

STUDENT : SAARUHAN SELLAPPAN FROM CY-TECH, FRANCE

SUPERVISOR : MR PRADEEP SINGH SHEKHAWAT FROM BK BIET, INDIA

SECURE AND DECENTRALIZED IDENTITY MANAGEMENT



ISSUER, HOLDER AND VERIFIER
PRINCIPLE



Ganache



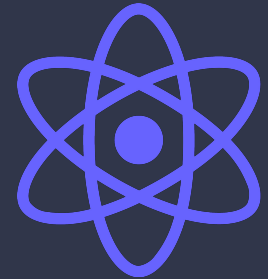
TRUFFLE

SMART CONTRACT AND USE OF
GANACHE AND TRUFFLE



HYPERLEDGER

USE OF THE HYPERLEDGER
ARIES API



WEB APPLICATION

THE PRINCIPLE OF THE ISSUER, HOLDER AND VERIFIER IN A SELF-SOVEREIGN IDENTITY MANAGEMENT

INTRODUCTION TO SELF-SOVEREIGN IDENTITY MANAGEMENT

- WHAT IS SELF-SOVEREIGN IDENTITY MANAGEMENT (SSI)?
- BENEFITS OF SSI: USER CONTROL OF DATA, SECURITY, AND PRIVACY.
- INTRODUCTION OF THE THREE MAIN ACTORS: ISSUER, HOLDER, VERIFIER.



THE PRINCIPLE OF THE ISSUER, HOLDER AND VERIFIER IN A SELF-SOVEREIGN IDENTITY MANAGEMENT

ROLE OF THE ISSUER

Holder

- DEFINITION OF THE ISSUER.



- FUNCTION OF THE ISSUER: ISSUING VERIFIABLE CREDENTIALS (VCs).



- PROCESS: CREATION AND ISSUANCE OF VCS.

- EXAMPLES OF ISSUERS: GOVERNMENT ISSUING PASSPORTS, UNIVERSITY ISSUING DIPLOMAS.

Issuer



Verifier



THE PRINCIPLE OF THE ISSUER, HOLDER AND VERIFIER IN A SELF-SOVEREIGN IDENTITY MANAGEMENT

ROLE OF THE **HOLDER**

Holder

- DEFINITION OF THE HOLDER.



- FUNCTION OF THE HOLDER: RECEIVING, STORING, AND MANAGING VERIFIABLE CREDENTIALS.

- USE OF A DIGITAL WALLET FOR MANAGING VCS.



- IMPORTANCE OF CONSENT AND DATA CONTROL BY THE HOLDER.

- EXAMPLE SCENARIOS: USING A VC TO PROVE IDENTITY OR QUALIFICATIONS.

Issuer



Verifier



THE PRINCIPLE OF THE **ISSUER**, **HOLDER** AND **VERIFIER** IN A SELF-SOVEREIGN IDENTITY MANAGEMENT

ROLE OF THE **VERIFIER**

Holder

- DEFINITION OF THE VERIFIER.



- FUNCTION OF THE VERIFIER: VERIFYING THE AUTHENTICITY OF THE CREDENTIALS PRESENTED BY THE HOLDER.

- VERIFICATION PROCESS: VALIDATION OF THE ISSUER'S DIGITAL SIGNATURE.



- EXAMPLE SCENARIOS: VERIFYING A DIPLOMA FOR EMPLOYMENT, VERIFYING A DRIVER'S LICENSE.



Issuer

Verifier

- BENEFITS FOR THE VERIFIER: INCREASED SECURITY, REDUCTION IN FRAUD AND DOCUMENT FORGERY.

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

INTRODUCTION TO TRUFFLE, GANACHE, AND ETHEREUM IDE

- OVERVIEW OF TOOLS: TRUFFLE, GANACHE, AND ETHEREUM IDE. (QUICK DEMO)



-PURPOSE OF EACH TOOL: DEVELOPMENT, TESTING, AND DEPLOYMENT OF SMART CONTRACTS..



-BENEFITS OF USING THESE TOOLS: SIMPLIFIED DEVELOPMENT WORKFLOW, ROBUST TESTING ENVIRONMENT, AND STREAMLINED DEPLOYMENT PROCESS.



USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

DEVELOPING SMART CONTRACTS WITH ETHEREUM IDE



-OVERVIEW OF ETHEREUM IDE (SUCH AS REMIX): AN ONLINE EDITOR FOR WRITING AND DEPLOYING SMART CONTRACTS.

KEY FEATURES: CODE EDITOR, COMPILER, AND DEPLOYMENT TOOLS.

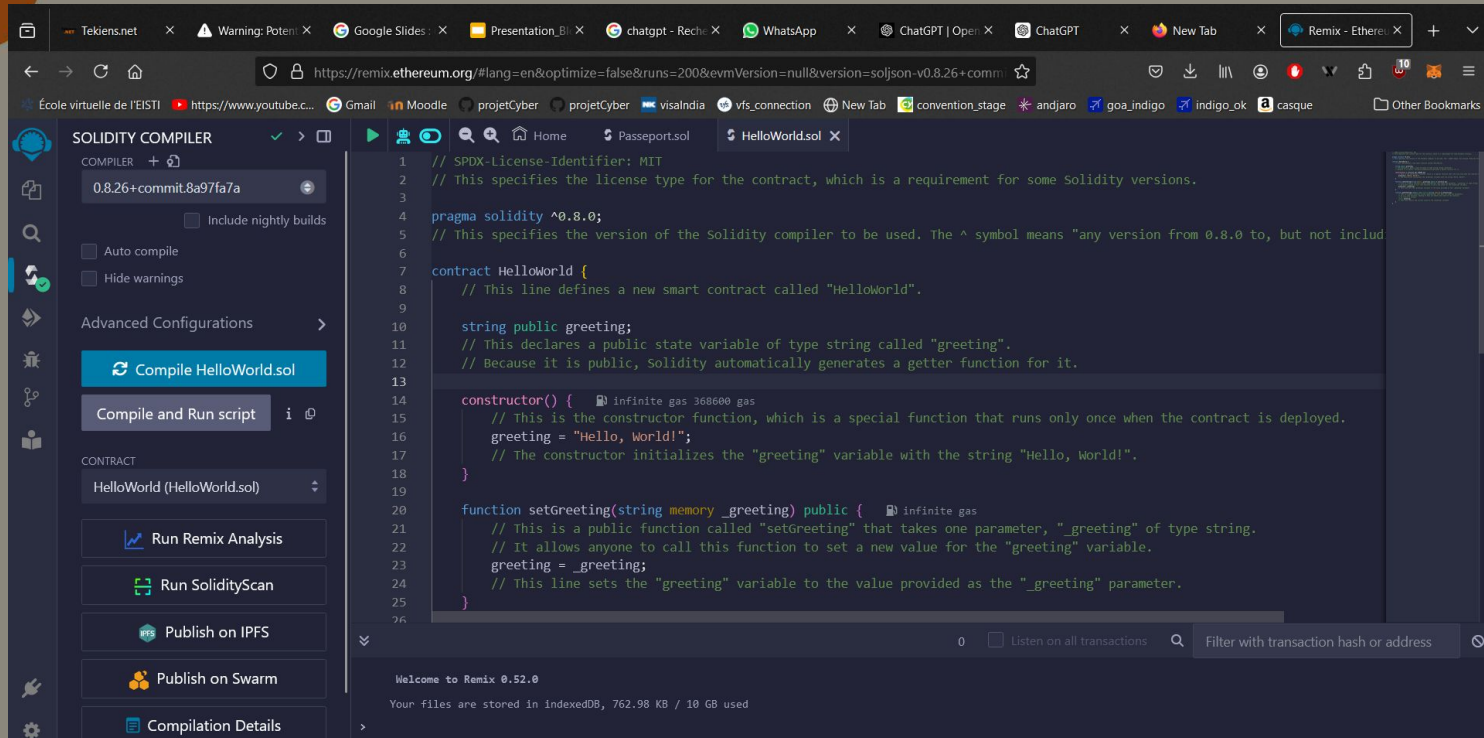
STEPS TO DEVELOP A CONTRACT:

- WRITING THE SMART CONTRACT CODE IN THE IDE.
- COMPILING THE CONTRACT USING THE BUILT-IN COMPILER.



EXAMPLE: WRITING AND COMPILING A SIMPLE "HELLO WORLD" SMART CONTRACT :

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE



The screenshot displays the Remix Ethereum IDE interface. The top browser tab is titled "Remix - Ether..." and the address bar shows the URL "https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit-8a97fa7a". The left sidebar contains a "SOLIDITY COMPILER" section with a dropdown menu set to "0.8.26+commit-8a97fa7a". Below this are checkboxes for "Include nightly builds", "Auto compile", and "Hide warnings". The "Advanced Configurations" section is expanded, showing a "Compile HelloWorld.sol" button and a "Compile and Run script" button. The "CONTRACT" section lists "HelloWorld (HelloWorld.sol)". The main editor area displays the Solidity code for the "HelloWorld" contract, which includes a constructor, a public "greeting" variable, and a "setGreeting" function. The bottom status bar shows "Welcome to Remix 0.52.0" and "Your files are stored in indexedDB, 762.98 KB / 10 GB used".

```
1 // SPDX-License-Identifier: MIT
2 // This specifies the license type for the contract, which is a requirement for some Solidity versions.
3
4 pragma solidity ^0.8.0;
5 // This specifies the version of the Solidity compiler to be used. The ^ symbol means "any version from 0.8.0 to, but not includ
6
7 contract HelloWorld {
8     // This line defines a new smart contract called "HelloWorld".
9
10    string public greeting;
11    // This declares a public state variable of type string called "greeting".
12    // Because it is public, Solidity automatically generates a getter function for it.
13
14    constructor() {
15        // This is the constructor function, which is a special function that runs only once when the contract is deployed.
16        greeting = "Hello, World!";
17        // The constructor initializes the "greeting" variable with the string "Hello, World!".
18    }
19
20    function setGreeting(string memory _greeting) public {
21        // This is a public function called "setGreeting" that takes one parameter, "_greeting" of type string.
22        // It allows anyone to call this function to set a new value for the "greeting" variable.
23        greeting = _greeting;
24        // This line sets the "greeting" variable to the value provided as the "_greeting" parameter.
25    }
26 }
```


USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

DEPLOY & RUN TRANSACTIONS

Custom 3000000

VALUE 0 Wei

CONTRACT HelloWorld - contracts/HelloWorld.sol

evm version: cancun

Deploy

☐ Publish to IPFS

At Address Load contract from Address

Transactions recorded 1

Pinned Contracts (network: vm-cancun)

No pinned contracts found for selected workspace & network

Deployed/Unpinned Contracts

No pinned contracts found for selected workspace & network

Deployed/Unpinned Contracts

HELLOWORLD AT 0XD8B...33FA

HELLOWORLD AT 0XF8E...9FBE

Balance: 0 ETH

setGreeting Hello Saaru !

getGreeting

0: string: Saaru

greeting

0: string: Saaru

Low level interactions

CALLDATA

Transact

```
14 constructor () {
15     // This is the constructor function, which is a special function that runs only once when the contract is deployed.
16     greeting = "Hello, World!";
17     // The constructor initializes the "greeting" variable with the string "Hello, World!".
```

[vm] from: 0x583...eddC4 to: HelloWorld.setGreeting(string) 0xf8e...9fBe8 value: 0 wei data: 0xa41...00000 logs: 0 hash: 0xe84...fff539

status 0x1 Transaction mined and execution succeed

transaction hash 0xe84415da3c20b05aff2017e17eb3bbeda4c0126a32f3be93f81dba56ff539

block hash 0x9850725acc2b071603abad2efe06462bedb6145a12bbca14c6b5b9642c8cdfdd

block number 5

from 0x5838Da6a701c568545dCfcB03Fc875f56beddC4

to HelloWorld.setGreeting(string) 0xf8e81D47203A594245E36C48e151709f0C19f8e8

gas 32373 gas

transaction cost 28150 gas

execution cost 6522 gas

input 0xa41...00000

decoded input { "string _greeting": "Hello Saaru !" }

decoded output {}

AFTER COMPILING

AFTER DEPLOY

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

TRANSITIONING TO TRUFFLE FOR ADVANCED
DEVELOPMENT



-WHAT IS TRUFFLE? A DEVELOPMENT FRAMEWORK FOR ETHEREUM THAT PROVIDES A PROJECT STRUCTURE AND AUTOMATION TOOLS.

KEY FEATURES:

- SIMPLIFIES CONTRACT DEVELOPMENT.
- PROVIDES AUTOMATED TESTING AND MIGRATION SCRIPTS.

-STEPS:INSTALL TRUFFLE: NPM INSTALL -G TRUFFLE.

- INITIALIZE A TRUFFLE PROJECT WITH TRUFFLE INIT.
- IMPORT THE SMART CONTRACT FROM ETHEREUM IDE.

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

WRITING AND TESTING SMART CONTRACTS WITH
TRUFFLE



-WRITING SMART CONTRACTS:

- WRITE SMART CONTRACTS IN SOLIDITY WITHIN THE TRUFFLE PROJECT.

-COMPILING CONTRACTS: COMPILE CONTRACTS USING TRUFFLE COMPILE.

-TESTING CONTRACTS: WRITE AND RUN TESTS USING TRUFFLE TEST.

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

SETTING UP AND USING GANACHE



-WHAT IS GANACHE? A PERSONAL BLOCKCHAIN FOR ETHEREUM DEVELOPMENT THAT ALLOWS FOR FASTER AND EASIER TESTING.

KEY FEATURES:

PROVIDES A LOCAL BLOCKCHAIN FOR DEVELOPMENT.

ALLOWS FOR TRANSACTION LOGGING AND INSPECTION.

-STEPS:

INSTALL GANACHE (VIA GANACHE CLI OR GANACHE GUI).

CONFIGURE AND START THE LOCAL BLOCKCHAIN.

CONNECT TRUFFLE TO GANACHE WITH TRUFFLE-CONFIG.JS.

USING TRUFFLE, GANACHE, AND CREATING SMART CONTRACTS, AND DEPLOYING ON ETHEREUM IDE

QUICK DEMO

```
//  
development: {  
  host: "127.0.0.1",  
  port: 7545,  
  network_id: "*",  
},
```

```
}  
truffle(development)> let passport = await instance.getPassport(account[0]);  
undefined  
truffle(development)> console.log("Passport:", passport);  
Passport: result {  
  0: 'John Doe',  
  1: 'American',  
  2: '123456789',  
  3: '1990-01-01',  
  4: 'New York',  
  5: BN {  
    negative: 0,  
    words: [ 1664264, 14, 0 empty item ],  
    length: 2,  
    red: null  
  }  
},  
  6: BN {  
    negative: 0,  
    words: [ 1060600, 25, 0 empty item ],  
    length: 2,  
    red: null  
  }  
}  
truffle(development)> await instance.updatePassport("John Smith", "American", "123456789", "1990-01-01", "Los Angeles", 162526000, 168832000, { from  
account: 0 }));  
{  
  tx: '0x28ec26a2ba8cd9d70e64bd29003395ddfd3472d34260009cc559dd2817f5d70',  
  receipt: {  
    transactionHash: '0x28ec26a2ba8cd9d70e64bd29003395ddfd3472d34260009cc559dd2817f5d70',  
    transactionIndex: 0,  
    blockNumber: 50,  
    blockHash: '0x98c2ac5fc823e33b062e84e927a1cf5af6fde80bb57c38b2564ca5e6d8d54939',  
    from: '0x775cd1d9547d07C282330a31775C8752F2CE90E2',  
    to: '0x015a5a2060963c272d77a30a9f9f5d07dd07',  
    cumulativeGasUsed: 72800,  
    gasUsed: 72800,  
    gasPrice: 20000000000,  
    effectiveGasPrice: 20000000000,  
    isSuccessful: true  
  }  
}
```

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 51	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE PASSEPORTLOCAL	SWITCH	⚙️
---------------------	--------------------------	----------------------	-------------------	--------------------	-------------------------------------	-----------------------------	-----------------------------	--------	----

← BACK

BLOCK 50

GAS USED	GAS LIMIT	MINED ON	BLOCK HASH
72800	6721975	2024-07-28 08:11:46	0x98c2ac5fc823e33b062e84e927a1cf5af6fde80bb57c38b2564ca5e6d8d54939

TX HASH

0x28ec26a2ba8cd9d70e64bd29003395ddfd3472d34260009cc559dd2817f5d70

FROM ADDRESS

0x775cd1d9547d07C282330a31775C8752F2CE90E2

TO CONTRACT ADDRESS

DigitalPassport

GAS USED

72800

VALUE

0

CONTRACT CALL

USING HYPERLEDGER ARIES AND CREATING A REACT PLATFORM TO INTERACT WITH THE

BLOCKCHAIN

INTRODUCTION TO HYPERLEDGER ARIES



-**OVERVIEW:** HYPERLEDGER ARIES IS A PROJECT WITHIN THE HYPERLEDGER ECOSYSTEM AIMED AT BUILDING INTEROPERABLE, DECENTRALIZED IDENTITY SOLUTIONS.

-**PURPOSE:** IT FACILITATES SECURE PEER-TO-PEER INTERACTIONS, IDENTITY MANAGEMENT, AND THE ISSUANCE AND VERIFICATION OF VERIFIABLE CREDENTIALS.

-**ECOSYSTEM:** ARIES INTEGRATES WITH OTHER HYPERLEDGER PROJECTS SUCH AS HYPERLEDGER INDY FOR DECENTRALIZED IDENTITY AND HYPERLEDGER URSAL FOR CRYPTOGRAPHIC FUNCTIONS.

USING HYPERLEDGER ARIES AND CREATING A REACT PLATFORM TO INTERACT WITH THE

BLOCKCHAIN

CORE COMPONENTS OF HYPERLEDGER ARIES



- **ARIES FRAMEWORK:** PROVIDES A SET OF TOOLS AND LIBRARIES FOR DEVELOPERS TO CREATE IDENTITY SOLUTIONS.
- **AGENTS:** SOFTWARE ENTITIES THAT MANAGE IDENTITIES, CREDENTIALS, AND INTERACTIONS ON BEHALF OF INDIVIDUALS OR ORGANIZATIONS.
- **PROTOCOLS:** DEFINE STANDARD METHODS FOR SECURE COMMUNICATION AND CREDENTIAL MANAGEMENT BETWEEN AGENTS.
- **WALLETS:** SECURE STORAGE SOLUTIONS FOR KEYS AND CREDENTIALS, ENSURING THE PRIVACY AND INTEGRITY OF IDENTITY DATA.

USING HYPERLEDGER ARIES AND CREATING A REACT PLATFORM TO INTERACT WITH THE

BLOCKCHAIN

VERIFIABLE CREDENTIALS AND PROOFS



- VERIFIABLE CREDENTIALS: DIGITAL REPRESENTATIONS OF CREDENTIALS THAT CAN BE ISSUED, HELD, AND VERIFIED.
- ISSUING AND HOLDING: THE PROCESS OF CREATING, ISSUING, AND STORING CREDENTIALS USING ARIES.
- PRESENTING PROOFS: MECHANISMS FOR HOLDERS TO PROVE POSSESSION OF CREDENTIALS WITHOUT REVEALING SENSITIVE DATA, ENSURING PRIVACY.

USING HYPERLEDGER ARIES AND CREATING A REACT PLATFORM TO INTERACT WITH THE

BLOCKCHAIN

BUILDING WITH HYPERLEDGER ARIES



- DEVELOPMENT TOOLS: OVERVIEW OF THE SDKS AND APIS AVAILABLE FOR BUILDING APPLICATIONS WITH ARIES.
- INTEGRATION: METHODS FOR INTEGRATING ARIES WITH OTHER BLOCKCHAIN SOLUTIONS, SUCH AS HYPERLEDGER INDY FOR DECENTRALIZED IDENTITY MANAGEMENT.
- EXAMPLE USE CASES: APPLICATIONS IN VARIOUS INDUSTRIES, INCLUDING IDENTITY VERIFICATION, SUPPLY CHAIN MANAGEMENT, AND FINANCIAL SERVICES.

USING HYPERLEDGER ARIES AND CREATING A REACT PLATFORM TO INTERACT WITH THE

BLOCKCHAIN

DEMO



Non sécurisé 0.0.0.0:11003/api/doc

Swagger powered by SMARTBEAN /api/docs/swagger.json [Explore](#)

Verifier v0.12.1

/api/docs/swagger.json

action-menu Menu interaction over connection

- POST** /action-menu/{conn_id}/close Close the active menu associated with a connection
- POST** /action-menu/{conn_id}/fetch Fetch the active menu
- POST** /action-menu/{conn_id}/perform Perform an action associated with the active menu
- POST** /action-menu/{conn_id}/request Request the active menu
- POST** /action-menu/{conn_id}/send-menu Send an action menu to a connection

basicsmessage Simple messaging

Specification: <https://github.com/hyperledger/aries-rfcs/tree/527649ec3aa2a8fd47a7bb6c57f918ff8bcb5e6c/features/0095-basic-message>

- POST** /connections/{conn_id}/send-message Send a basic message to a connection

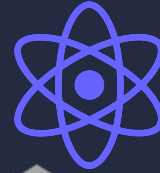
connection Connection management

Specification: <https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol>

- GET** /connections Query agent-to-agent connections
- POST** /connections/create-invitation Create a new connection invitation

USING HYPERLEDGER ARIES IN REACT

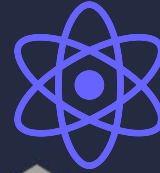
SETTING UP A REACT PROJECT



- INTRODUCTION TO REACT AS A POWERFUL FRONTEND FRAMEWORK FOR BUILDING INTERACTIVE USER INTERFACES.
- CREATING A NEW REACT PROJECT USING CREATE REACT APP.
- ORGANIZING YOUR REACT PROJECT TO FACILITATE INTEGRATION WITH HYPERLEDGER ARIES.

USING HYPERLEDGER ARIES IN REACT

INTEGRATING **HYPERLEDGER ARIES** WITH
REACT



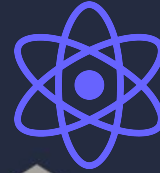
-INSTALLING DEPENDENCIES: REQUIRED LIBRARIES AND PACKAGES FOR INTEGRATING HYPERLEDGER ARIES (E.G., @HYPERLEDGER/ARIES-FRAMEWORK-JAVASCRIPT).

-INITIALIZING ARIES: SETTING UP AN ARIES AGENT IN COMMAND-LINE (VERIFIER, HOLDER AND ISSUER)

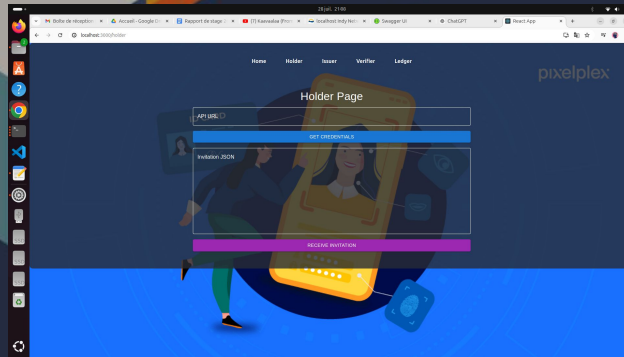
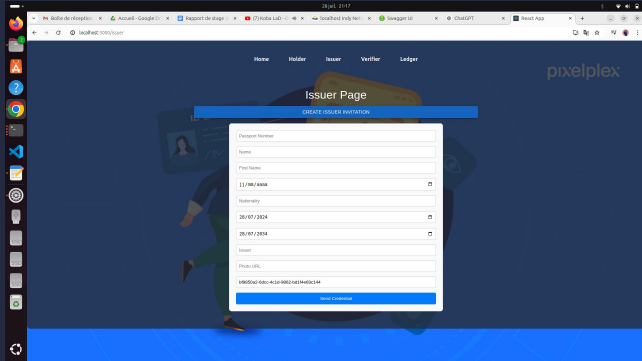
-CONNECTING TO BLOCKCHAIN: CONFIGURING YOUR ARIES AGENT TO CONNECT TO YOUR CHOSEN BLOCKCHAIN NETWORK FOR IDENTITY MANAGEMENT.

USING HYPERLEDGER ARIES IN REACT

DEVELOPING THE USER INTERFACE



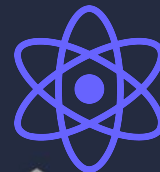
- CREATING COMPONENTS: BUILDING REACT COMPONENTS TO HANDLE USER INTERACTIONS AND DISPLAY DATA.
- MANAGING STATE: UTILIZING REACT STATE MANAGEMENT SOLUTIONS (E.G., REDUX, CONTEXT API) TO MANAGE ARIES-RELATED DATA.
- UI ELEMENTS: DESIGNING FORMS AND INTERFACES FOR CREDENTIAL ISSUANCE, DISPLAY, AND PROOF PRESENTATION.



USING HYPERLEDGER ARIES IN REACT



HANDLING CREDENTIALS AND PROOFS IN
REACT



FINAL DEMO

