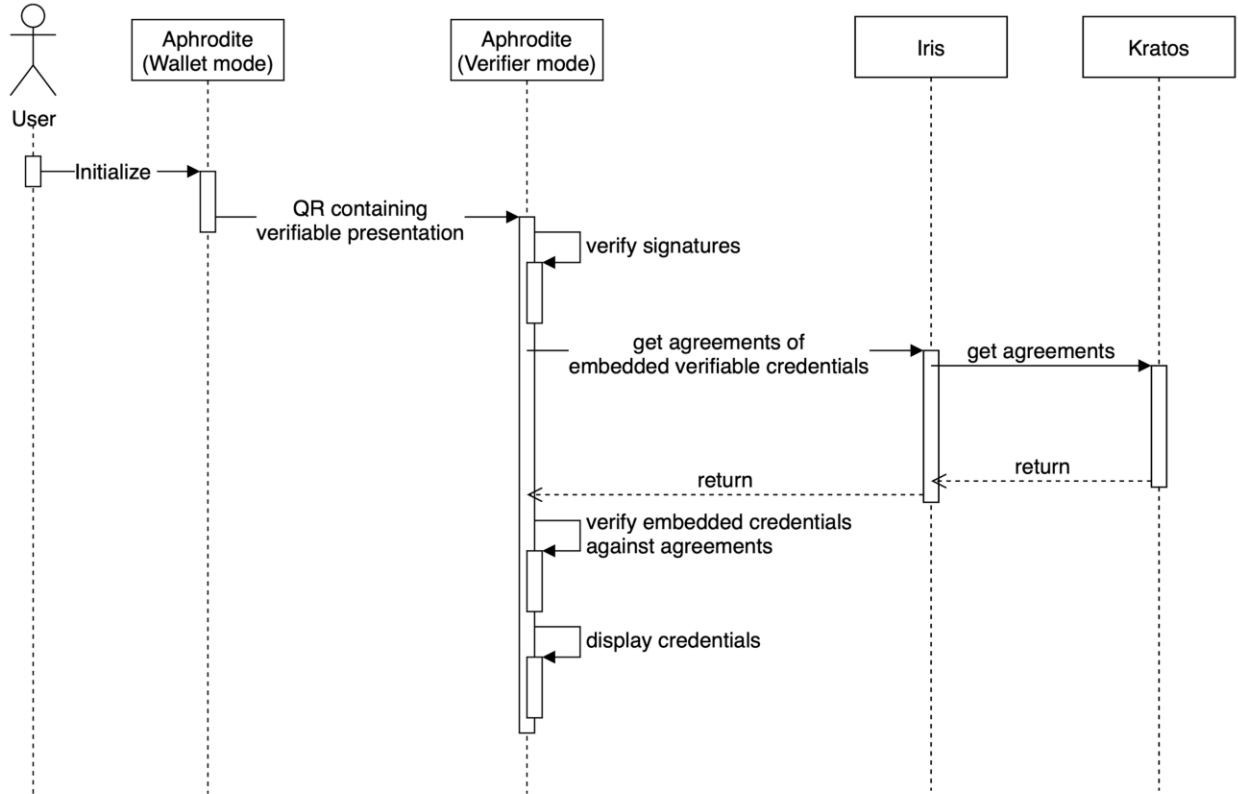


Figure 3.2: Sequence diagram of physical verification

The issuance of an ID from an issuer to a wallet is another action in the ecosystem. A centralized backend is responsible for contacting the ID provider by requesting a new ID. When the document is returned, it must be registered in the DLT (blockchain). The ID is then returned to the application for storage in the wallet.

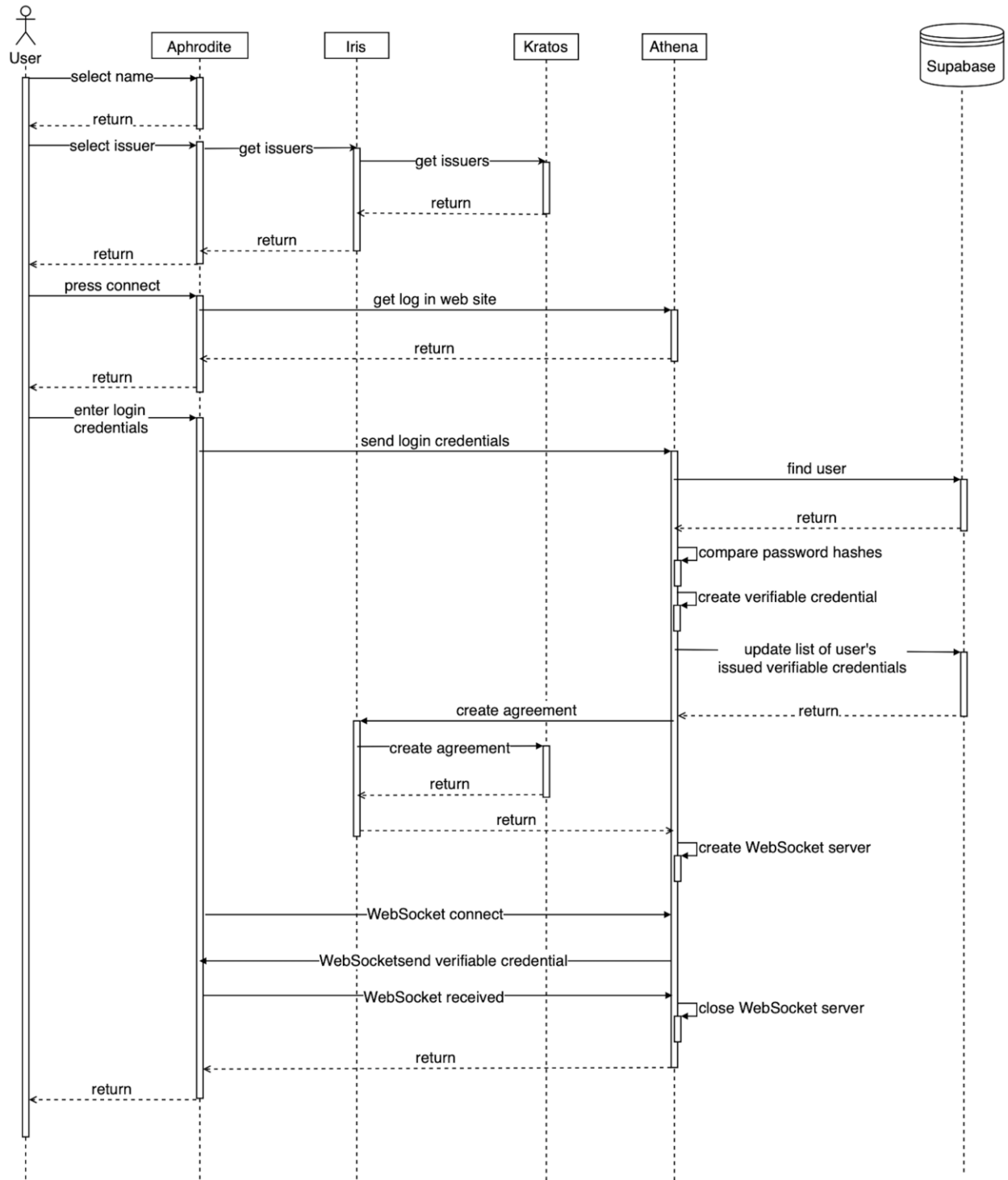


Figure 3.3: Sequence diagram of issuing an ID

4. Prototypes

This section consists of different prototypes to better visualize how the application will look like and perform.

4.1 Wireframes

Figma has been used for wireframing, anyone with the link should be able to view the wireframes, but not edit them: <https://www.figma.com/file/HebhRgDKdJ6548nRspAkdB/Bachelor-app-draft-%232?node-id=0%3A1>

4.2 Storyboards

The two figures below are storyboards that describe how physical verification may take place during a job interview.



Figure 4.1: Storyboard describing applications use case during interview

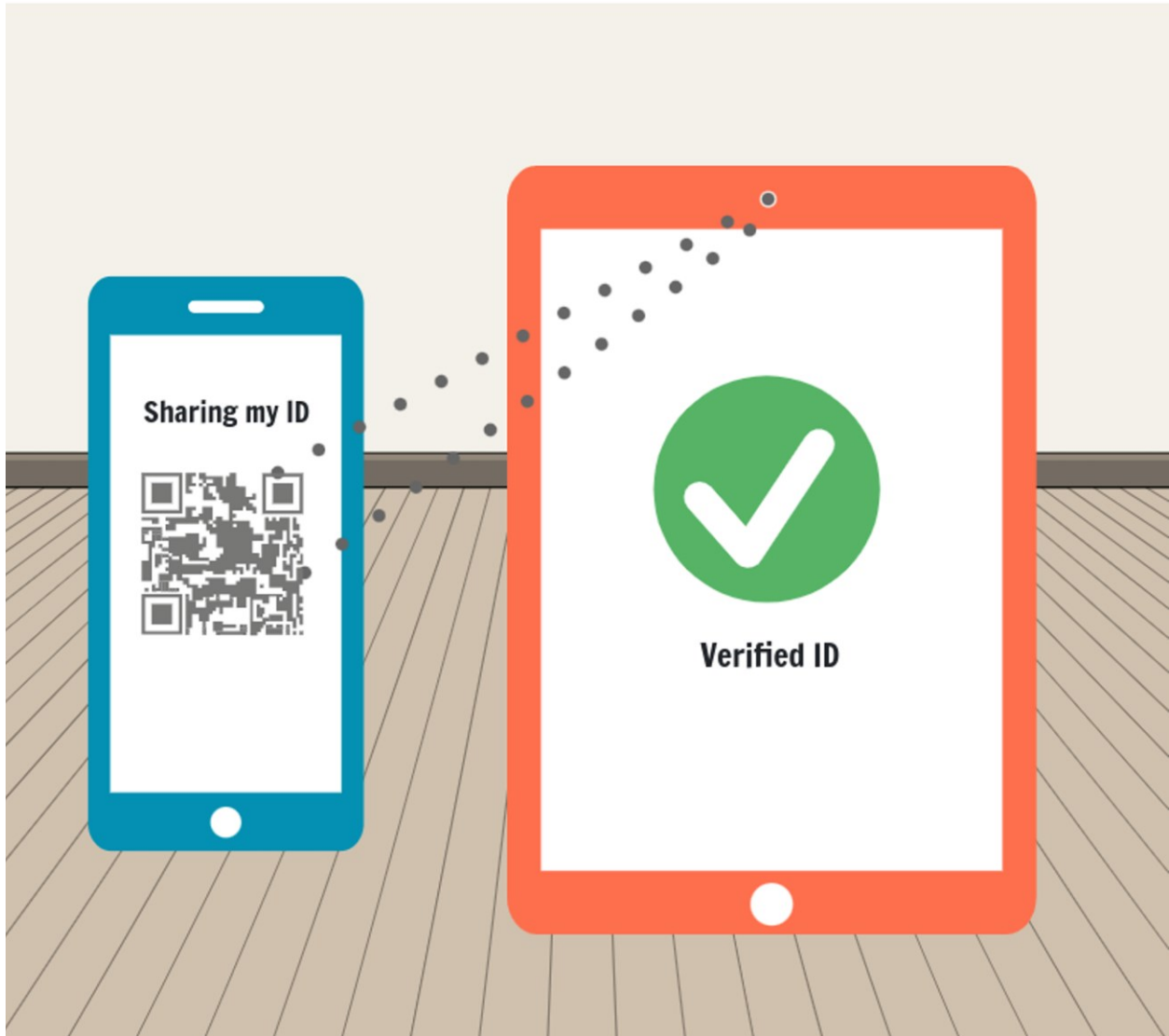


Figure 4.2: Storyboard displaying physical verification

Appendix H: System documentation

**Digital ID Wallet App
System documentation**

Version <1.6>

Revision history

Date	Version	Description	Author
11.01.2022	1.0	Initial revision	Diderik, Rokas, Martin
04.05.2022	1.1	Added continuous integration and testing	Diderik
05.05.2022	1.2	Added introduction, architecture, structure, security, installation manual	Diderik, Rokas, Martin
06.05.2022	1.3	Updated installation manual and security section	Rokas
11.05.2022	1.4	Updated pictures to 310 DPI	Martin
12.05.2022	1.5	Added captions to figure along with table of figures	Martin
18.05.2022	1.6	Fixed reference list	Rokas

Table of contents

Figures	iv
1. Introduction	5
2. Architecture	5
3. Project structure	7
3.1 Aphrodite	8
3.2 Athena	9
3.3 Iris	10
3.4 Kratos	11
4. Database model	12
4.1 PostgreSQL	12
4.2 Supabase	12
4.3 Redis	13
5. Server services	14
5.1 Athena	14
5.2 Iris	18
6. Security	21
7. Installation manual	22
7.1 Aphrodite	22
7.2 Athena	23
7.3 Iris	24
7.4 Kratos	24
8. Continuous integration and testing	26
9. References	26

Figures

Figure 2.1: The system's architecture, containing mock issuer, wallet, physical verifier, centralized backend, smart contract and blockchain components. As well as Supabase, PostgreSQL and Redis databases	5
Figure 2.2: Athena's architecture, containing web page, controller, service, connector and Supabase client components	6
Figure 2.3: Iris's architecture, containing web page, controller, service and Prisma ORM components	7
Figure 3.1: Aphrodite's project structure	8
Figure 3.2: Athena's project structure	9
Figure 3.3: Iris's project structure	10
Figure 3.4: Kratos's project structure	11
Figure 4.1: ER diagram of the PostgreSQL database	12
Figure 4.2: ER diagram of Supabase database	13
Figure 5.1: Part of Athena's API documentation	14
Figure 5.2: Part of Athena's API documentation	15
Figure 5.3: Part of Athena's API documentation	16
Figure 5.4: Part of Athena's API documentation	17
Figure 5.5: Part of Iris's API documentation	18
Figure 5.6: Part of Iris's API documentation	19
Figure 5.7: Part of Iris's API documentation	20

1. Introduction

This document is written to describe the entirety of the Digital ID Wallet project. By including both diagrams and text regarding topics such as REST API, it aims to give the reader a better understanding of the project’s architecture and structure, as well as the technicalities of the final MVP (Minimum Viable Product).

2. Architecture

There are three diagrams explaining the system architecture. The first one is the overall architecture, the second is Athena’s architecture and the third is Iris's architecture. All dashed arrows are components that are outside the system described in a diagram. All arrows include a description of the data that is exchanged between the components, and what operations can be done with the data. The operations are: C for create, R for read, U for update and D for delete/revoked (an agreement cannot be deleted but can be marked as revoked).

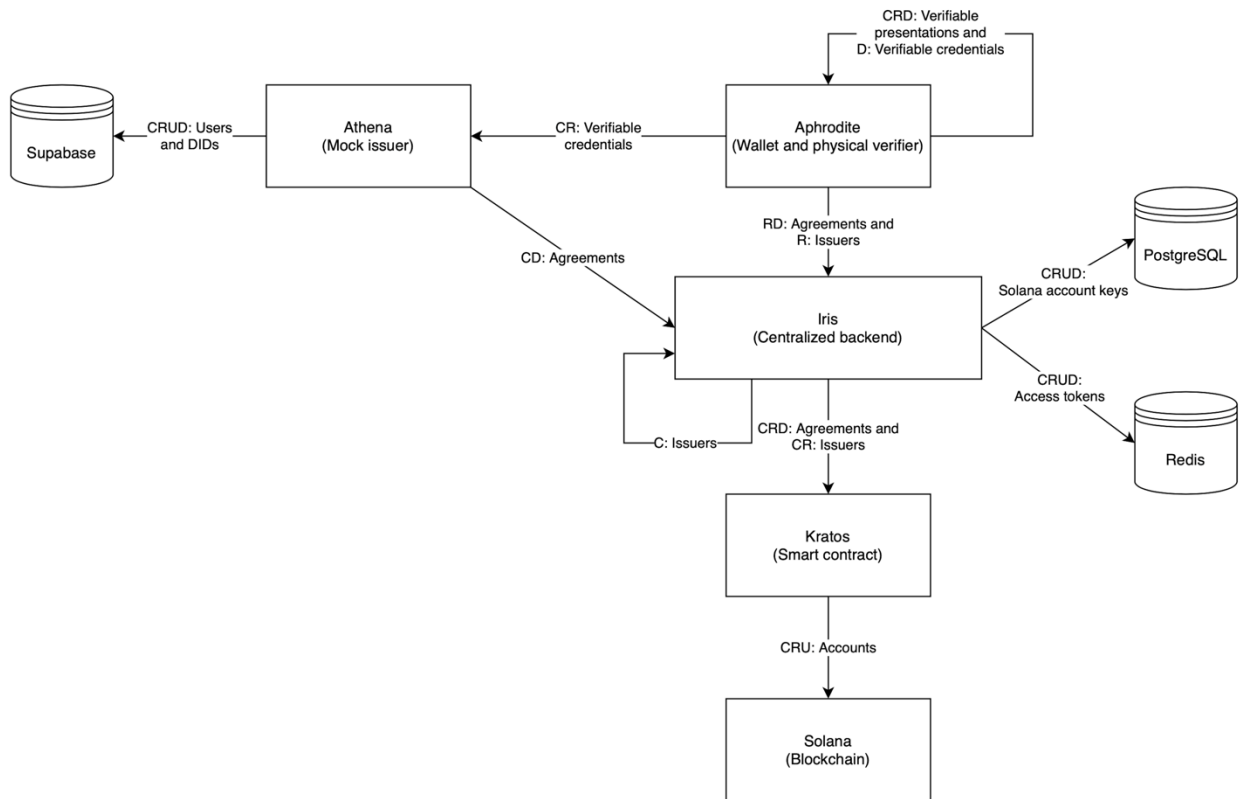


Figure 2.1: The system’s architecture, containing mock issuer, wallet, physical verifier, centralized backend, smart contract and blockchain components. As well as Supabase, PostgreSQL and Redis databases

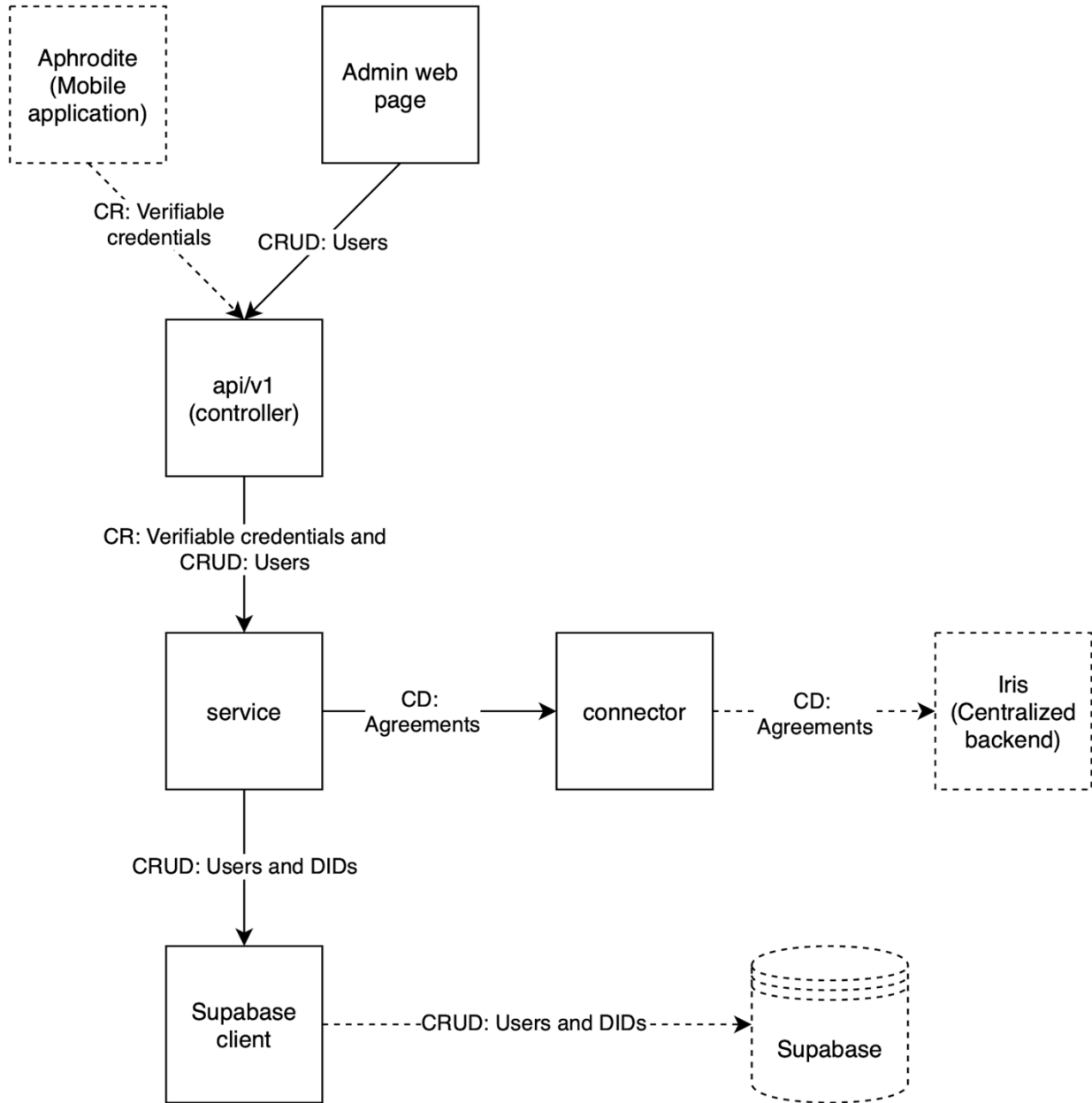


Figure 2.2: Athena’s architecture, containing web page, controller, service, connector and Supabase client components

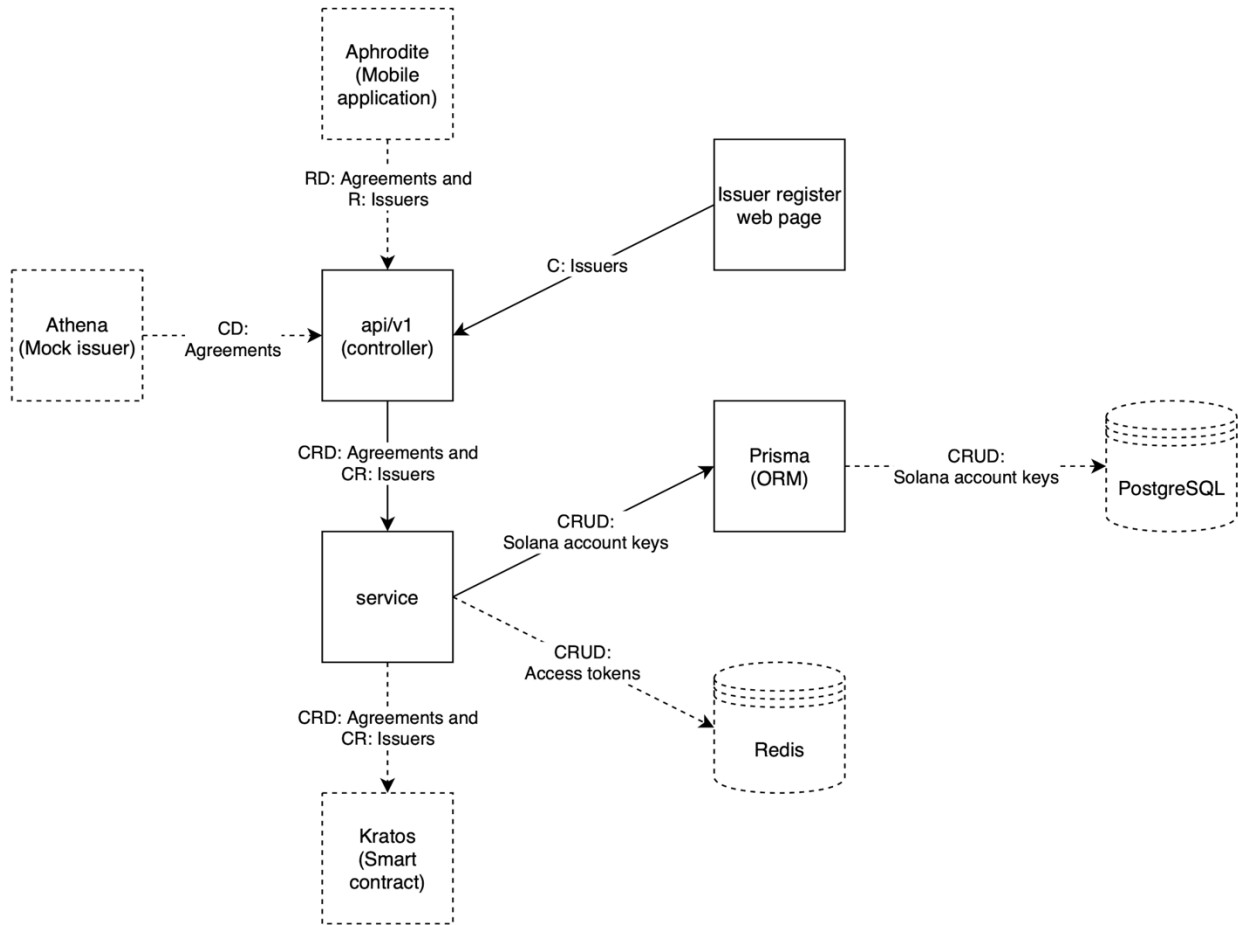


Figure 2.3: Iris’s architecture, containing web page, controller, service and Prisma ORM components

3. Project structure

The project is divided into four repositories:

Aphrodite – Frontend mobile application

Athena – Mock issuer

Iris – Backend

Kratos – Distributed ledger smart contract

3.1 Aphrodite

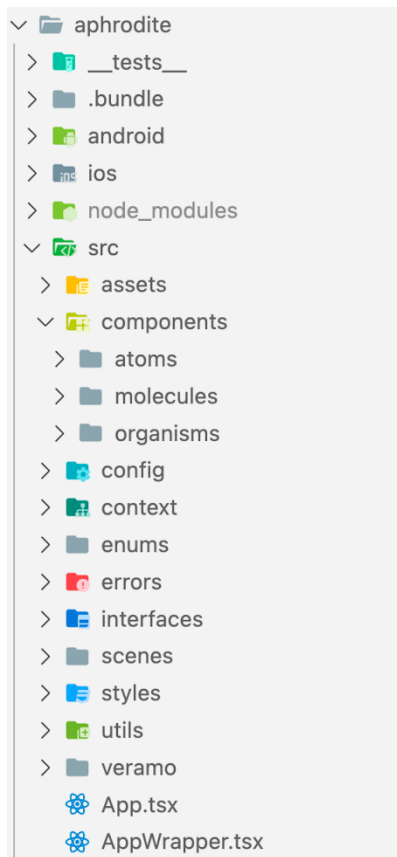


Figure 3.1: Aphrodite’s project structure

Aphrodite uses the React Native framework for creating a mobile application. Its most essential dependencies are the Veramo library for handling DID, Verifiable Credentials and Verifiable Presentations, rn-nodeify allowing developers to utilize Node libraries in a client JavaScript environment and lastly, React Native libraries.

Most folders are self-explanatory, for instance, the folders context, enums, interface and errors respectively contain React Contexts, JavaScript Enums, TypeScript interfaces and custom JavaScript Error classes. The assets folder contains external images or icons. The components folder contains smaller parts of a scene and is divided into three folders, atoms, molecules, organisms following the template created by Muskan Jain (Jain, 2021). Atoms contains the simplest components such as custom buttons. Molecules that are components built up of atom components.

Lastly, organisms combine molecules and atoms components to create complex structures. The scene folder contains components that can be navigated to using the React Native navigator, while the styles folder contains React Native style files, for most components and scene components, and the veramo folder contains necessary files to initialize and configure Veramo.

3.2 Athena

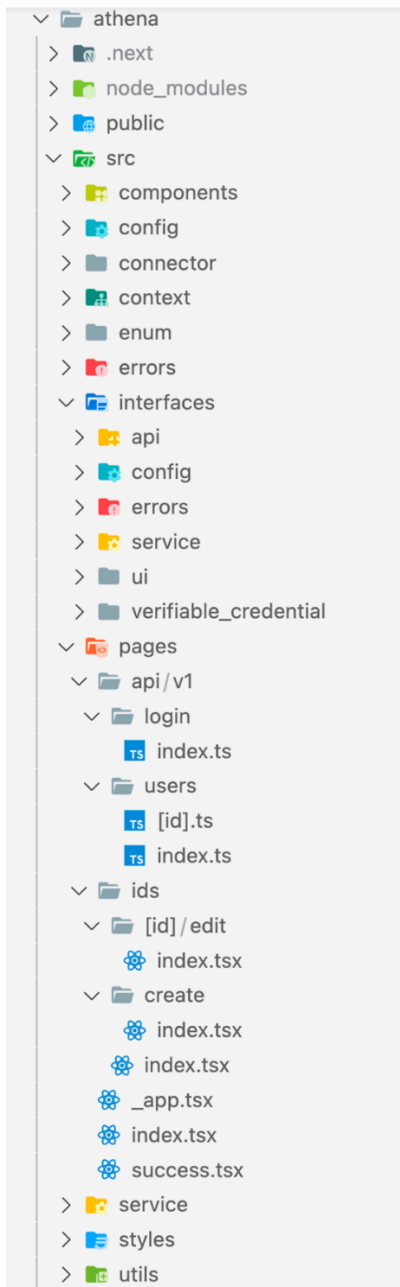


Figure 3.2: Athena’s project structure

Athena uses the Next JS framework for creating an MVC application. Its most essential dependencies are the Veramo library for handling DIDs, Verifiable Credentials and Verifiable Presentations, Supabase for storing necessary data and MUI for styling React Components and providing UI tools.

Most folders are self-explanatory, for instance will components, enum, errors and config respectively contain React components, JavaScript Enums, custom JavaScript Error classes and configurations for the Next server.

The pages folder contains the api/v1 folders that contain all API endpoints. Next JS is configured to use the file path to equal its URL endpoint for instance, api/v1/example or api/v1/example/index will both create the endpoint /api/v1/examples. “[key]” files also render endpoints but additionally allow for queries. For instance, api/v1/example/[key].ts would create the endpoints /api/v1/examples/abc and /api/v1/examples/bca etc. React files located directly inside the pages folder will render web pages. The URL structure is equivalent to the API endpoints, using the file path.

3.3 Iris

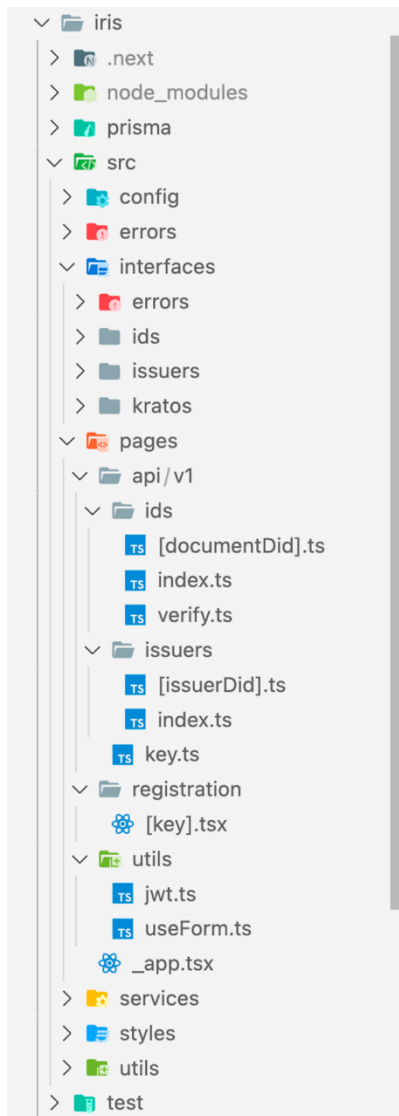


Figure 3.3: Iris's project structure

Iris uses the Next JS framework for creating an MVC application. Its most essential dependencies are the Veramo library for handling DIDs, Verifiable Credentials and Verifiable Presentations, the Prisma ORM for storing necessary data and MUI for styling React Components and providing UI tools. The structure is equivalent to Athena's structure.

3.4 Kratos

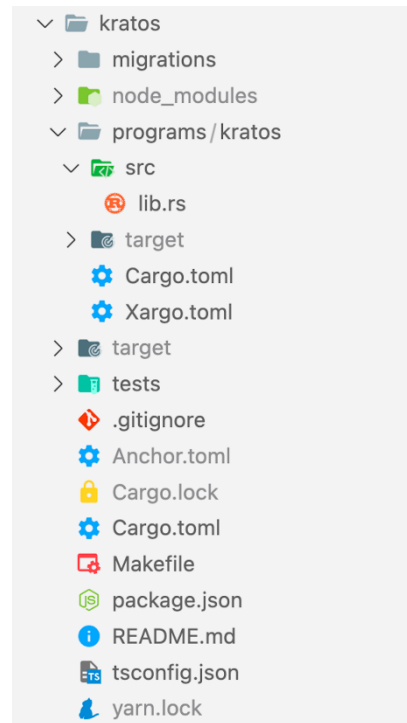


Figure 3.4: Kratos's project structure

Kratos uses the Anchor framework for creating smart contracts on Solana. All functionality is written in the file located at `/programs/kratos/src/lib.rs` from the root of the project. Secondly, `Anchor.toml` defined in the root of the project contains build settings and metadata, for instance, wallet address, deployment network and name.

4. Database model

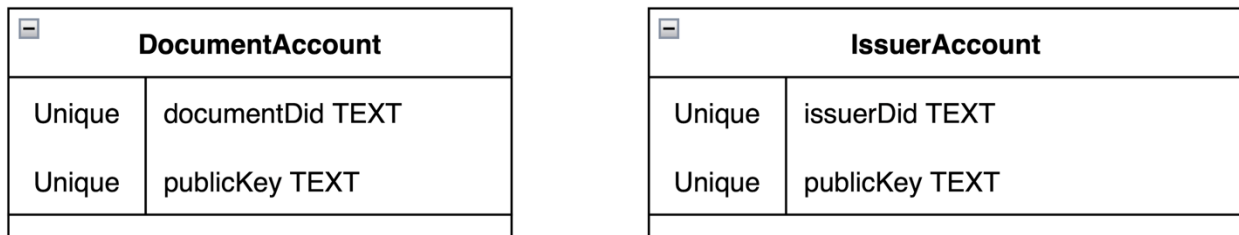
4.1 PostgreSQL

The PostgreSQL database is written to by the Prisma ORM. Using these models:

```
model IssuerAccount {
  issuerDid String @unique
  publicKey String @unique
}

model DocumentAccount {
  documentDid String @unique
  publicKey String @unique
}
```

By connecting to the PostgreSQL database through the command line, it can be seen that the tables have this schema:



DocumentAccount	
Unique	documentDid TEXT
Unique	publicKey TEXT

IssuerAccount	
Unique	issuerDid TEXT
Unique	publicKey TEXT

Figure 4.1: ER diagram of the PostgreSQL database

4.2 Supabase

Supabase features three tables. The users table contains information about the user such as personal ID and forename that is stored by the mock issuer and embedded in a verifiable credential as claims when a user requests it. The hash and salt are used for authentication purposes along with the personal ID. The expiration date signifies when the issuer will no longer make claims on behalf of the user, meaning it is embedded in requested verifiable credentials and passed on to the blockchain when new agreements are created. The document DIDs array is used to keep track of the verifiable credentials issued to a user, which is needed for revoking all agreements signed between the issuer and user, when a user is deleted from the system. The issuer DID keys table stores the private key and public key belonging to the issuer DID, which is needed for signing a verifiable credential. Both the private and public key are stored in the JSON format. The issuer DID table stores the DID document belonging to the issuer, which is required by Veramo.

Users	
PK	id INT8
UNIQUE	personalId VARCHAR hash VARCHAR salt VARCHAR forename VARCHAR surname VARCHAR expirationDate TIMESTAMPTZ documentDIDs VARCHAR[]

IssuerDIDKeys	
PK	kid VARCHAR
	publicKey JSONB privateKey JSONB

IssuerDID	
PK	did VARCHAR
	provider VARCHAR alias VARCHAR controllerKeyId VARCHAR keys JSONB service JSONB

Figure 4.2: ER diagram of Supabase database

4.3 Redis

All blacklisted JWT tokens are stored in the Redis in-memory store. The implemented Redis only use key values, where the key represents a JWT token which is pointing to a string value that by default is “invalid”. Therefore, there are no reasons for creating an ER-diagram.

5. Server services

The API documentation the repositories that have an API is written the corresponding README.md file.

5.1 Athena

API Documentation

Users

Register

POST /api/v1/users

Params:

```
{
  "personalId": "12345678901",
  "forename": "example forename",
  "surename": "example surname",
  "password": "Password123456",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Return:

Status 201

```
{
  "id": 1,
  "personalId": "12345678901",
  "forename": "example forename",
  "surname": "example surname",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Remove

DELETE /api/v1/users

Params:

```
{
  "ids": [
    "1",
    "4"
  ],
}
```

Return:

Status 204

Figure 5.1: Part of Athena's API documentation

Fetch

GET /api/v1/users

Return:

Status 200

```
{
  [
    {
      "id": 1,
      "personalId": "12345678901",
      "forename": "example forename1",
      "surname": "example surname1",
      "expirationDate": "2022-10-05T14:48:00.000Z"
    },
    {
      "id": 2,
      "personalId": "12345678902",
      "forename": "example forename2",
      "surname": "example surname2",
      "expirationDate": "2022-10-05T14:48:00.000Z"
    }
  ]
}
```

GET api/v1/users/{id}

Return:

Status 200

```
{
  "id": 1,
  "personalId": "12345678901",
  "forename": "example forename1",
  "surname": "example surname1",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Figure 5.2: Part of Athena's API documentation

Edit

PUT api/v1/users/{id}

Params:

```
{
  "personalId": "12345678902",
  "forename": "example forename",
  "surname": "example surname",
  "password": "Password123456",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Return:

Status 200

```
{
  "id": 1,
  "personalId": "12345678902",
  "forename": "example forename",
  "surname": "example surname",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Figure 5.3: Part of Athena's API documentation

ID

Fetch

POST /api/v1/login/?did=did:key:example123

Params:

```
{
  "personalId": "12345678901",
  "password": "Password123456",
}
```

Return:

Status 201

```
{
  "issuer": {
    "id": "did:key:z6MkjheWYC3AdV8UvFHcqWF3ZNvxbJRMab7HgJTxHqQUyRhN",
    "name": "Athena"
  },
  "credentialSubject": {
    "document": {
      "personalId": "12345678901",
      "forename": "example",
      "surname": "example",
      "expirationDate": "2097-04-30T11:20:24+00:00"
    },
    "id": "did:key:example123"
  },
  "id": "did:key:z6MkgvafhixiKASgp5vsU2heudpw568yUTUVM6YoKKiZkoA", "type": ["VerifiableCredential"],
  "@context": ["https://www.w3.org/2018/credentials/v1"], "issuanceDate": "2022-04-26T12:26:28.000Z",
  "proof": {
    "type": "JwtProof2020",
    "jwt": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ2IjYyI6eyJAY29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmlvdXQ6MzAwMC8vIiwiaWF0IjoiMjAyMi00LTI2VDE2OjI2OjI4LjAwMCZ"
  }
}
```

Figure 5.4: Part of Athena's API documentation

Athena also initializes a WebSocket endpoint when a user wants to fetch their endpoint, that closes after sending the Verifiable Credential. The endpoint is initialized at root path, meaning for instance if the server is hosted locally, the endpoint will be ws://localhost:3000/.

5.2 Iris

API Documentation

API Key

Generation

GET /api/v1/key (with Basic Auth)

Return:

```
{
  "key": "Example Key"
}
```

Issuers

Registration

POST /api/v1/issuers

Params:

```
{
  "issuer": {
    "issuerName": "example",
    "url": "https://example.com",
    "did": "example DID"
  },
  "token": "Example API Key"
}
```

Return:

```
{
  "blockHash": "4YRjMLbsTe5cPqbAB1nJv8XgMh5U74ZKT4Yws2yEYKPvBP6vysmB3K942VC566QXuSFVcc3gitk6YjZV7RYRuZsr",
  "publicKey": "9p1W6tT3mYWZaeJXeJ4eRNsY828rgmhNQMW84b6oS1T5"
}
```

Figure 5.5: Part of Iris's API documentation

Fetching

GET /api/v1/issuers

Return:

```
[
  {
    "did": "did:key:example",
    "issuerName": "Example 1",
    "url": "https://example1.com/",
    "valid": true
  },
  {
    "did": "did:key:123",
    "issuerName": "Example 2",
    "url": "https://example2.com",
    "valid": false
  },
]
```

ID agreements

Generation

POST /api/v1/ids

Params:

```
{
  "documentDid": "did:key:example",
  "issuerDid": "did:key:example",
  "holderDid": "did:key:example",
  "expirationDate": "2022-10-05T14:48:00.000Z"
}
```

Return:

```
{
  "blockHash": "4YRjMLbsTe5cPqbAB1nJv8XgMh5U74ZKT4Yws2yEYKpVBP6vysmB3K942VC566QXuSFVcC3gitk6YjZV7RYRuZsr",
  "publicKey": "9p1W6tT3mYWZaeJXeJ4eRNsy828rgmhNQMw84b6oS1T5"
}
```

Figure 5.6: Part of Iris's API documentation

Invalidating

DELETE /api/v1/ids/{documentDid}

Return:

```
{
  "blockHash": "4YRJmLbsTe5cPqbAB1nJv8XgMh5U74ZKT4YwS2yEYKpVBP6vysmB3K942VC566QXuSFVcC3gitk6YjZV7RYRuZsr",
  "publicKey": "9p1W6tT3mYWZaeJXeJ4eRNsY828rgmhNQMW84b6oS1T5"
}
```

Fetching

GET /api/v1/ids/{documentDid}

Return:

```
{
  "documentDid": "did:key:example",
  "issuerDid": "did:key:example",
  "holderDid": "did:key:example",
  "expirationDate": "2022-10-05T14:48:00.000Z",
  "valid": true
}
```

Figure 5.7: Part of Iris's API documentation

6. Security

Currently, there is no encrypted communication as the ecosystem is in an MVP phase and learning about encrypted communication is not in the scope of the thesis. However, encrypted communication needs to be implemented before the system can go to production.

Password hashing for both Aphrodite and Athena use the PBKDF2 algorithm (Kaliski, 2000) with sha-512 (Stinson & Paterson, 2018) where both the iteration count and derived key size is set in the environment files. Before deciding the number of iterations, the command `openssl speed` (OpenSSL, n.d.) should be run. By looking at the result of sha-512 for three seconds and the wished key size it can be determined how many iterations should be used for the wished number of seconds that authentication should take. Keep in mind that Aphrodite is run on mobile devices which are weaker than PCs and should therefore have a lower iteration count, if else it may crash.

Aphrodite takes multiple precautions to secure the user. It requires a pin code of 8 digits. If the user is inactive for a 5-minute period, the user will be kicked out. Also, all sensitive information in Aphrodite, such as verifiable credentials and wallet DID, are stored in an encrypted storage.

As Athena was to be made as simple as possible, it does not use access tokens as only one endpoint (/login) requires user credentials (added to simulate login process when retrieving a VC). Returning the data was decided to be a better solution rather than creating an unneeded access token endpoint. There is no authentication for the administrator functions either, meaning that Athena should not be deployed publicly, as it would enable anyone to CRUD users.

Iris utilizes JWT tokens to create “API keys” that protects the API endpoint for adding issuer to the ecosystem. The key generation endpoint is secured by Basic Authentication. A client therefore needs the admin username and password before receiving a key. The key is a JWT token with an expiration of one hour and signed by Iris using a random secret stored as an environment variable. After it has been validated, used and an issuer has been added to the ecosystem, the key is blacklisted in a Redis cache.

The ecosystem is considered safe against SQL-injection. As the connection to Supabase is done through their PostgreSQL API, prepared statements are used which hinders SQL-injection (GitHub, 2021). Prisma also prevents SQL-injections since it is an ORM, which saves the developer “from writing repetitive SQL statements for common CRUD operations and escaping user input to prevent vulnerabilities such as SQL injections” (Prisma, n.d.). The implemented Redis cache only receives used and validated JWT tokens. Therefore, it will never take any user input as parameters, meaning, no SQL injection or any type of security breach from user inputs will be possible.

React is used for creating the websites for Athena and Iris. In React’s documentation it is written that “React DOM escapes any value embedded in JSX” (React, n.d.), since the team has made sure to use JSX, XSS is unlikely to occur. As Aphrodite is written with React Native and that uses React (React Native, n.d.-a), it can be deduced that XSS is unlikely to occur.

There is also a security concern connected to the smart contract Kratos. When adding an agreement to the blockchain, the Kratos smart contract is called which creates a new account belonging to the agreement. When an account is created, it generates a keypair. In the current implementation, when an agreement is added to an account, its public key is returned to Iris and stored in PostgreSQL database using Prisma, together with the belonging document DID or issuer DID, depending on the agreement type. Thereafter, anyone can find the public key of an account by searching for the DID in the database. This means that anyone can fetch and view the agreement from the blockchain, which is an intended feature. Anyone should be able to fetch and view an agreement in case of verification. The strange part is that anyone can also “edit” the agreement in an account using the public key. This means that anyone can invalidate a document agreement, whether it is owned by the person or not. The public key acts like a private key, where any actions on an account can be made by using the public key. This is a security issue with either the Anchor framework or the Kratos smart contract implementation itself, since “editing” of an agreement should only be available for private key holders.

7. Installation manual

Dependencies:

- Node: JavaScript runtime environment (Node.js, n.d.)
- Solana: Blockchain (Solana, n.d.-b)
- Anchor: Solana framework (Anchor, n.d.-b)
- React Native: JavaScript framework for mobile application development (React Native, n.d.-b)
- Yarn or npm: package manager (Yarn, n.d.) (npm, n.d.)

7.1 Aphrodite

Currently only Android works, as a bug was discovered, that causes “yarn ios” to crash (to test on ios remember to run “pod install” in /ios folder). Due to the projects's time constraint, and that only one team member can run iOS (as MacOS developer platform is required), aborting the issue for now was decided.

1. Set up a development environment (React Native, 2022).
2. In the root of the repository, add an .env file containing:

```
RESET_APPLICATION_ON_STARTUP=false # Must be exactly "true" for
application to reset on startup. Else no reset is done.
IS_DEV=true # Must be exactly "true" for dev mode to be activated. Else
prod is used. Also, decides websocket url in EmbeddedWeb.tsx.
DEV_BACKEND_URL=10.0.2.2:3000 # Iris URL. 10.0.2.2 is the mobile version
of localhost.
DEV_PBKDF2_ITERATIONS=2048 # The amount of times the pbkdf2 algorithm
runs sha512
DEV_PBKDF2_KEY_SIZE=64 # The key size used in the pbkdf2 algorithm
PROD_BACKEND_URL=13.70.193.218:3000 # Iris URL
# VERY important to run "openssl speed" (keep in mind it has to be run
on a mobile device, sine this is a mobile app) and view how many sha512
iterations can be done in 3s for the selected key size and choose the
prod iterations thereafter.
# Aim for it to take enough time to hinder mass trial and error, but
also not long enough time to disturb the user.
# E.g., 0.5s which would mean dividing the iterations by 3s/0.5s = 6.
# Make sure to keep the iteration count and key size hidden to make it
harder for potential threat actors to crack passwords.
# If the iterations or key size change remember to reset the device as
the hash for the same password will not equal the previously saved
hash.
# The mobile device may crash on log in if the iteration count is too
high. In that case lower it, till it does not crash anymore.
PROD_PBKDF2_ITERATIONS=3767 # The amount of times the pbkdf2 algorithm
runs sha512
PROD_PBKDF2_KEY_SIZE=1024 # The key size used in the pbkdf2 algorithm
# These are used as indexes for storing the user details and the list
containing all IDs stored in the RESET_APPLICATION_ON_STARTUP
# The names do not matter to much but cannot be the same
# IF VALUES ARE CHANGED REMEMBER THAT ALL SAVED DATA WILL BE LOST AS
THEY WILL BE LOCATED AT THE OLD VALUES
# THIS CAN BE CONSIDERED THE SAME AS A DEVICE RESET
# The nickname of an ID card cannot be the same as these two
SECURE_STORE_USER_KEY=LOCALHOST_USER
SECURE_STORE_ID_CARD_LIST_KEY=LOCALHOST_ID_CARD_LIST
```
3. Install dependencies:

```
npm install
# or
```

- ```
yarn install
```
4. **Application is now ready for running, run server:**  

```
npm run start
```

```
or
```

```
yarn start
```
  5. **Connect your development environment to the server**  

```
npm run android
```

```
or
```

```
yarn android
```

## 7.2 Athena

1. **In the root of the repository, add a file called “.env.local” containing:**  

```
DEV_PROTOCOL=http # Protocol used when fetching data
DEV_SERVER_URL=localhost:3000 # Athena URL. Make sure that the server
starts on this domain and port. Cannot be the same as Iris URL.
DEV_SUPABASE_URL="https://example.supabase.co" # Valid supabase URL
DEV_SUPABASE_ANON_KEY="example key" # Valid supabase key
DEV_SUPABASE_USER_TABLE="Users" # User table name
DEV_BACKEND_URL=localhost:3001 # Iris URL. Cannot be the same as Athena
URL
This can be a lot less then the PROD version since no attacks are
expected.
DEV_PBKDF2_ITERATIONS=2048 # The amount of times th pbkdf2 algortihm
runs sha512
DEV_PBKDF2_KEY_SIZE=64 # The key size used in the pbkdf2 algorithm
IS_DEV=true # Must be exactly "true" for dev mode to be activated. Else
prod is used.
IMPORTANT that this is https to ensure encrypted communication
PROD_PROTOCOL=https # Protocol used when fetching data
PROD_SERVER_URL=13.70.193.218:3000 # Athena URL. Make sure that the
server starts on this domain and port. Cannot be the same as Iris URL
PROD_SUPABASE_URL="https://example.supabase.co" # Valid supabase URL
PROD_SUPABASE_ANON_KEY="example key" # Valid supabase key
PROD_SUPABASE_USER_TABLE="Users" # User table name
PROD_BACKEND_URL=localhost:3000 # Iris URL. Cannot be the same as Athena
URL
VERY important to run "openssl speed" and view how many sha512
iterations can be done in 3s for the selected key size and choose the
prod iterations thereafter.
Aim for it to take enough time to hinder mass trial and error, but
also not long enough time to disturb the user.
E.g., 0.5s which would mean dividing the iterations by 3s/0.5s = 6.
Make sure to keep the iteration count and key size hidden to make it
harder for potential threat actors to crack passwords.
PROD_PBKDF2_ITERATIONS=376890 # The amount of times th pbkdf2 algortihm
runs sha512
PROD_PBKDF2_KEY_SIZE=1024 # The key size used in the pbkdf2 algorithm
```
2. **Install dependencies:**  

```
npm install
```

```
or
```

```
yarn install
```
3. **Application is now ready for running, run development server:**  

```
npm run dev
```

```
or
```

```
yarn dev
```

### 7.3 Iris

1. Install Solana (Solana, n.d.-a).
2. If installed, run following commands in console:  
solana config set --url testnet  
solana airdrop 1  
solana config get keypair
3. In the root of the repository, add an .env file containing:  
DATABASE\_URL="postgres://example:5432/example" # Valid Postgres Database URL  
KEY\_PAIR\_PATH="/PATH/TO/SOLANA/WALLET/KEYPAIR" # Path received from "solana config get keypair"  
REDIS\_URL="redis://example" # Valid Redis Database URL  
JWT\_SECRET="VGhpcyBpcyBhbiBlyXN0ZXIgaZWdnIGhhaGFoYWg=" # Random String  
ADMIN\_USERNAME="admin"  
ADMIN\_PASSWORD="test"  
ANCHOR\_ADDRESS="https://api.testnet.solana.com"
4. Install dependencies:  
npm install  
# or  
yarn install
5. Application is now ready for running, run development server:  
npm run dev  
# or  
yarn dev

### 7.4 Kratos

1. Install Solana (Solana, n.d.-a) and Anchor (Anchor, n.d.-a).
2. In the root of the repository, add an Anchor.toml file containing:  
[features]  
seeds = false  
[programs.localnet]  
kratos = "KEYPAIR\_ADDRESS"  
  
[registry]  
url = "https://anchor.projectserum.com"  
  
[provider]  
cluster = "localnet"  
wallet = "KEYPAIR\_PATH"  
  
[scripts]  
test = "yarn run ts-mocha -p ./tsconfig.json -t 1000000 tests/\*\*/\*.ts"
3. Fetch the path to your wallet's public key:  
solana config get keypair
4. Replace KEYPAIR\_PATH in Anchor.toml with the output path string from the previous step.
5. Run the following command inside the project's root folder to create a keypair:  
solana-keygen new -o target/deploy/kratos-keypair.json
6. Replace KEYPAIR\_ADDRESS in Anchor.toml with the output pubkey. Go to programs/kratos/src/lib.rs and paste the same output where it says "declare\_id! ("PASTE\_IT\_HERE")"
7. Run this command to include the new program id in the binary:  
anchor build
8. Now it can either be deployed on a Solana cluster or run a local test validator.
  1. To run a local test validator:
    - a. Change the Solana configuration to localhost:  
solana config set --url localhost

- b. Change the provider.cluster variable in Anchor.toml:  
cluster = "localnet"
  - c. Run a local test validator:  
solana-test-validator
  - d. While running the test validator, open a separate terminal and airdrop some tokens  
solana airdrop 1
  - e. Deploy the program to localnet if not already done. Use the makefile command (make sure to replace the copy target directory in Makefile to the correct relative directory path to Iris if needed):  
make deploy\_and\_copy
  - f. (Optional) To connect Iris to the local test validator, the address in Iris src/config/anchor.ts file needs to be changed to the localhost address. You can get the url address (RPC URL) by running the command:  
solana config get
2. To deploy on a public Solana cluster:
- a. Depending on where the program is being deployed, change the provider.cluster variable in Anchor.toml:  
cluster = "devnet" / "testnet" / "mainnet-beta"
  - b. Set the solana config url to the same cluster as chosen above (replace <cluster> with devnet, testnet or mainnet-beta):  
solana config set --url <cluster>
  - c. Run the following command (make sure to replace the copy target directory in Makefile to the correct relative directory path to Iris if needed):  
make deploy\_and\_copy
  - d. If any errors arise while deploying, there might be an insufficient amount of tokens. If the program is deployed on a cluster other than the mainnet, tokens can be freely acquired with the following command as they are not real:  
solana airdrop 1
9. Make sure a (test) validator is not running on the machine used for testing. Then run this command to test the program:  
anchor test

## 8. Continuous integration and testing

Continuous integration and continuous deployment are only implemented in the backend part of the project. Gitlab is used for hosting their remote repository, and where the continuous integration and the continuous deployment is executed. When a merge request is submitted in the backend repository, it will begin executing a pipeline, which will chronologically execute jobs for installing dependencies, running the tests, before it deploys the changes. These will be retried if changes are made to the merge request's branch.

After installing the dependencies, a pipeline proceeds by executing tests written in the “test/” folder using the Jest library. The tests consist of both unit-tests and integration tests. The unit-tests are running positive and negative tests for the service functions and are used for validating that the API logic works as expected. External dependencies like database and blockchain connections are mocked. The integration is checking that most of the API endpoints work and respond as expected. These are mostly negative tests as there was no straightforward way in Next JS or Jest to inject dependencies for the database or blockchain connection. After completing all tests without failure, the pipeline continues by executing the last job, which is to deploy the changes to a virtual machine using SSH. As defined in the “gitlab-ci.yml” file, Gitlab runs the CI/CD on a Docker Node image using the 14.19.1 version. To run the tests manually, first navigate to the root of the backend folder and write the command in a command line “yarn install && yarn test”. This will install all necessary dependencies before running the tests.

## 9. References

- Anchor. (n.d.-a). *Installing Dependencies*. Retrieved May 5, 2022, from <https://project-serum.github.io/anchor/getting-started/installation.html>
- Anchor. (n.d.-b). *Introduction - The Anchor Book*. Retrieved May 18, 2022, from <https://book.anchor-lang.com/>
- GitHub. (2021). *Does supabase sanitize data before putting into database? #1452*. Retrieved May 5, 2022, from <https://github.com/supabase/supabase/discussions/1452>
- Jain, M. (2021, July 21). *Best Folder Structure for React Native Project*. Retrieved May 5, 2022, from Habilelabs: <https://learn.habilelabs.io/best-folder-structure-for-react-native-project-a46405bdba7>
- Kaliski, B. (2000, September). *Password-Based Cryptography Specification*. Retrieved May 5, 2022, from <https://datatracker.ietf.org/doc/html/rfc2898#section-5.2>
- Node.js. (n.d.). *Node.js*. Retrieved May 5, 2022, from <https://nodejs.org/en/>
- npm. (n.d.). *npm*. Retrieved May 5, 2022, from <https://www.npmjs.com/>
- OpenSSL. (n.d.). *openssl speed*. Retrieved May 5, 2022, from <https://www.openssl.org/docs/man1.1.1/man1/openssl-speed.html>
- Prisma. (n.d.). *Is Prisma an ORM?* Retrieved May 5, 2022, from <https://www.prisma.io/docs/concepts/overview/prisma-in-your-stack/is-prisma-an-orm#benefits-of-orms>
- React. (n.d.). *Introducing JSX*. Retrieved May 5, 2022, from <https://reactjs.org/docs/introducing-jsx.html>
- React Native. (2022). *Setting up the development environment*. Retrieved May 5, 2022, from <https://reactnative.dev/docs/environment-setup>
- React Native. (n.d.-a). *React Fundamentals*. Retrieved May 5, 2022, from <https://reactnative.dev/docs/intro-react>
- React Native. (n.d.-b). *React Native - Learn once, write anywhere*. Retrieved May 5, 2022, from <https://reactnative.dev/>
- Solana. (n.d.-a). *Install the Solana Tool Suite*. Retrieved May 5, 2022, from <https://docs.solana.com/cli/install-solana-cli-tools>
- Solana. (n.d.-b). *Solana*. Retrieved May 5, 2022, from <https://solana.com/>
- Stinson, D. R., & Paterson, M. (2018). *Cryptography: Theory and Practice* (4th ed.). New York: Taylor & Francis Group.
- Yarn. (n.d.). *Home | Yarn - Package Manager*. Retrieved May 5, 2022, from <https://yarnpkg.com/>



# **Appendix I: Manual testing**

---

**077**

**Digital ID Wallet App  
Manual Testing**

**Version <1.3>**

## Revision history

| <b>Date</b> | <b>Version</b> | <b>Description</b>                                                                                             | <b>Author</b> |
|-------------|----------------|----------------------------------------------------------------------------------------------------------------|---------------|
| 08.05.2022  | 1.0            | Initial revision and translate to English                                                                      | Martin        |
| 09.05.2022  | 1.1            | Added pictures of the tests in chapter 4.1-4.3, along with front page, revision history and table of contents. | Martin        |
| 10.05.2022  | 1.2            | Added pictures for tests in chapter 4.4 and on.                                                                | Martin        |
| 11.05.2022  | 1.3            | Also added captions for all figures.                                                                           | Martin        |

# Table of contents

|                                  |           |
|----------------------------------|-----------|
| <b>Figures</b>                   | <b>4</b>  |
| <b>1 Purpose</b>                 | <b>6</b>  |
| <b>2 How to run repositories</b> | <b>6</b>  |
| <b>3 Before starting</b>         | <b>6</b>  |
| <b>4 Tests</b>                   | <b>6</b>  |
| <b>4.1 Case 1</b>                | <b>6</b>  |
| <b>4.2 Case 2</b>                | <b>9</b>  |
| <b>4.3 Case 5</b>                | <b>12</b> |
| <b>4.4 Case 6/10</b>             | <b>17</b> |
| <b>4.5 Case 8</b>                | <b>19</b> |
| <b>4.6 Case 11</b>               | <b>23</b> |
| <b>4.7 Case 12</b>               | <b>29</b> |

## Figures

|                                                                                                                  |    |
|------------------------------------------------------------------------------------------------------------------|----|
| Figure 3.1: Section of Aphrodite's environment file                                                              | 6  |
| Figure 3.2: Section of Athena's environment file                                                                 | 6  |
| Figure 4.1: Section of Aphrodite's environment file                                                              | 7  |
| Figure 4.2: The startup page                                                                                     | 7  |
| Figure 4.3: The registration process of creating and confirming a PIN code                                       | 8  |
| Figure 4.4: The home page along with a confirmation message for new users                                        | 9  |
| Figure 4.5: Section of Aphrodite's environment file                                                              | 9  |
| Figure 4.6: The login screen                                                                                     | 10 |
| Figure 4.7: The screen when a wrong PIN is entered                                                               | 11 |
| Figure 4.8: The screen when a correct PIN is entered, after a wrong attempt                                      | 12 |
| Figure 4.9: The issuer's registration schema for creating a new user                                             | 13 |
| Figure 4.10: The issuer's overview of registered users                                                           | 13 |
| Figure 4.11: A row from the user table in Supabase                                                               | 13 |
| Figure 4.12: The start of the process of adding an ID                                                            | 14 |
| Figure 4.13: The process of connecting and sending the user's credentials to the selected issuer                 | 15 |
| Figure 4.14: The screen when the ID has been sent from the issuer to the application                             | 16 |
| Figure 4.15: A row from the user table in Supabase, containing data such as the document DIDs linked to the user | 16 |
| Figure 4.16: A row from the users table in Supabase, with the option to copy the documentDIDs column             | 16 |
| Figure 4.17: A REST call for viewing the data for the newly created agreement                                    | 17 |
| Figure 4.18: The process of generating a QR-code for physical verification                                       | 18 |
| Figure 4.19: The process of scanning a QR-code for verification                                                  | 19 |
| Figure 4.20: The screen displaying the verification status and the data sent in the ID                           | 19 |
| Figure 4.21: A REST API call to view the existing agreement                                                      | 20 |
| Figure 4.22: The process of deleting an ID from the mobile application                                           | 21 |
| Figure 4.23: Metro, giving the option to restart the application                                                 | 21 |
| Figure 4.24: The home screen without the previously deleted ID                                                   | 22 |
| Figure 4.25: A REST API call to view the same agreement after it has been revoked                                | 23 |
| Figure 4.26: The process of logging in to a user that has already added some IDs                                 | 24 |
| Figure 4.27: The process of exiting from the add ID screen to the home screen                                    | 25 |
| Figure 4.28: The REST API call necessary to retrieve a token for issuer registration                             | 26 |
| Figure 4.29: The issuer registration web page                                                                    | 26 |
| Figure 4.30: A REST API call for getting all registered issuers                                                  | 27 |
| Figure 4.31: The issuer registration web page filled in with credentials for registration                        | 28 |
| Figure 4.32: The success message after an issuer registration                                                    | 28 |
| Figure 4.33: The process of viewing the available issuers, starting from the home screen                         | 29 |
| Figure 4.34: The documentDIDs of a user being displayed by the Supabase table editor                             | 29 |
| Figure 4.35: The process of physically verifying an ID through a QR-code                                         | 30 |
| Figure 4.36: The issuer's overview of users                                                                      | 31 |
| Figure 4.37: The issuer's overview of users, with one user being selected                                        | 31 |
| Figure 4.38: The issuer's overview of users, with no users left                                                  | 31 |
| Figure 4.39: The Supabase table editor for the user table, with no users left                                    | 32 |

Figure 4.40: A REST API call for retrieving an agreement

32

Figure 4.41: The process of physically verifying an ID through a QR-code

33

## 1 Purpose

This document is used to give an overview of manual test that can be done for different use cases from the requirement document, along with pictures of the result from when the team completed them. Therefore, working as proof that the use cases in this document have been accomplished.

## 2 How to run repositories

Read the system documentation or the README.md file of the corresponding repository. E.g., Athena.

## 3 Before starting

Keep in mind that process that communicate with the blockchain may take some time.

For the tests it is assumed that Athena runs on localhost:3000 and that Iris runs on localhost:3001, as well that the ecosystem is in developer mode. This means that Aphrodite, Athena and Iris will run locally. While Kratos runs on the test net. Both Supabase and Prisma run on their designated cloud server. Make sure to change the corresponding dotenv file values from that of those in the README.md files to reflect that.

For Aphrodite (.env):

```
IS_DEV=true
DEV_BACKEND_URL=10.0.2.2:3001
```

Figure 3.1: Section of Aphrodite's environment file

For Athena (.env.local):

```
IS_DEV=true
DEV_SERVER_URL=localhost:3000
DEV_BACKEND_URL=localhost:3001
DEV_PROTOCOL=http
```

Figure 3.2: Section of Athena's environment file

## 4 Tests

### 4.1 Case 1

Run a reset instance of Aphrodite. Meaning that RESET\_APPLICATION\_ON\_STARTUP is set to exactly "true".

```
.env aphrodite x .env.local .env iris
aphrodite > .env
1 RESET_APPLICATION_ON_STARTUP=true|
2 IS_DEV=true
3 DEV_BACKEND_URL=10.0.2.2:3001
4 PROD_BACKEND_URL=13.70.193.218:3000
5
```

Figure 4.1: Section of Aphrodite's environment file

Press the get started button.

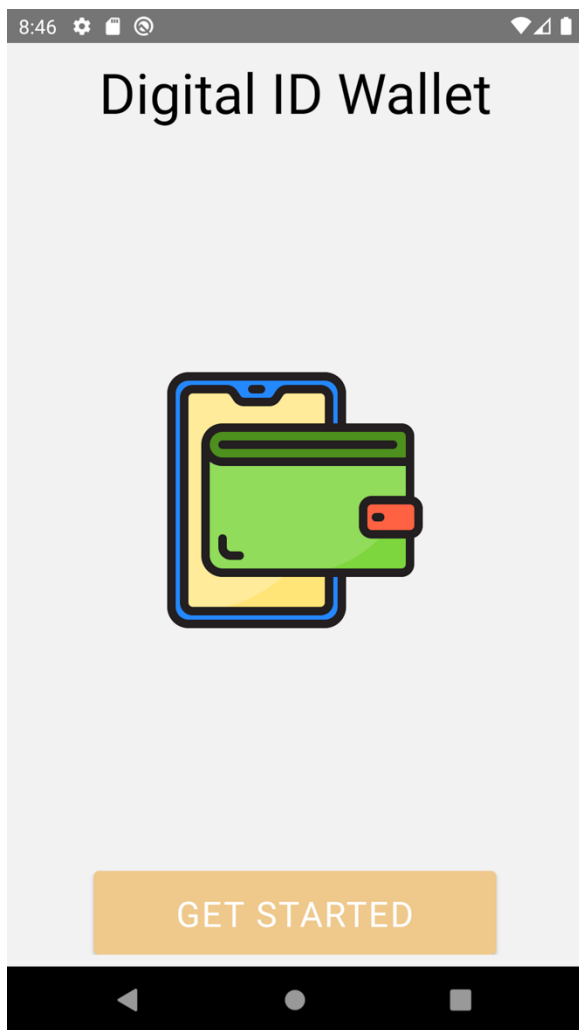
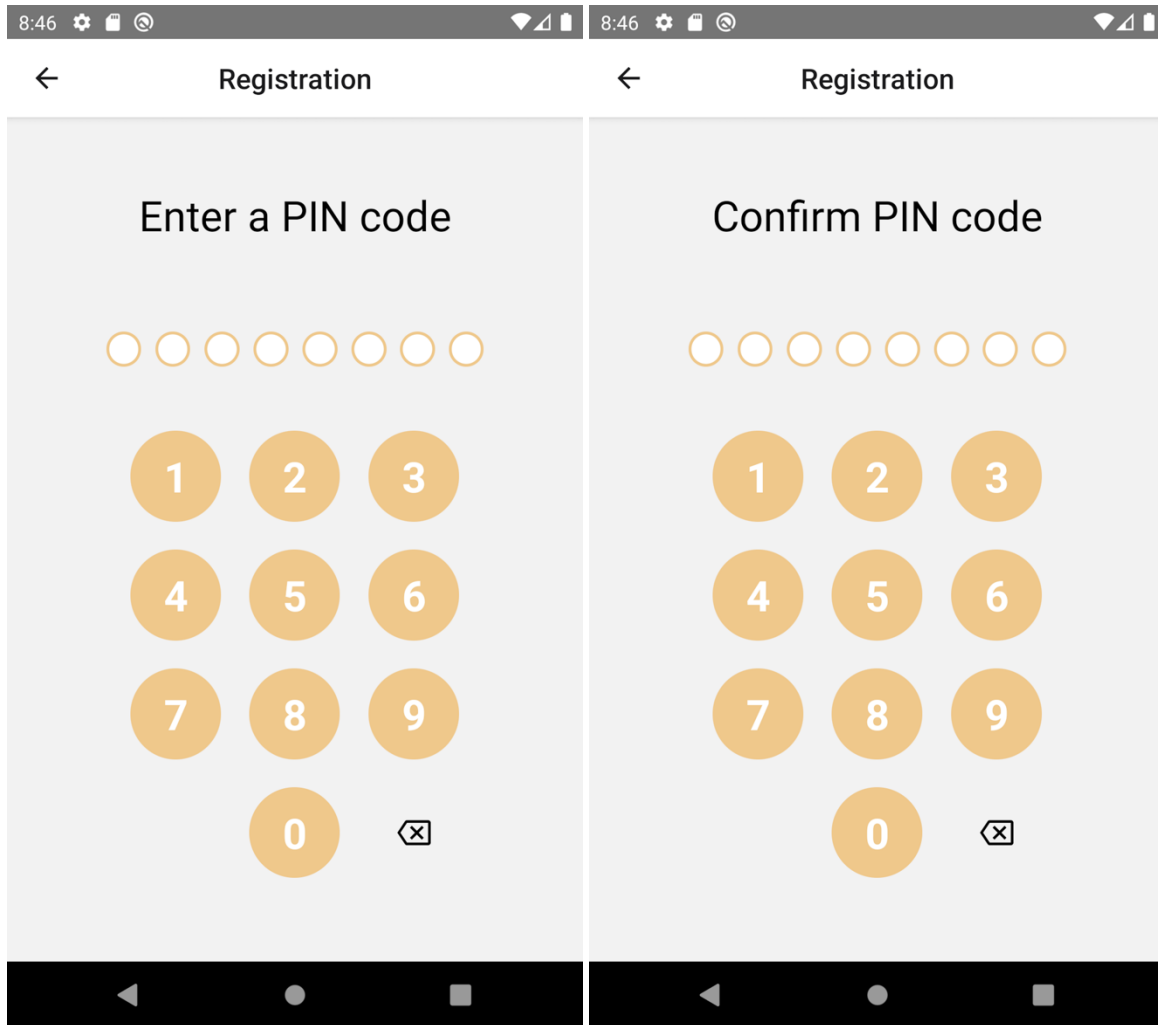


Figure 4.2: The startup page

Enter a PIN-code, then confirm it.





**Figure 4.3: The registration process of creating and confirming a PIN code**

Then a message confirming the creation of the wallet appears.

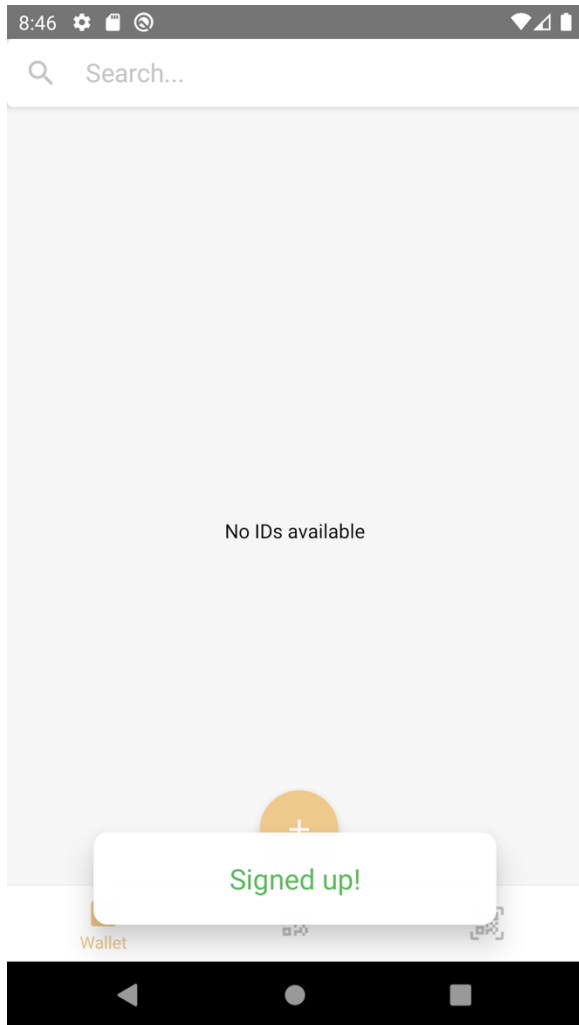


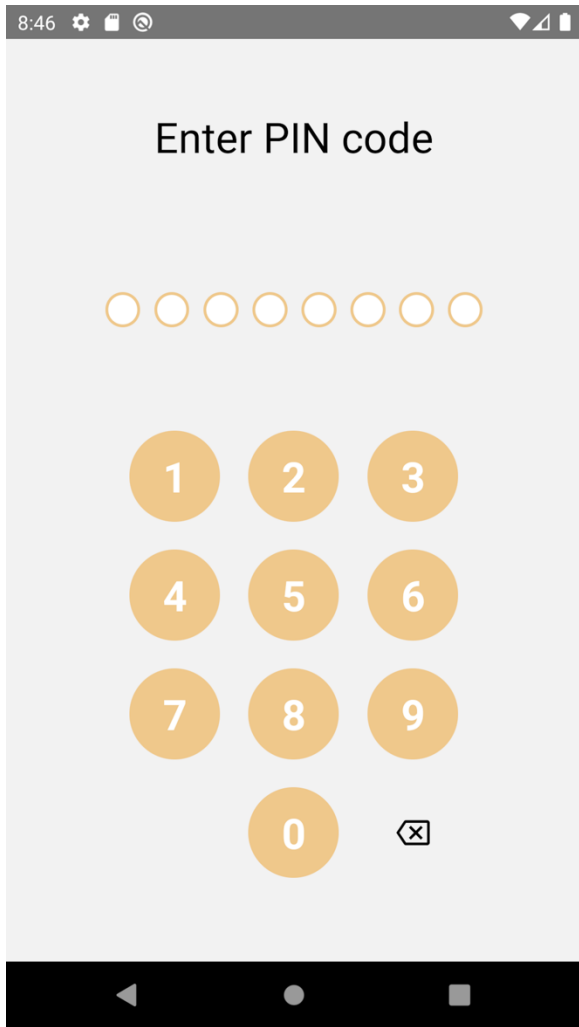
Figure 4.4: The home page along with a confirmation message for new users

## 4.2 Case 2

Run an instance of Aphrodite where a user has been created as in case 1. However, make sure to set the `RESET_APPLICATION_ON_STARTUP` environment variable to `false` and restart the application (run `yarn start` and `yarn android` again to make sure it loads the new environment variables).

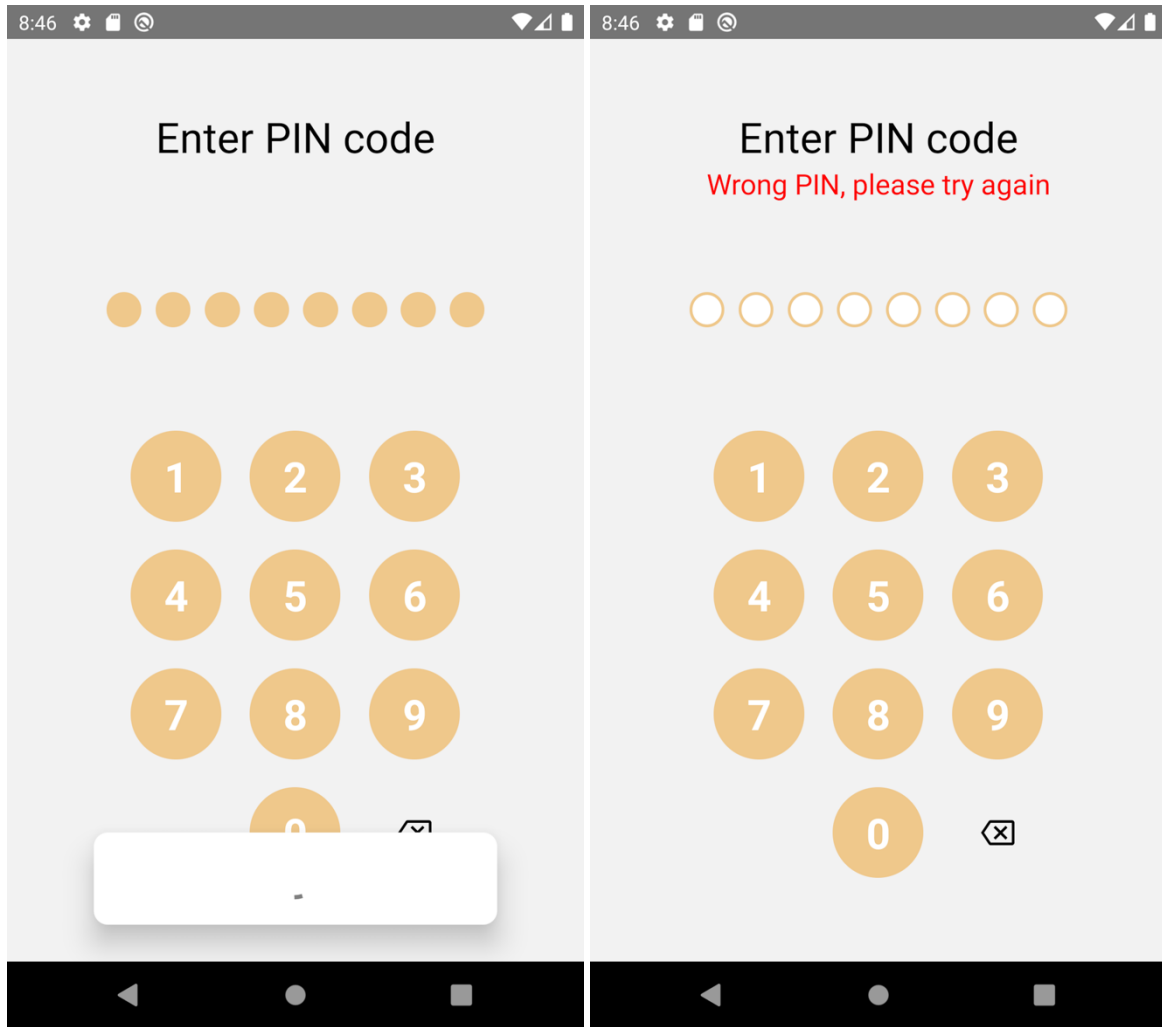
```
.env ×
aphrodite > .env
1 RESET_APPLICATION_ON_STARTUP=false
2 IS_DEV=true
3 DEV_BACKEND_URL=10.0.2.2:3001
```

Figure 4.5: Section of Aphrodite's environment file



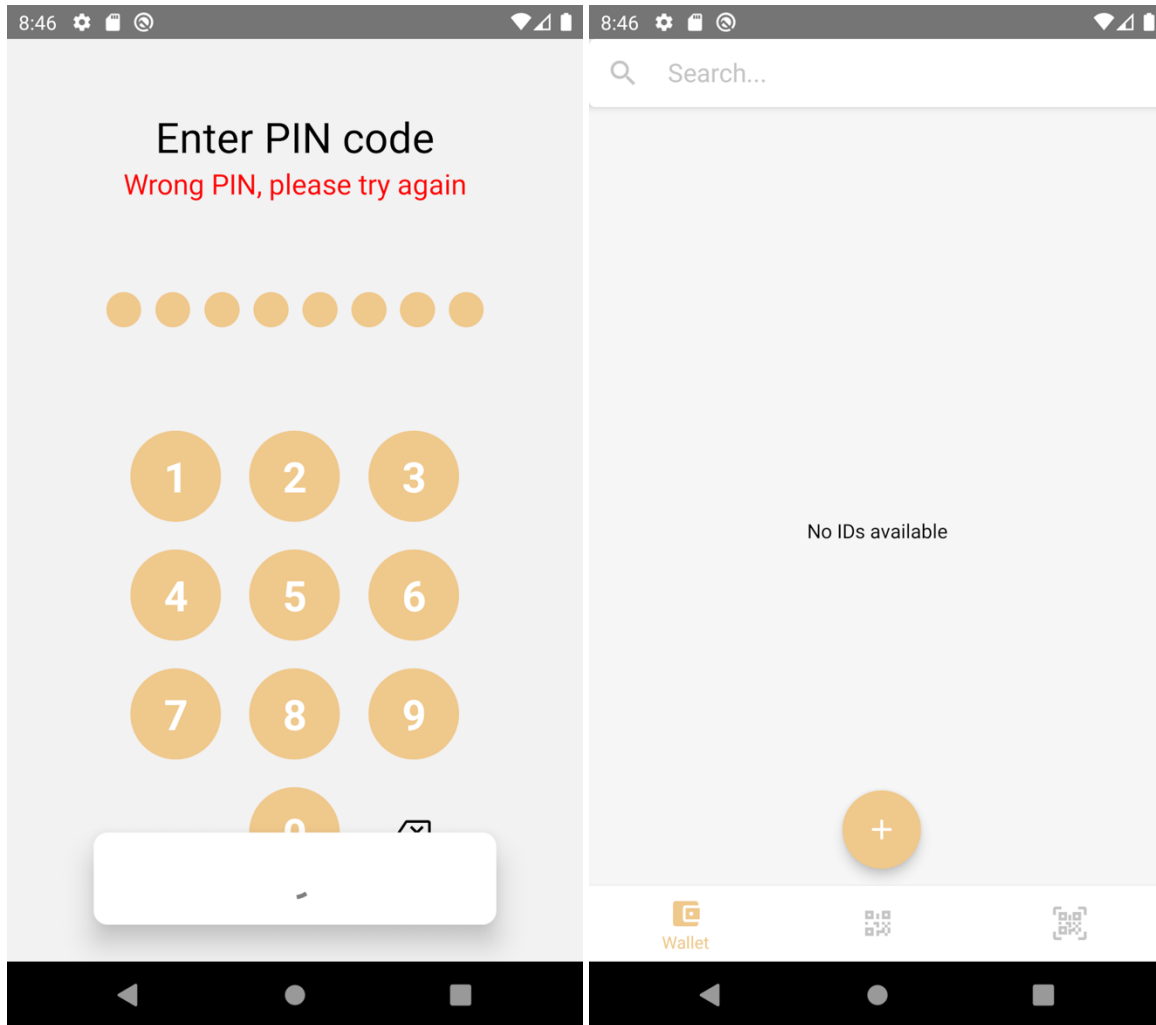
**Figure 4.6: The login screen**

Enter the wrong PIN-code which leads to an error message as the user has not been verified.



**Figure 4.7:** The screen when a wrong PIN is entered

Enter the correct PIN-code which verifies the user and sends the user to the home screen.



**Figure 4.8:** The screen when a correct PIN is entered, after a wrong attempt

### 4.3 Case 5

If the experiment needs a restart at any point remember to restart Athena. It is needed for it to be able to deliver the VC over WebSocket as there currently is a bug.

Add a user via Athena at `localhost:3000/ids/create` in a web browser. Remember the `personalId` and password.

**ID Issuer** Administer ID Get credentials

### Add ID

Personal ID \*  
1111111111

Success

Password \*  
.....

Success

Forename \*  
John

Success

Surname \*  
Smith

Success

Expiration date \*  
05/15/2024

Add subjects expiration date

**ADD ID**

**Figure 4.9: The issuer's registration schema for creatin a new user**

**ID Issuer** Administer ID Get credentials

Search

COLUMNS FILTERS DENSITY EXPORT

| <input type="checkbox"/> | ID | Personal ID | Forename | Surname | Expiration date           |
|--------------------------|----|-------------|----------|---------|---------------------------|
| <input type="checkbox"/> | 20 | 1111111111  | John     | Smith   | 2024-05-15T12:12:36+00:00 |

Rows per page: 5 1-1 of 1

EDIT SELECTED DELETE SELECTED

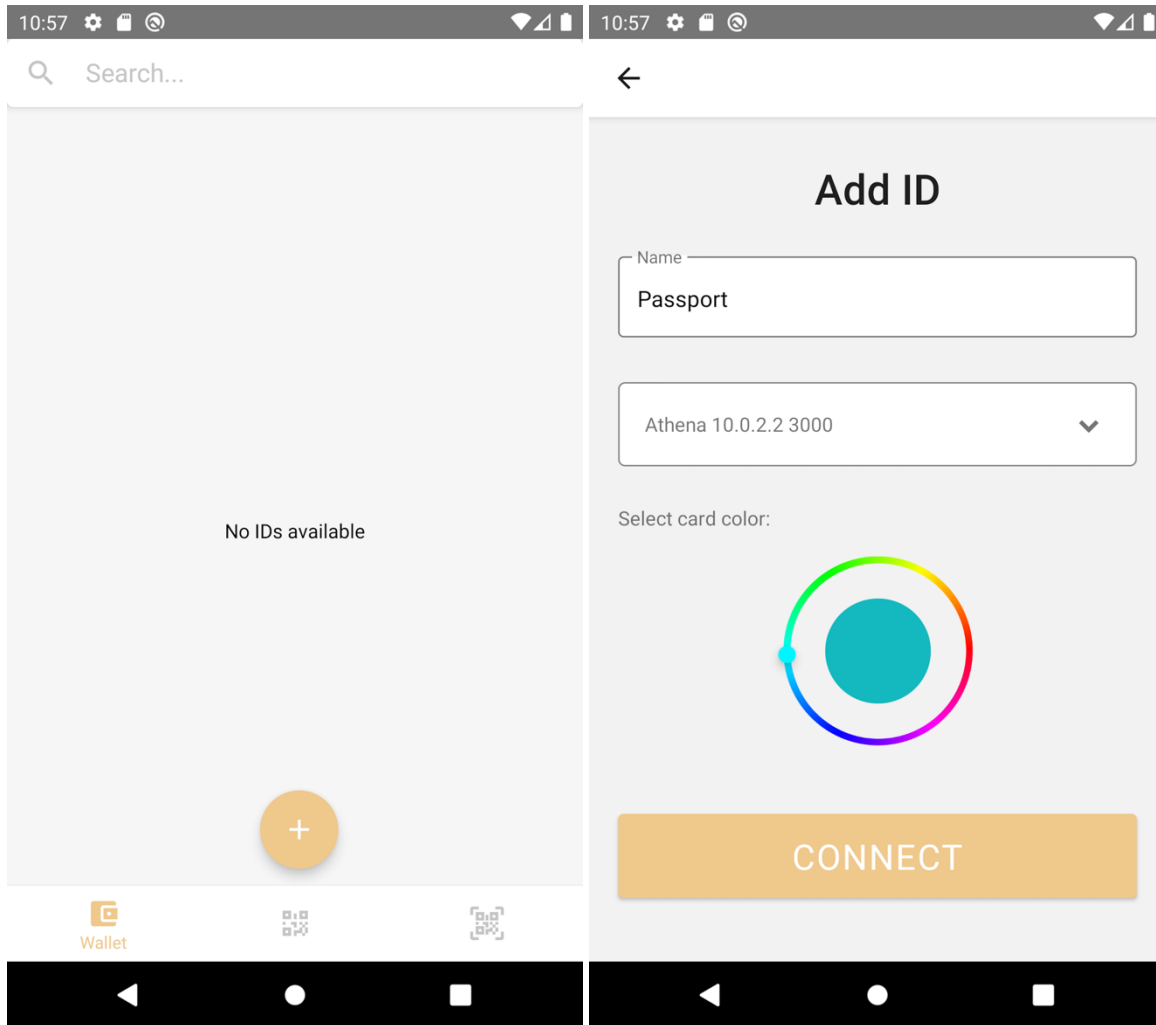
**Figure 4.10: The issuer's overview of registered users**

Check the documentDIDs column in the user table in Supabase for the user to be used for testing.

| forename varchar | surname varchar | expirati... timest...  | documentDIDs varchar[] |
|------------------|-----------------|------------------------|------------------------|
| John             | Smith           | 2024-05-15 12:12:36... | []                     |

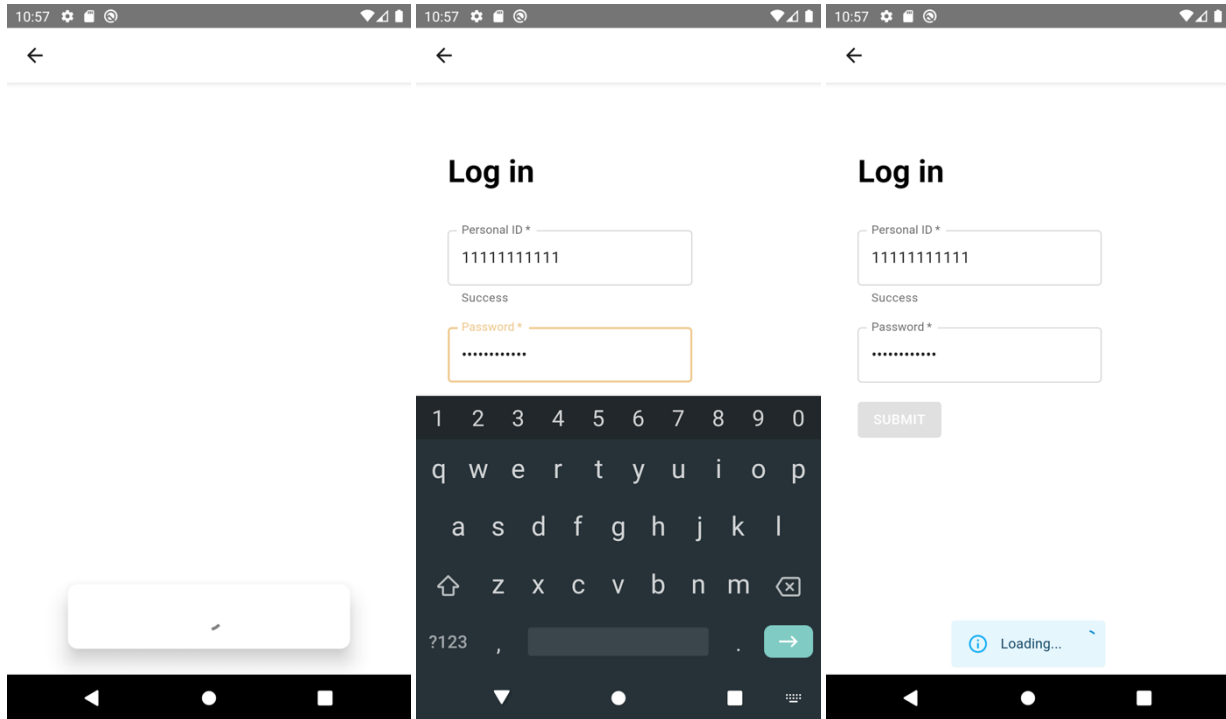
**Figure 4.11: A row from the user table in Supabase**

Run an instance of Aphrodite and log in as in case 2 or create a wallet as in case 1. Thereafter, press the “+” button on the home screen. An Add ID form appears, select a name and the ID provider Athena 10.0.2.2 3000 and press the “CONNECT” button.



**Figure 4.12: The start of the process of adding an ID**

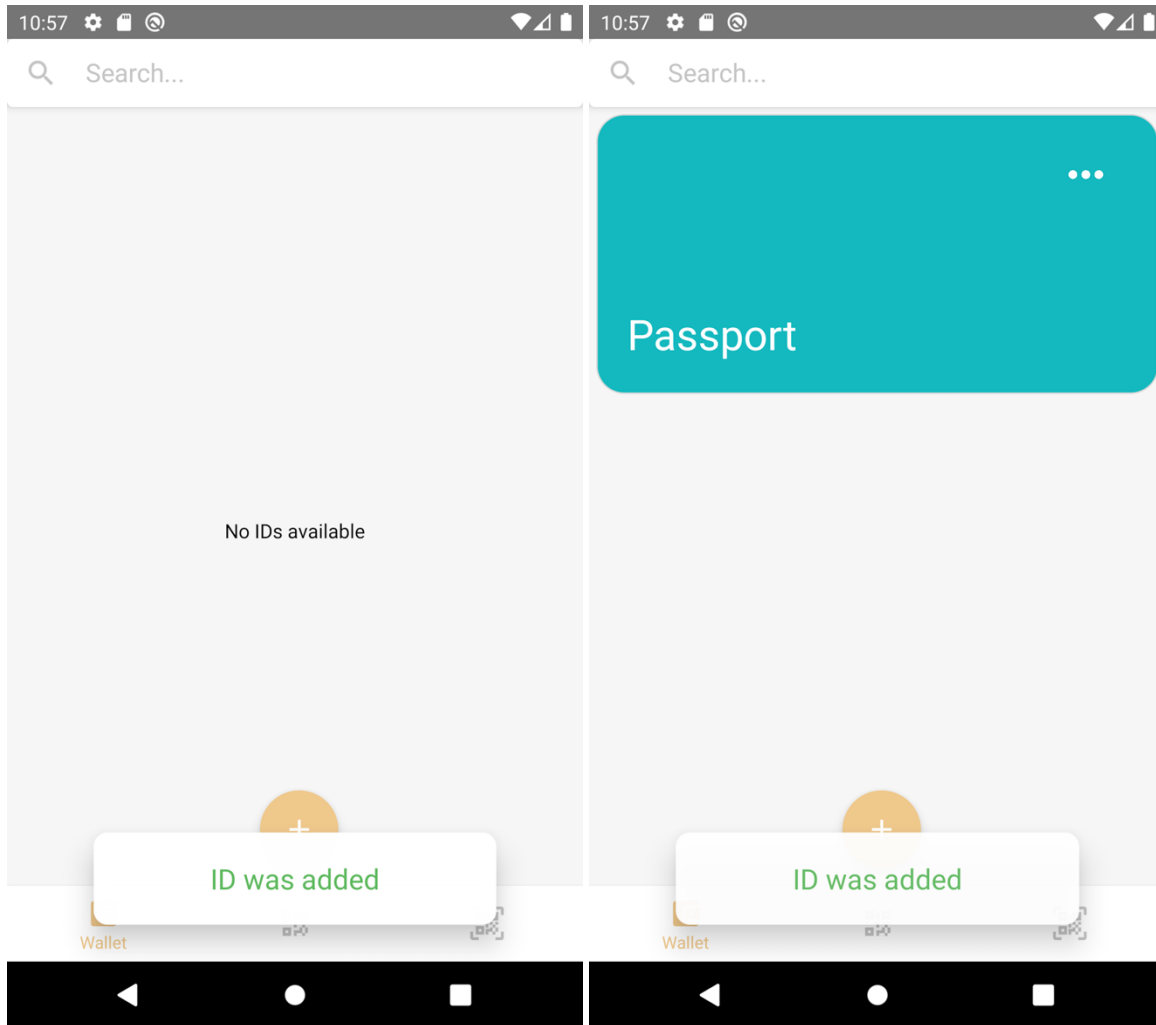
Enter the login credentials used in Athena in embedded login screen (personalId and password) and press the "SUBMIT" button.



**Figure 4.13: The process of connecting and sending the user's credentials to the selected issuer**

Wait for the success message. The user is then redirected to the home screen where the new ID appears.





**Figure 4.14: The screen when the ID has been sent from the issuer to the application**

Reload the Supabase table editor and check the user table and see that a new value has been appended to the documentDIDs column for the user being tested. Copy the value by right clicking the value and clicking “Copy cell content”. The value should be at the documentDIDs column and the corresponding user’s row.

| forename varchar | surname varchar | expirati... timest...  | documentDIDs varchar[]            |
|------------------|-----------------|------------------------|-----------------------------------|
| John             | Smith           | 2024-05-15 12:12:36... | ["did:key:z6MkicHA1YQL4VmQLUG3... |

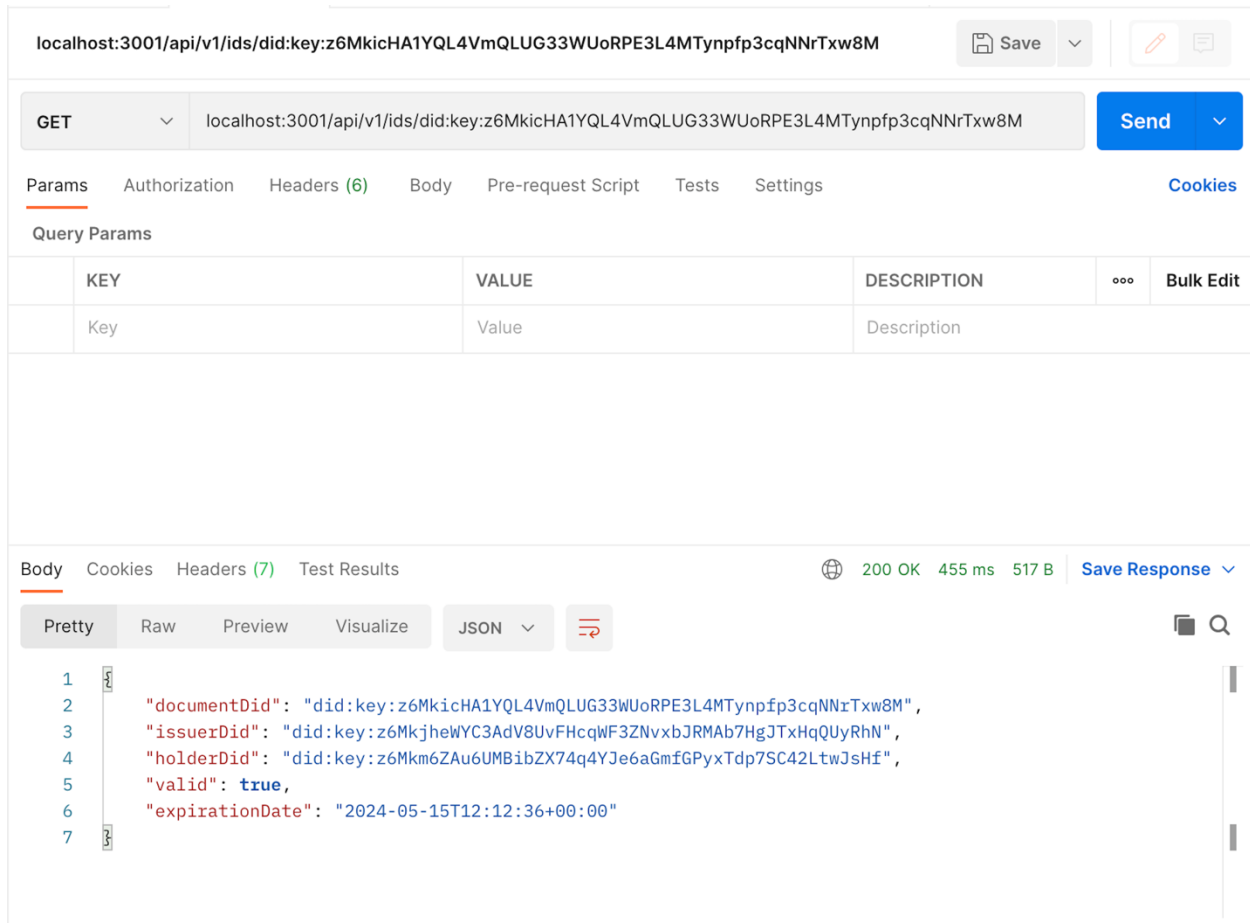
**Figure 4.15: A row from the user table in Supabase, containing data such as the document DIDs linked to the user**

| forename varchar | surname varchar | expirati... timest...  | documentDIDs varchar[]             |
|------------------|-----------------|------------------------|------------------------------------|
| John             | Smith           | 2024-05-15 12:12:36... | ["did:key:z6MkicHA1YQL4VmQLUG3...] |

- Copy cell content
- Edit row
- Delete row

**Figure 4.16: A row from the users table in Supabase, with the option to copy the documentDIDs column**

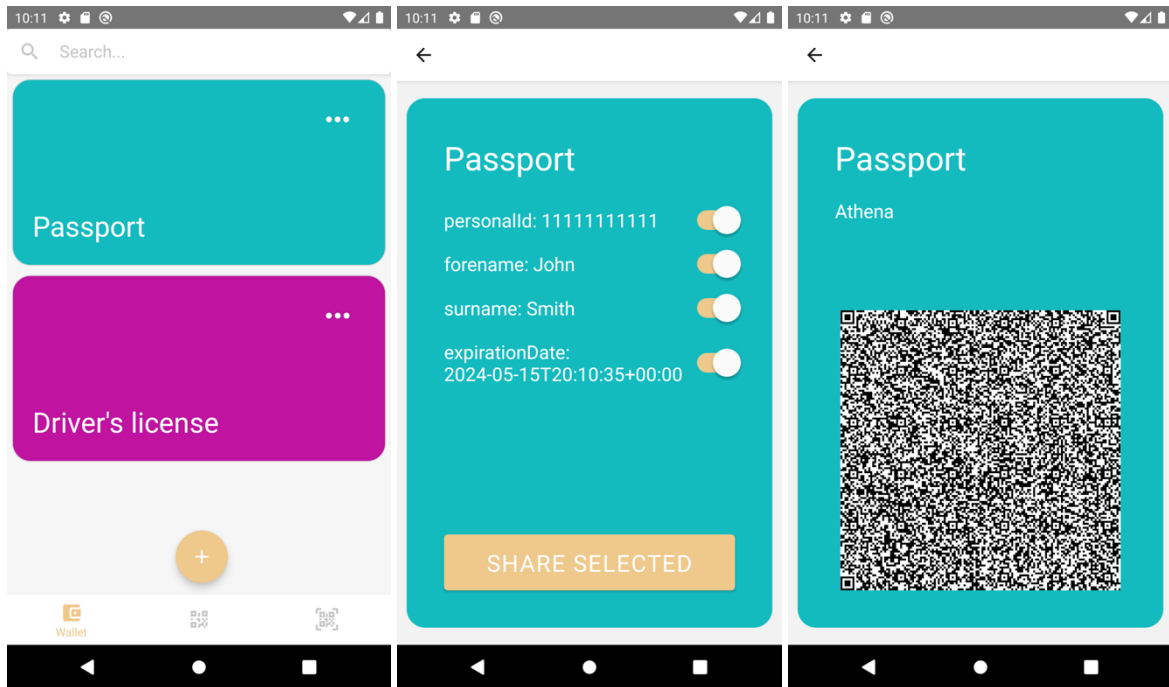
Check that the agreement exists in Kratos. This can be accomplished by using e.g., Postman to send a GET REST call to `/api/v1/ids/{document-did}`, where `document-did` is the DID of the newly created ID card. Use the written down value from Supabase as document DID. A response with the same `documentDid`, and other variables such as `“valid”`, `“issuerDid”`, `“expirationDate”` and `“holderDid”` is returned.



**Figure 4.17: A REST call for viewing the data for the newly created agreement**

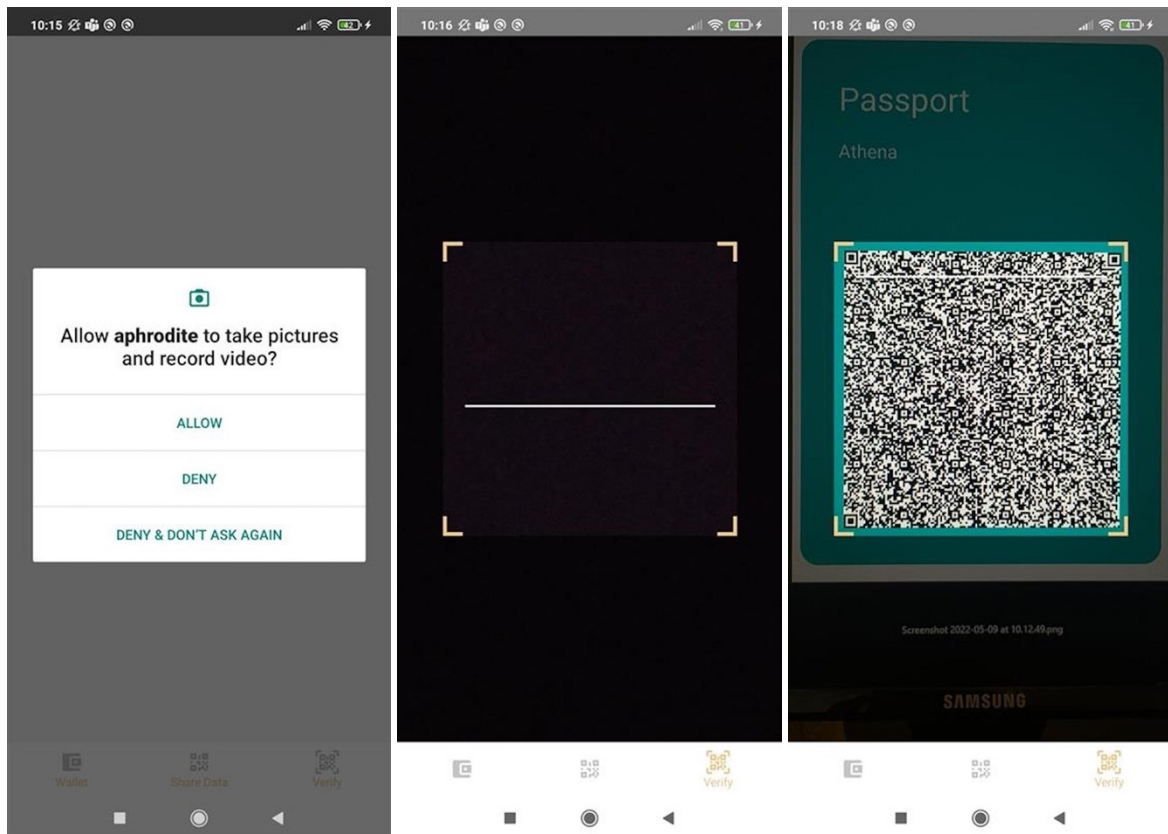
#### 4.4 Case 6/10

Run Iris and two instances of Aphrodite, one should be on a mobile device (verifier). The other can be run on an emulator (ID holder). The ID holder logs in and presses on an ID card on the home screen, then selects all attributes and presses the share button. A QR code will then appear onscreen.



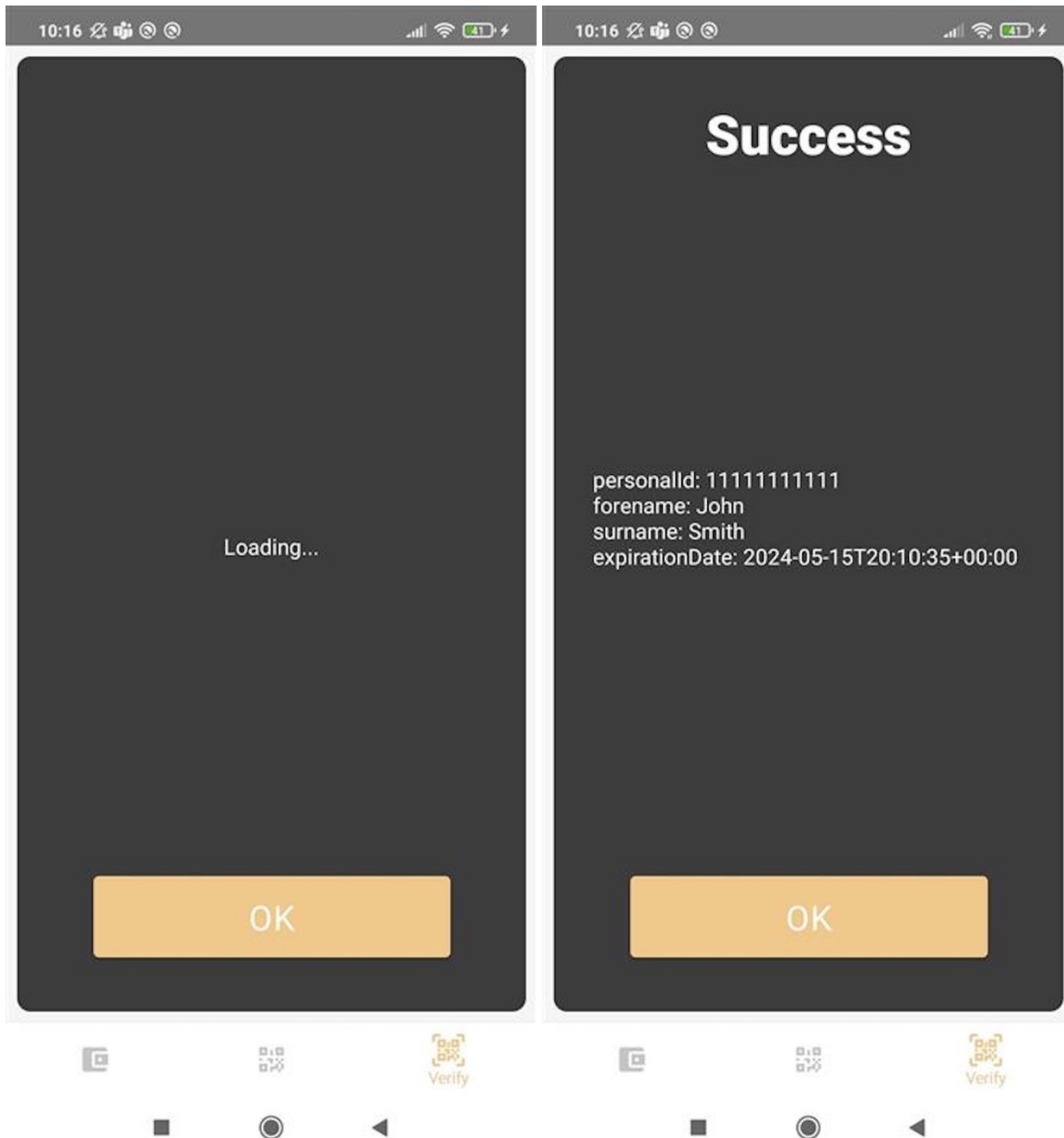
**Figure 4.18: The process of generating a QR-code for physical verification**

The verifier opens the scanner tool via the menu bar (option 3 from the left) and allows access to the camera. Then scans the user's QR-code with the scanner tool and makes sure the QR-code fits inside the square while keeping it there.



**Figure 4.19: The process off scanning a QR-code for verification**

A pop up will appear with the status of the verification process (success, revoked, invalid and more). If successful, the details will be displayed.



**Figure 4.20: The screen displaying the verification status and the data sent in the ID**

#### 4.5 Case 8

Run Iris and Aphrodite. Then log into the mobile application as in case 2. Add an ID and remember the document DID being stored in Supabase, as in case 5. Check that the agreement exists in Kratos. This can be accomplished by using e.g., Postman to send a GET REST call to `/api/v1/ids/{document-did}`, where document-did is the newly created ID card's DID. The "valid" variable of the newly created ID card should be "true".

localhost:3001/api/v1/ids/did:key:z6MkrN2xXyxGZzd5hPjZYJGKhCiWYqSUNHCU2i2vNouH2d69

GET localhost:3001/api/v1/ids/did:key:z6MkrN2xXyxGZzd5hPjZYJGKhCiWYqSUNHCU2i2vNouH2d69

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description |     |           |

Body Cookies Headers (7) Test Results 200 OK 702 ms 517 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2 "documentDid": "did:key:z6MkrN2xXyxGZzd5hPjZYJGKhCiWYqSUNHCU2i2vNouH2d69",
3 "issuerDid": "did:key:z6MkjheWYC3AdV8UvFhcqWF3ZnvxbJRMab7HgJTxBhQyRhN",
4 "holderDid": "did:key:z6MktSuiHzM8PL3Tfajo3h8eNB4undMH5d8Zjy8NvaG9AAfu",
5 "valid": true,
6 "expirationDate": "2024-05-15T20:10:35+00:00"
7 }

```

Figure 4.21: A REST API call to view the existing agreement

Press on the three dots on the card correlating to the document DID and press the delete option. Thereafter press the confirm button.

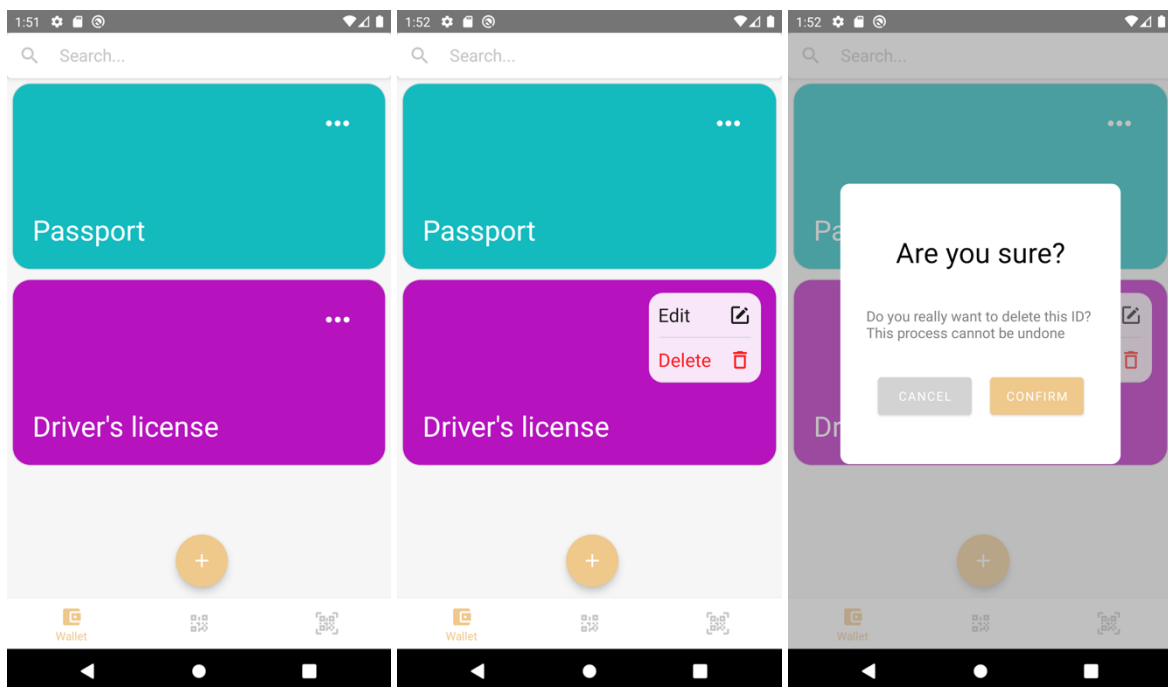


Figure 4.22: The process of deleting an ID from the mobile application

There currently is a UI bug which makes the ID card stay on screen after deletion. However, by pressing R to reload Metro and logging in to the application again, the bug is worked around. Thereafter the ID card should no longer exist on the home menu.

```
...na -- node - node /opt/local/libexec/yarn/bin/yarn.js dev e /opt/local/libexec/yarn/bin/yarn.js start --reset-cache ...cuments/programming/DATT2900/aphrodite -- -zsh ... node - node /opt/local/libexec/yarn/bin/yarn.js dev +

#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####

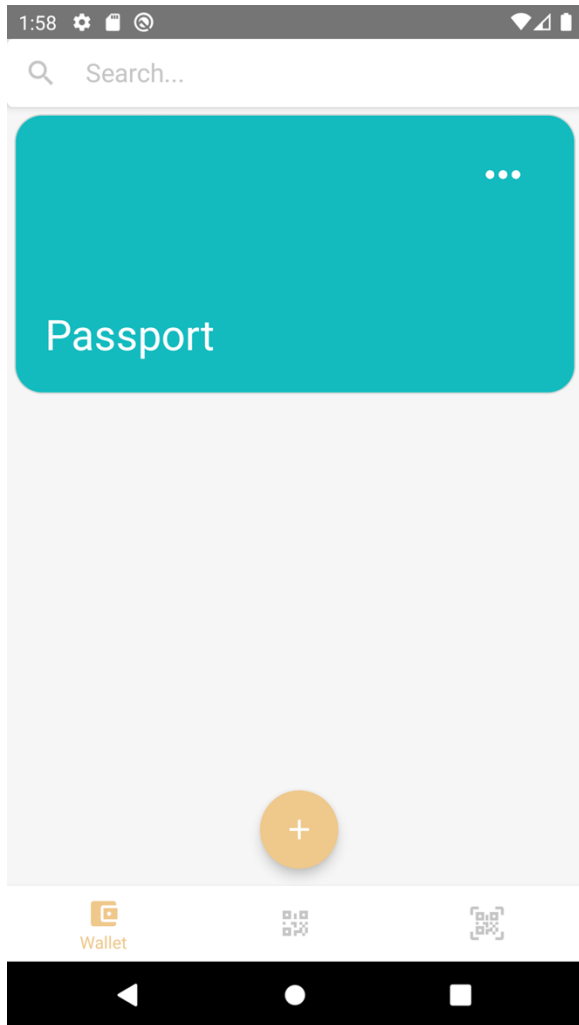
warning: the transform cache was reset.
Welcome to Metro!
Fast - Scalable - Integrated

To reload the app press "r"
To open developer menu press "d"

[BUNDLE] ./index.js
LOG WARNING: This environment is missing a secure random source; generated private keys may be at risk, think VERY carefully about not adding a better secure source.
LOG Shims Injected:
LOG - atob
LOG - btoa
LOG - nextTick
LOG - crypto.getRandomValues
LOG - FileReader.prototype.readAsArrayBuffer
WARN Require cycle: node_modules/react-native-crypto/index.js -> node_modules/react-native-randombytes/index.js -> node_modules/sjcl/sjcl.js -> node_modules/react-native-crypto/index.js
Require cycles are allowed, but can result in uninitialized values. Consider refactoring to remove the need for a cycle.
WARN Require cycle: node_modules/stream-browserify/index.js -> node_modules/stream-browserify/node_modules/readable-stream/readable.js -> node_modules/stream-browserify/index.js
Require cycles are allowed, but can result in uninitialized values. Consider refactoring to remove the need for a cycle.
LOG Running "aphrodite" with {"rootTag":!}
Info Reloading app...
[BUNDLE] ./index.js
LOG WARNING: This environment is missing a secure random source; generated private keys may be at risk, think VERY carefully about not adding a better secure source.
LOG Shims Injected:
LOG - atob
LOG - btoa
LOG - nextTick
LOG - crypto.getRandomValues
LOG - FileReader.prototype.readAsArrayBuffer
WARN Require cycle: node_modules/react-native-crypto/index.js -> node_modules/react-native-randombytes/index.js -> node_modules/sjcl/sjcl.js -> node_modules/react-native-crypto/index.js
Require cycles are allowed, but can result in uninitialized values. Consider refactoring to remove the need for a cycle.
WARN Require cycle: node_modules/stream-browserify/index.js -> node_modules/stream-browserify/node_modules/readable-stream/readable.js -> node_modules/stream-browserify/index.js
Require cycles are allowed, but can result in uninitialized values. Consider refactoring to remove the need for a cycle.
LOG Running "aphrodite" with {"rootTag":!}

```

Figure 4.23: Metro, giving the option to restart the application



**Figure 4.24: The home screen without the previously deleted ID**

Check that the agreement is revoked in Kratos. This can be accomplished by using e.g., Postman to send a GET REST call to `localhost:3001/api/v1/ids/{document-did}`, where `document-did` is the newly created ID card's DID. The “valid” variable should be “false” now.

The screenshot shows a REST client interface with the following details:

- URL:** localhost:3001/api/v1/ids/did:key:z6MkrN2xXyxGZzd5hPjZYJGKhCiWYqSUNHCU2i2vNouH2d69
- Method:** GET
- Response Status:** 200 OK, 4.38 s, 518 B
- Response Body (JSON):**

```

1 {
2 "documentDid": "did:key:z6MkrN2xXyxGZzd5hPjZYJGKhCiWYqSUNHCU2i2vNouH2d69",
3 "issuerDid": "did:key:z6MkjheWYC3Adv8UvFHcqwF3ZNVxbJRMab7HgJTxHqQUyRhN",
4 "holderDid": "did:key:z6MktSuiHzM8PL3Tfajo3h8eNB4unDMH5d8Zjy8NvaG9AAfu",
5 "valid": false,
6 "expirationDate": "2024-05-15T20:10:35+00:00"
7 }

```

**Figure 4.25:** A REST API call to view the same agreement after it has been revoked

## 4.6 Case 11

Run Iris and Aphrodite. Then log into Aphrodite. Press the “+” button on the home screen. Look at the ID providers drop down list and take note of the available providers. Thereafter, go back to the home screen by pressing the left pointing arrow in the upper left corner of the device.



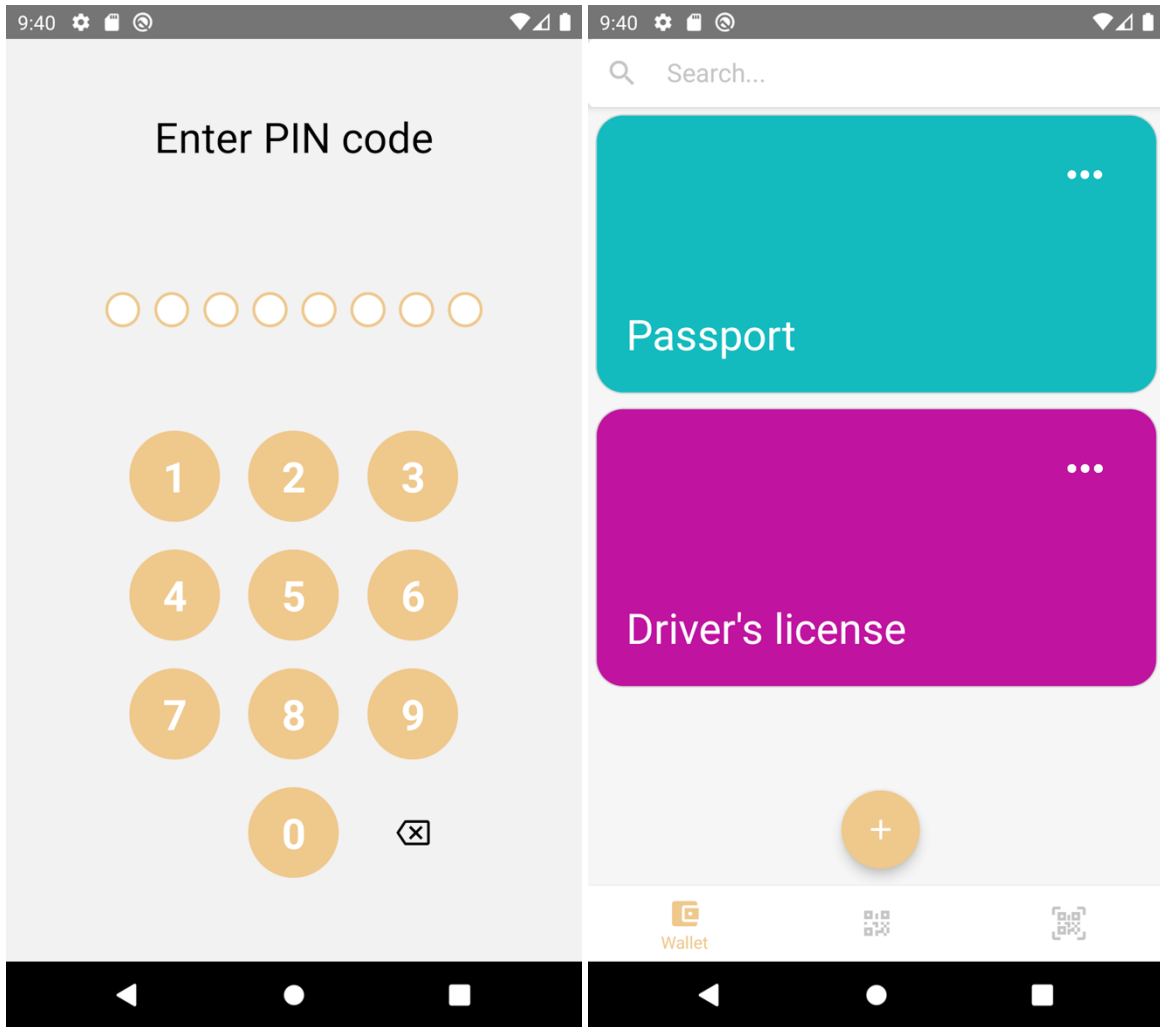
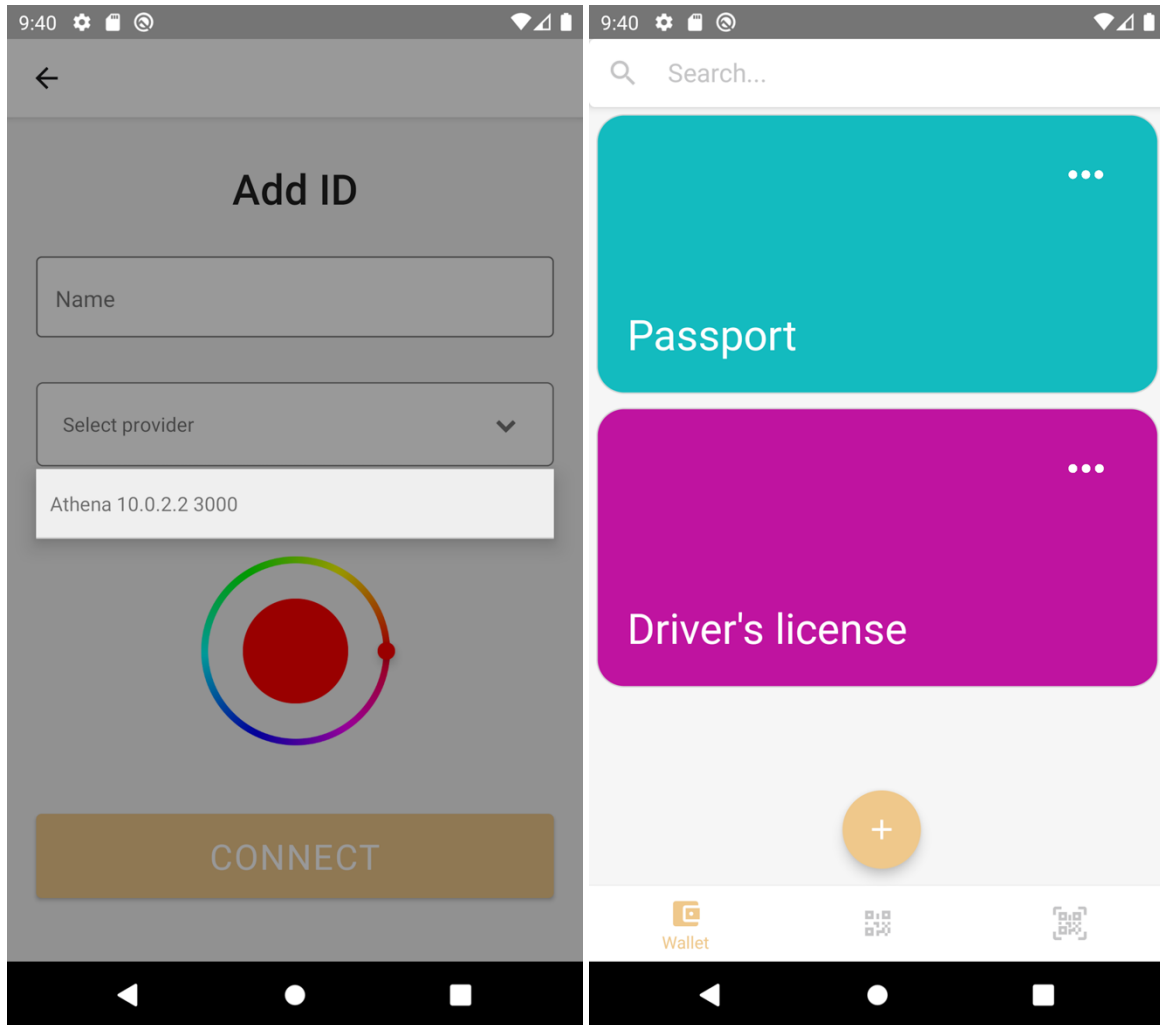


Figure 4.26: The process of logging in to a user that has already added some IDs



**Figure 4.27: The process of exiting from the add ID screen to the home screen**

Run a GET REST call to `/api/v1/key`, with BasicAuth using the ADMIN password and username defined in the `.env` file to fetch a valid token. Make sure to copy it.

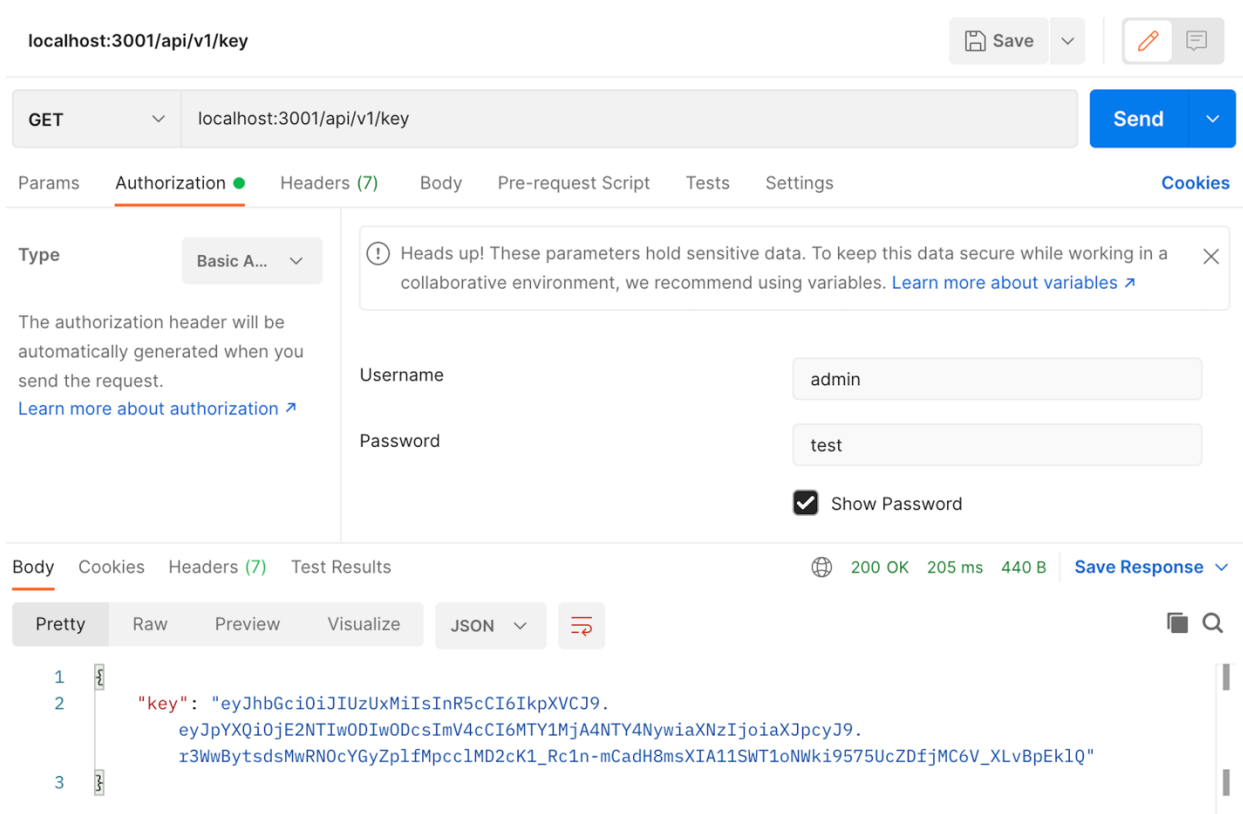
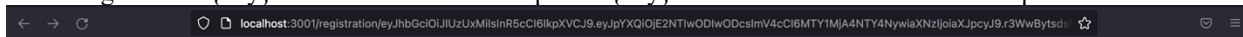


Figure 4.28: The REST API call necessary to retrieve a token for issuer registration

Go to /registration/{key} in a web browser and replace “{key}” with the token from the last step.



## Issuer Registration

Name  
Please enter issuer name

URL  
https://  
Please enter issuer URL

DID  
Please enter issuer DID

REGISTER

Figure 4.29: The issuer registration web page

Write the wished name for the issuer, it should be distinguishable from that of those already registered, to be able to prove that this one is registered when the test is complete. The URL is the URL that Athena is running on (if set up correctly it is 10.0.2.2:3000, because that is corresponding to localhost:3000 for the Android emulator). For DID use a value of format did:key:{arbitrary number} (e.g., did:key:123), make sure that no issuers are already registered with this did. This can be checked by running GET localhost:3001/api/v1/issuers. Keep in mind that the arbitrary value will hinder the issuer from being used for verification in the future, but it will suffice for this test.

The screenshot shows a REST client interface with the following components:

- URL Bar:** localhost:3001/api/v1/issuers
- Method:** GET
- Send Button:** A blue button labeled "Send".
- Params Tab:** Active tab showing "Query Params" with a table:

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description |     |           |

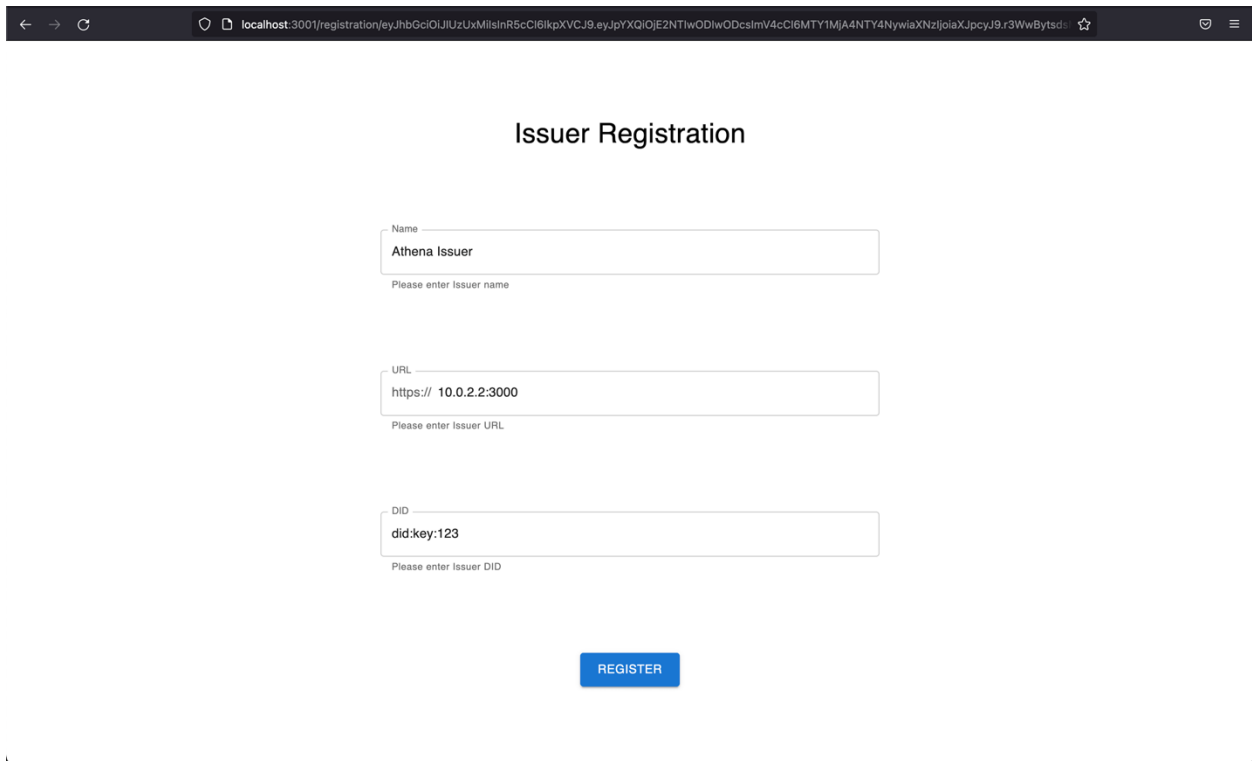
- Body Tab:** Active tab showing response details: 200 OK, 128 ms, 389 B. Includes a "Save Response" button.
- Response Format:** Pretty (selected), Raw, Preview, Visualize, JSON.
- Response Content (JSON):**

```

1 {
2 "did": "did:key:z6MkjheWYC3AdV8UvFHcqWF3ZNvxbJRMab7HgJTxHqQUyRhN",
3 "issuerName": "Athena 10.0.2.2 3000",
4 "url": "https://10.0.2.2:3000",
5 "valid": true
6 }
7
8

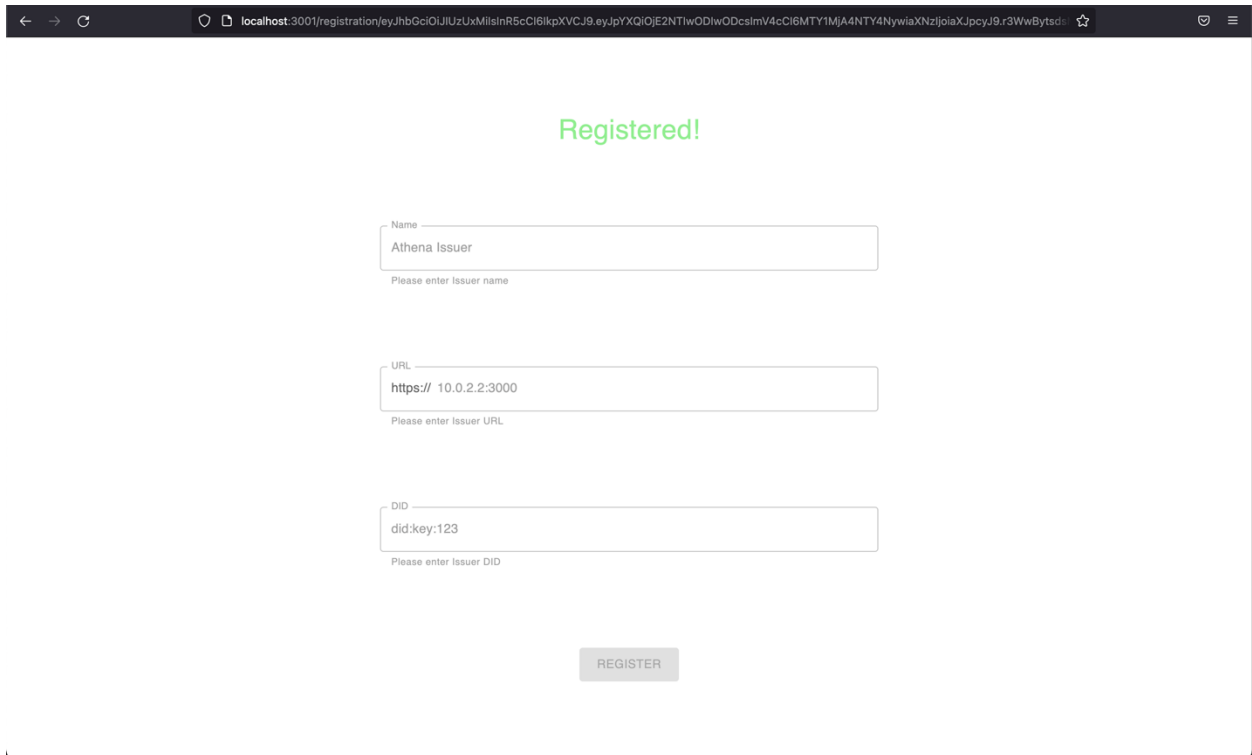
```

Figure 4.30: A REST API call for getting all registered issuers



**Figure 4.31: The issuer registration web page filled in with credentials for registration**

Press the submit button and wait for some time before the success message arrives.



**Figure 4.32: The success message after an issuer registration**

Make sure to remember the name of the issuer registered. Then press the “+” button on the home screen. The new ID provider should now appear as an option in the dropdown list for ID providers along with the previous providers.

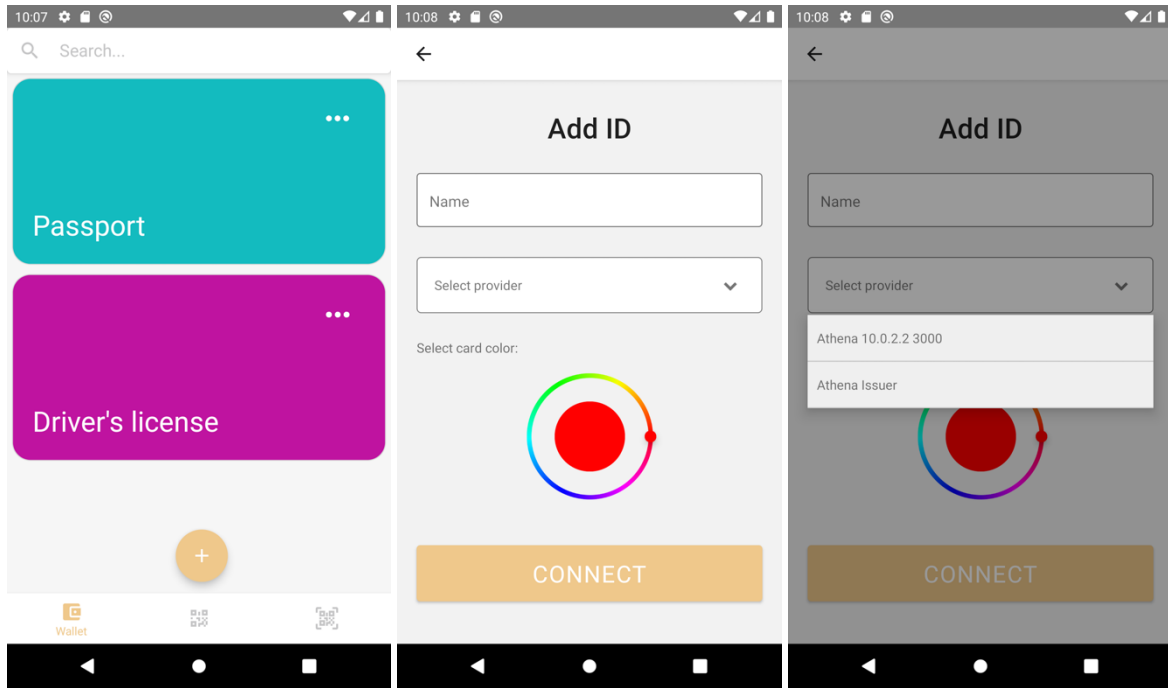


Figure 4.33: The process of viewing the available issuers, starting from the home screen

#### 4.7 Case 12

Run Athena and Iris, and two instances of Aphrodite. An ID holder on an emulator and a verifier on a mobile device. Thereafter, complete case 5, thereby checking that a user has been created in Athena, a verifiable credential has been issued to a wallet, and that an agreement between Athena and the wallet has been reached and published on the blockchain. Take note of the user's id, and all document DIDs belonging to the user which can be found in the Users table in Supabase.

| forename varchar | surname varchar | expirati... timest...  | documentDIDs varchar[]                                       |
|------------------|-----------------|------------------------|--------------------------------------------------------------|
| John             | Smith           | 2028-05-16 08:31:16... | ["did:key:z6MkpQhZESC3bDqrUoUvoRnrsd2YzEkaMLFPBYcd5BcbpRna"] |

Figure 4.34: The documentDIDs of a user being displayed by the Supabase table editor

By completing case 06/10 one can also verify that the credential is verifiable before being revoked.

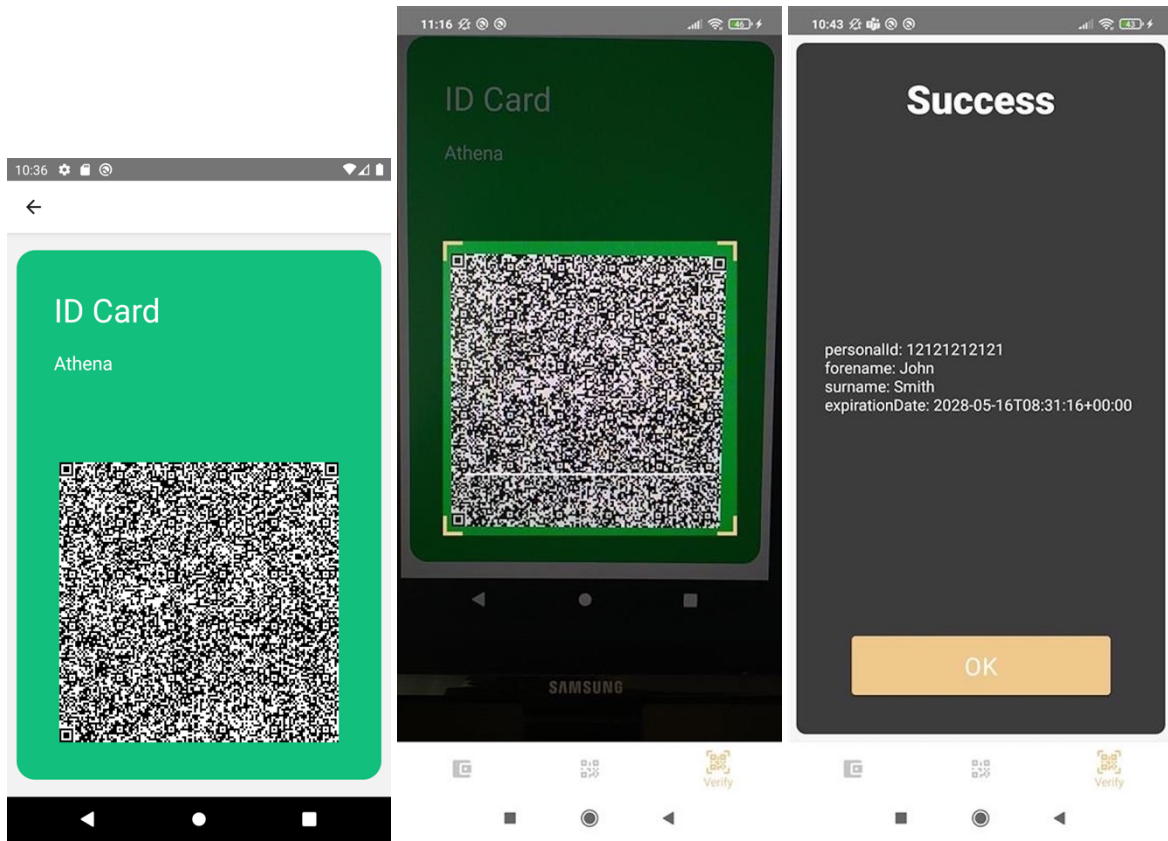
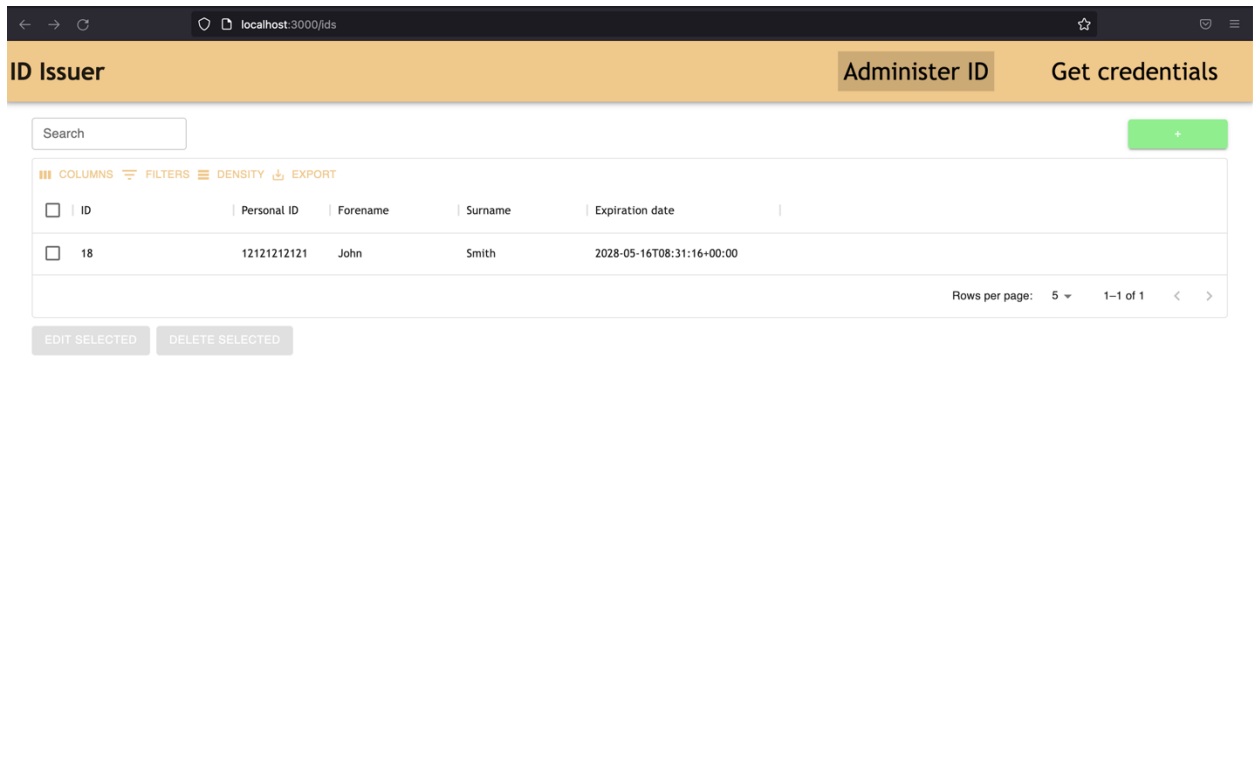


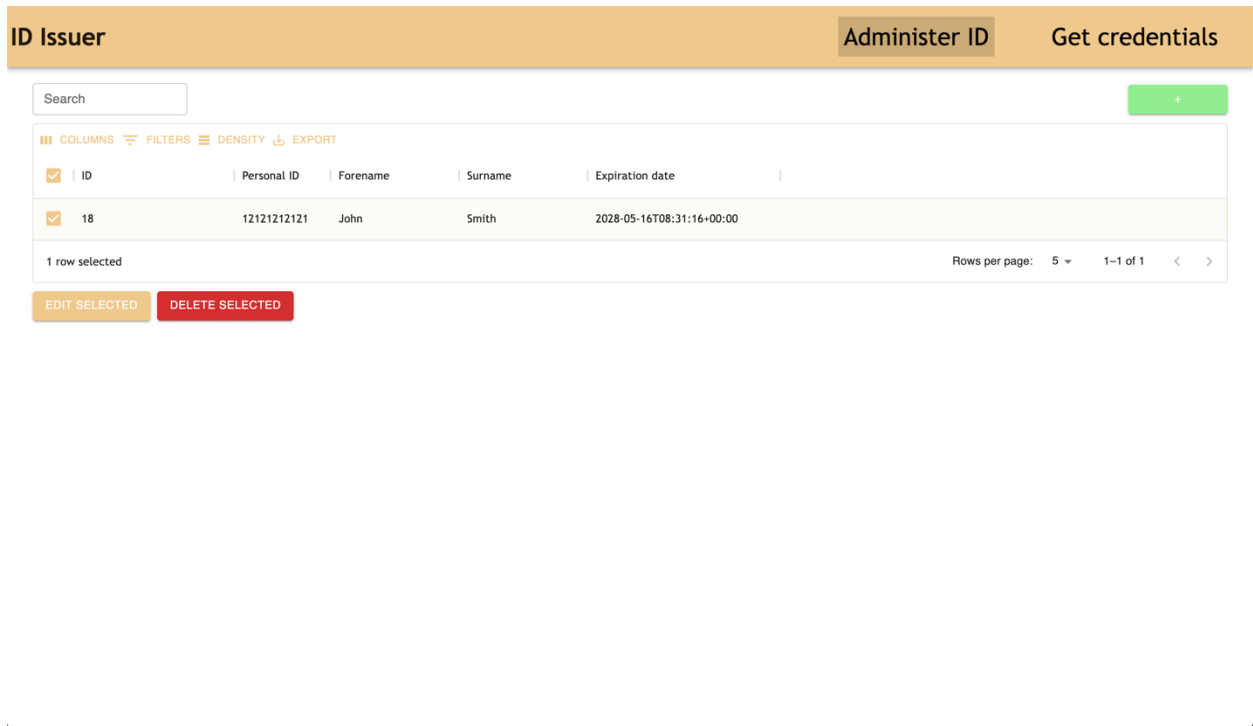
Figure 4.35: The process of physically verifying an ID through a QR-code

Go to localhost:3000/ids in a web browser.

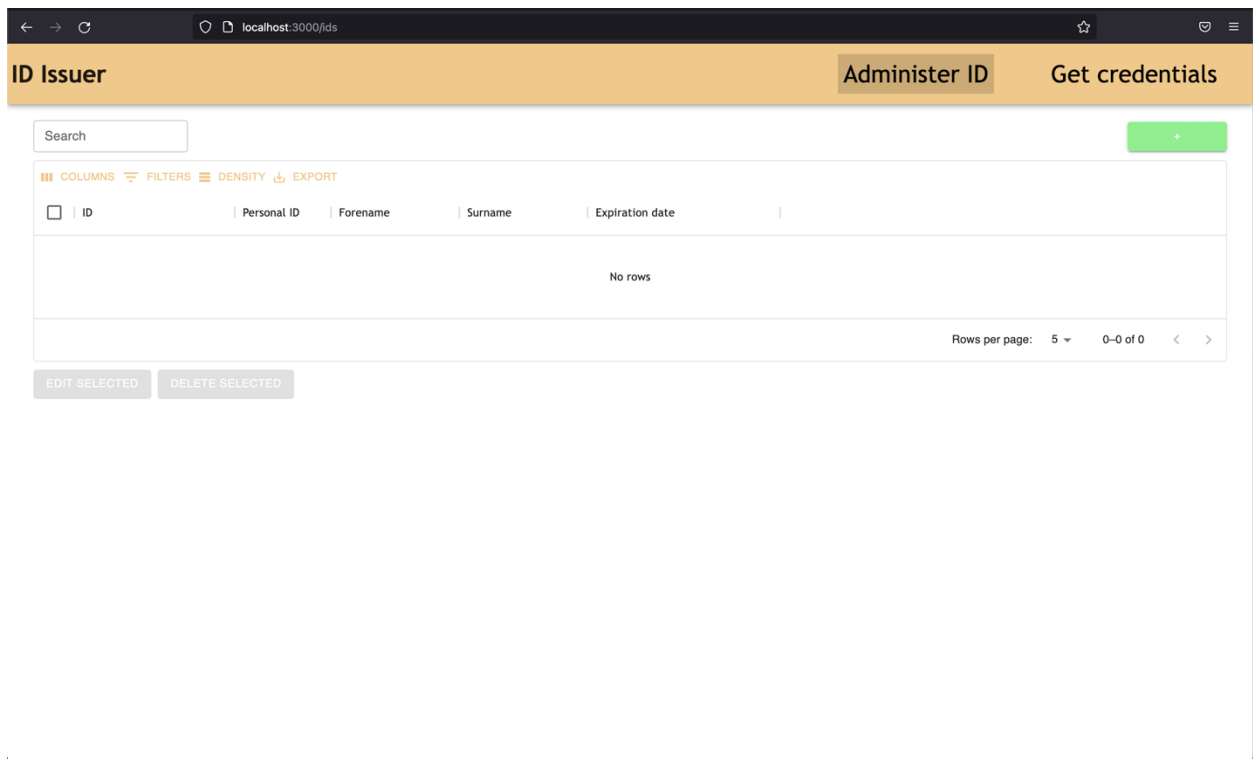


**Figure 4.36: The issuer's overview of users**

Thereafter hit the check mark for the created user and press the delete button.



**Figure 4.37: The issuer's overview of users, with one user being selected**



**Figure 4.38: The issuer's overview of users, with no users left**



Check that the user no longer exists in Supabase.

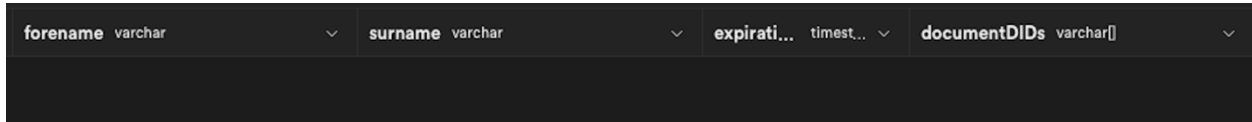


Figure 4.39: The Supabase table editor for the user table, with no users left

There is currently a bug in the code that makes Iris return immediately after calling deleting, meaning that the process for revoking the agreements may run for a while afterwards. However, waiting a minute or two should suffice. For each document DID, check that the corresponding agreements have been revoked. This can be accomplished by using e.g., Postman to send a GET REST call to `/api/v1/ids/{document-did}` for each document DID, where document-did is the newly created ID card's DID. The "valid" variable for each call should be "false".

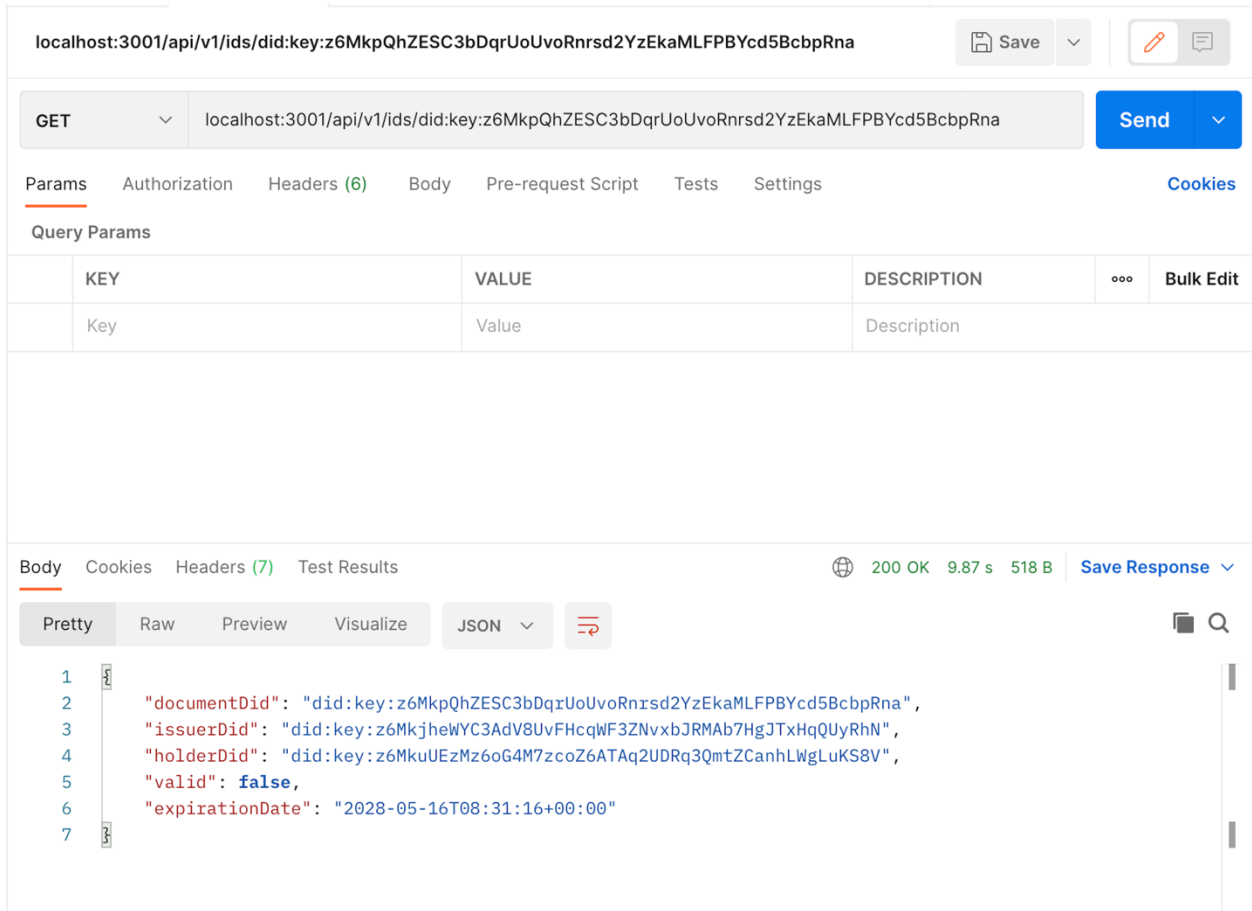


Figure 4.40: A REST API call for retrieving an agreement

Try to verify the ID card as in case 6/10. However, this will return the message "Revoked" instead of "Success" along with the data of the ID.

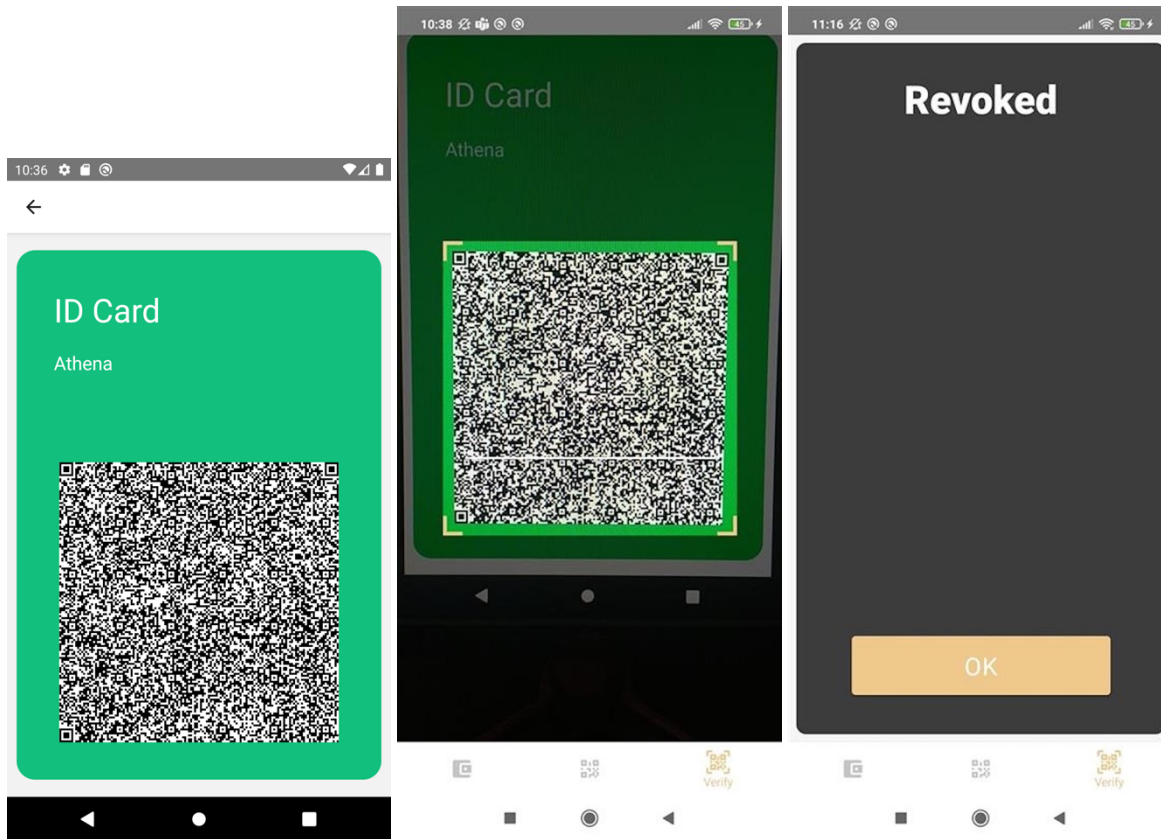


Figure 4.41: The process of physically verifying an ID through a QR-code

# **Appendix J: Kanban board description**

## **Kanban board rules**

### **Open**

Issues that are available to take.

Limit: None

### **Critical**

Issues that need to be completed. They may be critical for the operation of the application, or they may be considered bottlenecks.

Limit: 3

### **Doing**

Issues that are currently being solved.

Limit: 6

### **In review**

Issues that are currently being reviewed.

Limit: 6

### **Done**

Issues that have been completed and accepted by review.

Limit: None

### **Closed**

Issues that were aborted.

Limit: None

## **Appendix K: Source code**

The source code containing the four repositories, including environment files, is found in a folder called "IDATT2900".

