

# CSE 424

# INTRODUCTION TO BLOCKCHAIN

# PROJECT 1

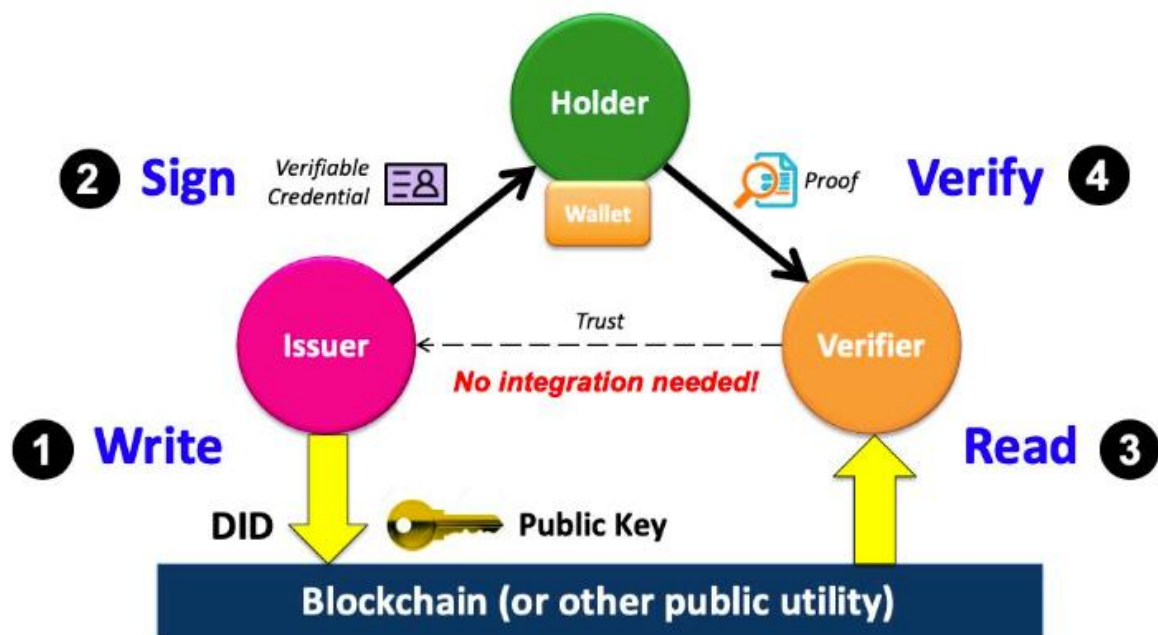
## Contents

Introduction to Self-Sovereign Identity (SSI).....	3
Key Components of SSI: .....	3
Operational Flow in SSI:.....	3
SSI Applications .....	4
E-Petition .....	5
Self-Sovereign Identity Based E-petition .....	5
Consensus Algorithm.....	6
Specific Issues in Centralized E-Petition Systems:.....	6
Limitations of Centralized E-Petition Platforms and Decentralized E-Petition Systems .....	7
Change.org .....	7
Avaaz.....	7
UK Government and Parliament Petitions.....	7
Limitations of Centralised E-petition Platforms.....	8
Censorship Risk .....	8
Data Security Concerns .....	8
Access Restrictions .....	8
System Failures.....	9
Alternatives to Centralised e-Petition Platforms: Decentralised e-Petition Systems..	9
Censorship Resistant.....	9
High Data Security .....	9
Access Cannot Be Blocked .....	9
Self-Sovereign Identity (SSI) and Smart Wallet .....	9
A digital identity wallet based on self-sovereign identity (SSI) principles.....	10
Introduction to Uport .....	10
UPort Technical Overview .....	11
Introduction to The European Digital Identity Wallet .....	11
Prototype .....	12
Digital Identity Class.....	12
EPetition Class.....	13

<b>Create Account Function .....</b>	<b>14</b>
<b>Login Function .....</b>	<b>14</b>
<b>Main Menu Function .....</b>	<b>15</b>
<b>Petition Menu Function .....</b>	<b>15</b>
<b>Sub Menu Function.....</b>	<b>16</b>
<b>Main Function .....</b>	<b>17</b>
<b>Performance of Prototype.....</b>	<b>18</b>
<b>Main Menu .....</b>	<b>18</b>
<b>Entering the informations .....</b>	<b>18</b>
<b>Petition Menu .....</b>	<b>18</b>
<b>Petition Creation Process .....</b>	<b>19</b>
<b>Viewing Petition We Created.....</b>	<b>19</b>
<b>Signing Petition Succesfully .....</b>	<b>19</b>
<b>Trying to Sign Same Petition Again.....</b>	<b>20</b>
<b>Viewing Petition's ID.....</b>	<b>20</b>
<b>Exiting the System.....</b>	<b>20</b>

# Introduction to Self-Sovereign Identity (SSI)

Self-Sovereign Identity (SSI) is a digital identity paradigm that places individuals at the center of managing their personal data. Contrary to traditional identity systems where third parties like governments, corporations, and other entities control and manage personal data, SSI empowers individuals to own, control, and share their personal information at their discretion without the need for intermediaries. This model leverages decentralized technologies such as blockchain to create a secure, private, and interoperable framework for identity management. By utilizing cryptographic methods and decentralized identifiers (DIDs), SSI ensures that individuals can interact in the digital world with the same freedom and security as they do in the physical world. This approach not only enhances privacy and security but also increases the user's control over their personal information, making digital interactions more transparent and trustful.



## Key Components of SSI:

- **Decentralized Identifiers (DIDs):** These are unique identifiers associated with digital identities, enabling the identity to be verifiable and autonomous.
- **Verifiable Credentials:** Digital tokens that represent pieces of information about an individual, which can be presented as proof of identity or qualification. These credentials are issued by trusted entities and are cryptographically secure.
- **Wallets:** Digital wallets or apps store DIDs and Verifiable Credentials. These wallets enable individuals to manage their identities from their devices securely.

## Operational Flow in SSI:

1. **Issuance:** A trusted issuer, such as a government or educational institution, issues a verifiable credential to the individual's wallet.

2. **Ownership:** The individual manages and controls their credentials. They can decide when and with whom to share their data.
3. **Verification:** When required, the individual presents a verifiable credential. The verifier, such as a service provider, can use public keys and other data from a distributed ledger to verify the credential's authenticity without needing to check back with the issuer.
4. **Consent:** All transactions involving personal data are conducted with the user's consent, ensuring privacy and compliance with data protection regulations.

SSI models leverage blockchain technology for its ability to provide a decentralized, secure, and immutable ledger, enhancing trust among all parties in the identity ecosystem. This model addresses many drawbacks of traditional and federated identity systems by reducing risks of privacy breaches, eliminating silos of user data, and empowering users with control over their own identity.

## SSI Applications

Self-Sovereign Identity (SSI) using blockchain offers several promising applications across various sectors by empowering individuals to control their personal identity data without relying on centralized authorities. Here are some potential applications of SSI implemented through blockchain technology:

1. **Digital Identity Verification:** SSI can streamline identity verification processes across various domains such as banking, healthcare, and government services. It allows individuals to prove their identity online without repeatedly submitting personal data.
2. **Healthcare Data Management:** Blockchain-based SSI can transform how medical records are accessed and shared. Patients can have control over their health records, deciding which parts of their medical history to share with doctors, hospitals, or insurance companies, thus enhancing privacy and security.
3. **Educational Credentials:** Educational institutions can issue digital credentials that are verifiable and tamper-proof using SSI. Students and professionals can manage and share their diplomas and certificates with potential employers or educational institutions without intermediary verification.
4. **Employment and HR Processes:** In the employment sector, SSI can be used to verify the authenticity of applicant qualifications and employment history. This reduces the risk of fraudulent claims and speeds up the hiring process.
5. **Financial Services:** SSI can simplify and secure processes for opening bank accounts, securing loans, and other financial services. Customers can provide verified identity data securely and swiftly without repetitive KYC checks.
6. **Travel and Border Control:** Travelers can use SSI to manage and share their travel credentials, reducing the need for physical documents and speeding up the verification process at border controls.
7. **Voting Systems:** Blockchain-based SSI can be employed to create secure and transparent digital voting systems where voters can authenticate their identities securely and vote from any location, ensuring the integrity of the electoral process.
8. **Government Services:** Governments can use SSI for various services, such as tax collection, social security, and licensing. This can streamline processes, reduce fraud, and improve citizen access to services.

# E-Petition

An e-petition in blockchain refers to a digital petition system that utilizes blockchain technology to manage and secure the process of collecting signatures for petitions.

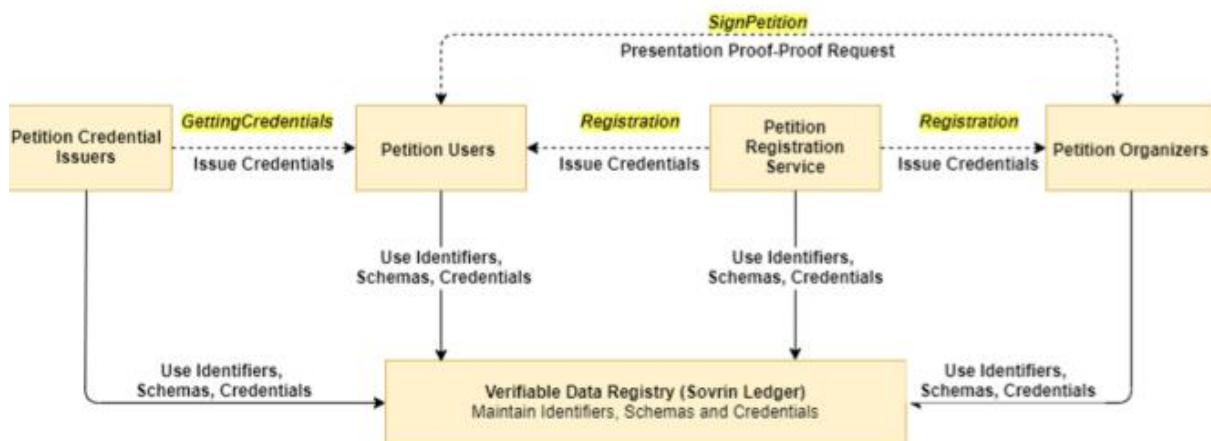
Blockchain's inherent characteristics—decentralization, immutability, and transparency—play a critical role in enhancing the integrity of e-petitions. Each petition and signature are recorded on the blockchain, creating a permanent and tamper-proof record. This ensures that once a signature is added, it cannot be altered or deleted, providing a robust level of security against fraud.

## Self-Sovereign Identity Based E-petition

Electronic signature systems based on Self-Sovereign Identity (SSI) represent an approach to digital identity management and petition processes through decentralized technologies. This type of system allows individuals to control their personal data and share their identifying information securely and selectively. In the context of electronic signatures, this means that petition signers can verify their identity independently of the central government, which has the advantage of privacy and security.



SSI-based electronic signature systems enhance privacy by allowing users to share only essential identifying attributes. At the same time, it increases security by reducing the risk of identity theft and unauthorized access to data. More importantly, the reliability of petition signatures increases when identities are verified through encryption methods. These features enable the e-signature system to be more transparent and user oriented.

Users can manage and share their identities without the involvement of a government agency or private company because they do not rely on a centralized source to verify their identity. This brings great benefits, especially when sensitive issues are raised, and large segments of society need to be mobilized.



## Consensus Algorithm

For an E-Petitions system using Self-Sovereign Identity (SSI), the **Proof of Authority (PoA)** consensus algorithm is particularly suitable due to its inherent qualities that align well with the needs of secure and efficient digital petition signing. PoA operates under a system where trusted validators, known entities like governmental organizations or NGOs, are responsible for creating new blocks and validating transactions. This system ensures that transactions are processed swiftly and securely, which is crucial for maintaining the integrity and promptness required in petition processes. Unlike Proof of Work, PoA does not require significant computational power, leading to lower operational costs and minimal energy consumption. Moreover, the scalability of PoA supports high transaction volumes, accommodating a large number of signatures, especially during critical voting or petition periods. The controlled environment of PoA, while somewhat centralized, offers transparency and accountability, crucial for legal and governmental applications where trust and verifiability are paramount. This blend of control, efficiency, and scalability makes PoA an ideal choice for blockchain-based applications that demand reliability and quick transaction confirmation, such as an E-Petitions system.

PROOF OF AUTHORITY	VS. PROOF OF WORK
	
Validators are curated by their reputation	Miners compete with each other
Less computing power is needed	More computing power is needed
Centrally managed and less automated	Decentralized design and more automated

## Specific Issues in Centralized E-Petition Systems:

### 1. Specific Issues in Centralized E-Petition Systems:

- **Signature Verification:** There is no straightforward method for petition signers or observers to verify the authenticity or count of signatures independently. This opacity can lead to suspicions of tampering or misrepresentation of public support.
- **Process Transparency:** Centralized platforms often do not disclose their operational procedures, including how signatures are collected, stored, and counted, leading to potential mistrust among users.

### 2. Data Security Risks:

- **Vulnerability to Attacks:** Centralized databases are prime targets for cyber-attacks. A successful breach can lead to the theft, alteration, or deletion of sensitive data, undermining the integrity of the petition process.
- **Single Point of Failure:** The centralized nature of traditional e-petition platforms means that the entire system's functionality is dependent on a single

network or server. Any technical failure or sabotage can render the system inoperable, potentially halting important public initiatives.

### 3. Data Integrity:

- **Tampering Risks:** With centralized control, platform administrators or hackers with access to the system could potentially alter petition details or signature counts, either to inflate or deflate public support for an issue.
- **Auditability:** The lack of an immutable audit trail in current e-petition platforms means that verifying the historical accuracy and integrity of petition data is challenging, if not impossible.

### 4. Accessibility and Inclusivity:

- **Geographic and Technological Limitations:** Centralized platforms may be inaccessible to individuals in regions with poor internet infrastructure or where the platform is not supported. Moreover, those without advanced technological tools or skills may find it difficult to participate.
- **Privacy Concerns:** Participants in centralized systems often must surrender significant personal data, deterring privacy-conscious individuals from participating.

## Limitations of Centralized E-Petition Platforms and Decentralized E-Petition Systems

Petitioning, which is important for the good functioning of democratic processes, enables citizens to submit their demands and complaints to the competent authorities, both for civil society organisations and for the government. The petitioning process, which is traditionally carried out on paper, has started to be carried out electronically with the development of the digital world. In order to meet this need in the digital environment, centralised e-petition platforms created by public institutions and non-governmental organisations have emerged. but the centralised structure of these platforms has some important limitations. these applications are mainly as follows:

### Change.org

Change.org is one of the most popular e-petition platforms in the world. many examples of petitions and solidarity have been realised on this platform. However, the centralised nature of the platform has its drawbacks, including the security of user data and the risks of manipulation. In addition, the administrators have full authority over the petitions on the site. this leads to manipulations in the acceptance and rejection processes. users cannot feel that they are in a comfortable and fair environment in such environments.

### Avaaz

It has been possible to initiate actions on a global scale in the past with Avaaz, which shows how effective an e-petition platform it is. Similarly, like other examples, Avaaz is centralised, which

### UK Government and Parliament Petitions

The UK Government and Parliament Petition System is an official platform where individuals can submit requests to the UK Parliament or Government. The system aims to ensure public participation in the legislative process by giving citizens the opportunity to propose changes or demand action directly from the government and parliament.

Any UK resident or national can submit a petition to start a petition. The petition should set out clear and unambiguous actions to be taken by the Government or Parliament and should not address issues outside the UK Government's jurisdiction. Submitted petitions are reviewed by Petitions Committee staff to determine whether they meet the necessary criteria, including the use of appropriate language and a clear purpose. Once approved, the petition will be published on the website and can be signed by anyone from any UK postcode. He needs to collect 10,000 signatures on the petition to get a response from the government. If 100,000 signatures are collected, the bill will be debated in the Diet. Once a petition reaches 100,000 signatures, the discussion will be evaluated by the Petitions Committee, which will decide whether to discuss the petition and set an appropriate schedule.

These debates usually take place in Westminster Hall and sometimes in the House of Commons. The system emphasizes transparency by publishing the progress of each petition and the government's response. It is also designed to be usable by a wide range of people, including people with disabilities. The UK government and parliamentary petition system actively engages the public in the political process, allowing them to play a more active role in policy-making. For example, important petitions on issues such as funding for mental health services and the cost of car insurance for young drivers have led to debate in Congress. This system is an example of how digital tools can foster democratic participation and enable citizens to take a more active role in governance and policy-making.

Exposes the platform to security and data privacy risks. political and commercial manipulation can undermine the petition. this can create a sense among users that the environment is unfair and dangerous.

## **Limitations of Centralised E-petition Platforms**

With decentralised e-petition platforms, citizens can reflect their ideas, opinions and requests on digital platforms operated by public institutions or civil society organisations. Through these platforms, citizens can create petitions in electronic environment and they can feel the contribution of the petitions in their lives with the feedback they receive. However, some limitations of these platforms can be listed as follows

### **Censorship Risk**

Centralised platforms are controlled by the institution that operates the platform, so it is inevitable in intellectually underdeveloped countries that governments or institutions will censor petitions critical of them. It can lead to problems of transparency and undermine trust between the public and institutions.

### **Data Security Concerns**

In a centralised structure, all data is collected at a single point. a platform structure that is open to cyber-attacks is presented. in an environment such as petition, which is very important and privacy is required in some issues, it becomes a frightening situation that citizens' personal information is exposed as a result of data leaks.

### **Access Restrictions**

As in the case mentioned in the previous paragraphs, access to centralised platforms can be restricted in order to impose censorship. therefore, at moments of greatest public solidarity, civil society organisations or the government can prevent citizens from exercising their right to petition.



## **System Failures**

Centralised platforms may experience the collapse of the entire system when there is a technical infrastructure problem. As a result of such a thing, petition exchange becomes impossible.

## **Alternatives to Centralised e-Petition Platforms: Decentralised e-Petition Systems**

With decentralised e-petition systems developed using blockchain technology, a rationalist structure that can completely eliminate the limitations of the above-mentioned centralised platforms is needed. Blockchain is a cryptographic technology that provides a database that is reliable, transparent and difficult to modify. [...] In decentralised e-petition systems, petitions are recorded on a blockchain. This makes it difficult to modify or delete petitions. At the same time, censorship is prevented as the system is not controlled at a single point. The potential benefits of decentralised e-petition systems are as follows:

### **Censorship Resistant**

Blockchain technology makes it difficult to modify or delete petitions. This strengthens freedom of expression and accountability.

### **High Data Security**

With the decentralized database structure, when exchanging petitions, shopping is provided in a safe environment with the security provided by the high encryption feature. Thus, the hackers' job becomes very difficult and the risk of data leakage is eliminated.

### **Access Cannot Be Blocked**

Another benefit of the decentralized structure is that access to that petition cannot be blocked by the decision of a public institution or non-governmental organization since access cannot be provided from a single point. This provides an environment where freedom of thought exists.

Even though there are examples of this theoretical structure in today's life, such as smart wallets, the e-petition system remains a structure used only theoretically or among a few private organizations. That's why there is no example to show. Perhaps this is because large organizations do not want to give such power to the people and want to keep the power and limitations under their own control.

## **Self-Sovereign Identity (SSI) and Smart Wallet**

Self-sovereign identity (SSI) systems, empowered by blockchain technology, offer a decentralized approach to managing digital identities, allowing individuals to control their own identity data. This method contrasts sharply with traditional centralized identity management systems, which are often vulnerable to data breaches and lack transparency in the handling of personal data.

SSI systems utilize a set of guiding principles that ensure user control, privacy, and interoperability across different platforms and services. These principles include transparency, persistence, portability, and consent, ensuring that users have complete oversight and control of their identities without the reliance on a central authority.

A digital wallet is a key management application, which provides a user with a graphical interface to store, manage and secure digital keys. These keys can be used to sign transactions, statements, credentials, documents or claims.

A digital identity wallet enables a user to establish relationships and interact with third parties in a trusted manner. While the wallet aspect is mainly dealing with key management, storage aspects and the graphical interface, the third-party interactions are rather organised by your agent, which is a part of your wallet. Your agent handles third-party interactions in your name and acts in your interest. It is a piece of software, which helps you to stay in control of your messaging, security, health records, privacy, purchases etc.

## **A digital identity wallet based on self-sovereign identity (SSI) principles.**

Wallets can be designed in different ways, and not all of them offer the same features. Some wallets, especially cloud wallets, are hosted and controlled by a third party. For example, wallets offered by exchanges such as Coinbase or Binance are not fully controlled by the user, although they belong to the user. Although the user has authentication tools (password and second factor) for access to these wallets, there is no data sovereignty.

On the other hand, self-custody wallets are installed directly on the user's device and are completely controlled by the user. Such wallets generate private keys randomly, and these keys are known only to the user, not to a third party.

A digital identity wallet designed in accordance with the principles of Self-Sovereign Identity (SSI) provides users with data sovereignty, complete control and data portability. Such wallets provide autonomy without dependence on a third party and make it possible not only ownership, but also ownership. It provides transparency and explicit consent in information sharing, and is also vendor-independent, meaning you can export your data and transfer it to another wallet of your choice. It also creates co-identifiers for each new contact or interaction, instead of getting an identifier assigned to it.

To solve the problem we have described, we will look at blockchain-based distributed systems and how they work. We will examine two systems in detail.

## **Introduction to Uport**

Uport is a secure, easy-to-use system for self-sovereign identity, built on Ethereum. The uPort technology consists of three main components: smart contracts, developer libraries, and a mobile app.

The mobile app holds the user's keys. Ethereum smart contracts form the core of the identity and contain logic that lets the user recover their identity if their mobile device is lost. Finally, the developer libraries are how third-parties app developers would integrate support for uPort into their apps.

uPort identities can take many forms: individuals, devices, entities, or institutions. Uport identities are self-sovereign, meaning they are fully owned and controlled by the creator, and don't rely on centralized third parties for creation or validation. A core function of a uPort

identity is that it can digitally sign and verify a claim, action, or transaction - which covers a wide range of use cases.

## UPort Technical Overview

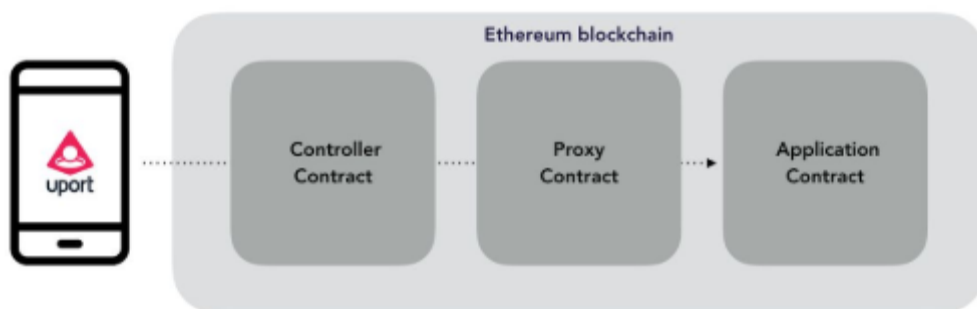
At the core of a uPort identity is the uPort identifier, a 20-byte hexadecimal string that acts as a globally unique, persistent identifier. This identifier is defined as the address of an Ethereum smart contract known as a Proxy contract. The Proxy contract can relay transactions and it is through this mechanism that the identity interacts with other smart contracts on the Ethereum blockchain.

When the user wants to interact with a particular application smart contract, they send a transaction through the Proxy contract, via a Controller contract, which contains the main access control logic. The Proxy contract then forwards this transaction to the application smart contract. This architecture allows the application to view the Proxy contract address as the interacting entity. The Proxy contract thus introduces a layer of indirection between the user's private key - stored on their mobile device - and the application smart contract.

The purpose of having a Proxy contract as the core identifier is that it allows the user to replace their private key while maintaining a persistent identifier. If the user's uPort identifier instead was the public key corresponding to their private key, they would lose control over their identifier if they were to lose the device where the private key is held.

In the case of device loss, the Controller contract maintains a list of recovery delegates that can help the uPort user recover their identity. These delegates can be individuals, such as chosen friends and family members, or institutions, like banks and credit unions. A quorum of delegates is required to let the user recover their identity and connect it to a new device.

uPort is seen as a powerful tool for managing users' identities, especially in areas such as healthcare, finance and e-government applications. While users' ability to manage their own identities and access rights increases their privacy and security, it also provides a more secure and efficient solution for service providers.



## Introduction to The European Digital Identity Wallet

The European Digital Identity Wallet is a platform designed for European Union (EU) citizens and businesses to manage their digital identities and various official documents. This system, developed by the European Commission, is acceptable and safe to use in all EU countries. Thanks to this digital wallet, users can store their documents such as ID, driver's license, education certificates in digital environment and share these documents easily and safely when necessary.

The wallet provides a high level of security and confidentiality using blockchain technology, protects users' personal information, and performs data sharing only with the explicit consent of the user. These features allow users to have full control over the data and prevent unauthorized use of personal information. Moreover, the European Digital Identity Wallet facilitates access to digital services and supports free movement within the EU, so that users can seamlessly access digital services such as e-government, e-health, and e-banking anywhere in the EU. This system aims to strengthen the EU's digital internal market and increase the digital autonomy of individuals.

There is an illustration below that shows how it works generally and it gives us a general idea for The European Digital Identity Wallet.



## Prototype

### Digital Identity Class

```
import hashlib

class DigitalIdentity:
    def __init__(self, full_name, id_number, password):
        self.full_name = full_name # Store the user's full name
        self.id_number = id_number # Store the user's identification number
        self.password = hashlib.sha256(password.encode()).hexdigest() # Hash the password for storage
        self.id = hashlib.sha256(id_number.encode()).hexdigest() # Generate a unique hash ID from the identification number
```

With this method, a digital identity is created using the `full_name`, `id_number`, and `password` parameters. It represents and stores the full name of the user named `full_name`, the identification number of the user named `id_number`, and the password of the user named `password`. The password is hashed using the SHA-256 algorithm and converted into a hash ID using the TR ID number.

# EPetition Class

```
class EPetition:
    def __init__(self):
        self.petitions = {} # Dictionary to store petitions
        self.signatures = {} # Dictionary to store signatures for each petition

    def create_petition(self, title, description, creator):
        # Create a petition and generate a unique ID based on content
        petition_content = f"{title}{description}{creator.id}".encode('utf-8')
        petition_id = hashlib.sha256(petition_content).hexdigest()
        self.petitions[petition_id] = {
            "title": title,
            "description": description,
            "creator": creator,
            "signature_count": 0
        }
        self.signatures[petition_id] = set()
        print(f"Petition successfully created: {petition_id}")

    def list_petitions(self):
        # List all available petitions
        if not self.petitions:
            print("No petitions available.")
            return None
        print("Available Petitions:")
        petition_ids = []
        for index, (petition_id, details) in enumerate(self.petitions.items(), 1):
            print(f'{index}. ID: {petition_id}, Title: {details["title"]}, Signatures: {details["signature_count"]}')
            petition_ids.append(petition_id)
        return petition_ids

    def sign_petition(self, petition_id, signer):
        # Sign a petition if not already signed by the user

        if signer.id in self.signatures[petition_id]:
            print("You have already signed this petition.")
            return
        self.signatures[petition_id].add(signer.id)
        self.petitions[petition_id]["signature_count"] += 1
        print("Petition successfully signed.")

    def get_signatures(self, petition_id):
        # Display all signature IDs for a given petition
        if petition_id not in self.petitions:
            print("Invalid petition ID.")
            return
        print("Signatures' IDs:")
        for signature_id in self.signatures[petition_id]:
            print(signature_id)
```

With the creation of this class, the basis for e-petition functions such as creating, listing, signing and obtaining e-petitions is provided. The main functions are as follows:

**\_\_init\_\_(self):** With this method, it creates an empty dictionary for e-petitions and signatures.

**create\_petition(self, title, description, creator):** With the creation of this method, the title, description and creator user information for the petition is taken and an e-petition is created. To make the e-petition unique, a unique ID is created and assigned to this e-petition. The above-mentioned information of this e-petition is stored securely and the number of signatures is kept.

**list\_petitions(self):** The main purpose of this method is to list the existing e-petitions via the menu, and the ID, title and number of signatures of each are displayed as output on the main screen.

**sign\_petition(self, petition\_id, signer):** By using this method, it is first checked whether the user trying to sign a particular e-petition has already signed it. Then the signature is made.

**get\_signatures(self, petition\_id):** This method displays all signature IDs for a given e-petition.

## Create Account Function

```
def create_account(accounts):  
    # Function to create a new account  
    full_name = input("Your Full Name: ")  
    id_number = input("Your ID Number: ")  
    password = input("Your Password: ")  
    if id_number in accounts:  
        print("An account already exists with this ID number.")  
        return None  
    new_account = DigitalIdentity(full_name, id_number, password)  
    accounts[id_number] = new_account  
    print("Account successfully created.")  
    return new_account
```

This function enables the necessary information to be obtained from the created user. then creates a new account and adds it to the dictionary named accounts. The function steps are:

- Full name, TC ID number and password information are obtained from the user.
- It is checked whether the entered identification number has already been created for another account.
- Creates a new DigitalIdentity instance and adds this account to the accounts dictionary.
- The user is notified in the main menu that the transaction is completed.

## Login Function

```
def login(accounts):  
    # Function for account login  
    id_number = input("Your ID Number: ")  
    password = input("Your Password: ")  
    if id_number in accounts and hashlib.sha256(password.encode()).hexdigest() == accounts[id_number].password:  
        print("Login successful.")  
        return accounts[id_number]  
    else:  
        print("Account not found or incorrect password.")  
        return None
```

With this function, the user's identity information is obtained to log in to the system and the accuracy of this information is checked. If the login is successful, it directs the relevant account to the main menu. If it is not successful, an error message is printed on the screen. The function steps are:

- TC ID number and password information is obtained from the user.
- It is checked whether the entered TR ID number is included in the accounts dictionary.

- If there is an account, this time the accuracy of the entered password is checked with the passwords registered in the system.
- If you log in with the correct credentials, the account will be directed to the main menu.

## Main Menu Function

```
def main_menu():
    # Display the main menu
    print("\nMain Menu:")
    print("1. Create Account")
    print("2. Login")
    print("3. Exit")
```

This function displays the main menu when the application is first run and provides the user with options. The function steps are:

- It shows the user the first main menu options: create an account, log in, or log out.
- It takes the user's selection and calls the appropriate function.

## Petition Menu Function

```
def petition_menu(ep, account):
    # Menu for petition-related actions
    while True:
        print("\nPetition Menu:")
        print("1. Create Petition")
        print("2. View Petitions")
        print("3. Return to Main Menu")
        choice = input("Make your selection: ")

        if choice == "1":
            title = input("Petition Title: ")
            description = input("Petition Description: ")
            ep.create_petition(title, description, account)

        elif choice == "2":
            petition_ids = ep.list_petitions()
            if petition_ids:
                index = int(input("Enter the number of the petition you want to select: ")) - 1
                if 0 <= index < len(petition_ids):
                    sub_menu(ep, account, petition_ids[index])

        elif choice == "3":
            break

        else:
            print("Invalid selection. Please try again.")
```

With this function, after passing the account login menu, a sub-menu appears where the user can perform transactions related to e-petitions. The function steps are:

- Shows the user options regarding e-petitions: creating an e-petition, viewing existing e-petitions, or returning to the main menu.
- Depending on the user's selection, the appropriate function is called or the submenu is exited.

## Sub Menu Function

```
def sub_menu(ep, account, petition_id):  
    # Sub-menu for selected petition actions  
    while True:  
        print("\nPetition Actions Menu:")  
        print("1. Sign Petition")  
        print("2. View Signatures")  
        print("3. Return")  
        choice = input("Make your selection: ")  
  
        if choice == "1":  
            ep.sign_petition(petition_id, account)  
  
        elif choice == "2":  
            ep.get_signatures(petition_id)  
  
        elif choice == "3":  
            break  
  
        else:  
            print("Invalid selection. Please try again.")
```

The purpose of using this function is to direct the user to a sub-menu where he can perform transactions related to a specific e-petition after performing e-petition related transactions in the previous menu. The function steps are:

- All actions that can be performed regarding a particular e-petition are presented to the user: signing the e-petition, viewing signatures, or exiting the submenu.
- Depending on the user's choice, it calls the appropriate function or exits the submenu and returns to the main menu.



## Main Function

```
def main():
    ep = EPetition()
    accounts = {}
    account = None
    while True:
        main_menu()
        choice = input("Make your selection: ")

        if choice == "1":
            account = create_account(accounts)

        elif choice == "2":
            account = login(accounts)
            if account:
                petition_menu(ep, account)

        elif choice == "3":
            print("Exiting...")
            break

        else:
            print("Invalid selection. Please try again.")

if __name__ == "__main__":
    main()
```

The purpose of this function is to execute the main menu loop and allow the user to navigate between these menus. Switch cases are used. The function steps are:

- It allows the creation of ePetition and accounts objects.
- Displays the main menu and calls appropriate functions using switch cases based on user selections.
- The program loop continues until the user logs out.

# Performance of Prototype

## Main Menu

```
...  
Main Menu:  
1. Create Account  
2. Login  
3. Exit  
Make your selection: 
```

When we run our prototype this main menu will host us. We selecting first options. This will gives us a option to create an account.

## Entering the informations

```
Main Menu:  
1. Create Account  
2. Login  
3. Exit  
Make your selection: 1  
Your Full Name: Alim Kadir Kara  
Your ID Number: 20190808013  
Your Password: 123456  
Account successfully created.
```

We select number 1 and then we enter the informations we need to enter. Ultimately account successfully created.

## Petition Menu

```
Main Menu:  
1. Create Account  
2. Login  
3. Exit  
Make your selection: 2  
Your ID Number: 20190808013  
Your Password: 123456  
Login successful.  
  
Petition Menu:  
1. Create Petition  
2. View Petitions  
3. Return to Main Menu  
Make your selection: 
```

After creating an account, the system directs us back to the main menu. Then we enter number 2 and enter the information we entered when creating the account. As a result, the system directs us to the petition menu.

## Petition Creation Process

```
Petition Menu:
1. Create Petition
2. View Petitions
3. Return to Main Menu
Make your selection: 1
Petition Title: Self-sovereign identity
Petition Description: blockchain, decentralized identifiers, and Verifiable Credentials.
Petition successfully created: f4dd8fd19acc38f0b75d5a9bd78f175a22496df2442f5fe755c5c16a0a635b23

Petition Menu:
1. Create Petition
2. View Petitions
3. Return to Main Menu
Make your selection: 
```

Then we choose option 1. After entering the title and description information of the petition we will create, we are automatically assigned a public key.

## Viewing Petition We Created

```
Petition Menu:
1. Create Petition
2. View Petitions
3. Return to Main Menu
Make your selection: 2
Available Petitions:
1. ID: f4dd8fd19acc38f0b75d5a9bd78f175a22496df2442f5fe755c5c16a0a635b23, Title: Self-sovereign identity, Signatures: 0
Enter the number of the petition you want to select: 
```

We can view the petition we just created by selecting the 2nd option. Thus, we see the ID, title name and number of signatures of the petition.

## Signing Petition Successfully

```
Petition Menu:
1. Create Petition
2. View Petitions
3. Return to Main Menu
Make your selection: 2
Available Petitions:
1. ID: f4dd8fd19acc38f0b75d5a9bd78f175a22496df2442f5fe755c5c16a0a635b23, Title: Self-sovereign identity, Signatures: 0
Enter the number of the petition you want to select: 1

Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 1
Petition successfully signed.
```

As you can see, we choose option number 1 to sign the petition we created. And the petition is signed successfully.

## Trying to Sign Same Petition Again

```
Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 1
You have already signed this petition.
```

```
Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 
```

If we want to sign again, the system informs us that we have signed this petition.

## Viewing Petition's ID

```
Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 2
Signatures' IDs:
85424944ab4e2d4cd9bce37772b4b6962887519a07d8293e1b8f3609c364225f

Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 
```

If we choose the second option, view the signature, we can see the ID of the signed petition.

## Exiting the System

```
Petition Actions Menu:
1. Sign Petition
2. View Signatures
3. Return
Make your selection: 3

Petition Menu:
1. Create Petition
2. View Petitions
3. Return to Main Menu
Make your selection: 3

Main Menu:
1. Create Account
2. Login
3. Exit
Make your selection: 3
Exiting...
```

And finally, we log out of the system by selecting options number 3.