EMERGENCY ACCESS CONTROL CREDIT LEVEL BASED SYSTEM USING BLOCKCHAIN WITH CLOUD STORAGE

CONFIGURATION MANUAL

Sneha Sankaran Arivukkarasu
X23307374
MSC CLOUD COMPUTING
(2024-2025)
NATIONAL COLLEGE OF IRELAND

Supervisor: XXX

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SNEHA SANKARAN ARIVUKKARASU
Student ID:	X23307374
Programme:	……………………………………… **Year:** …………………..
Module:	………………………………………………………………
Supervisor:	………………………………………………………………
Submission Due Date:	………………………………………………………………
Project Title:	EMERGENCY ACCESS CONTROL CREDIT LEVEL BASED SYSTEM USING BLOCKCHAIN WITH CLOUD STORAGE
Word Count:	………………………… **Page Count**………………………………..

Signature: ………………………………………………………………

Date: ………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

INTRODUCTION:

This guide provides full technical design and operating protocols required to ensure deployment and operation of the blockchain-oriented Credit-Level Based Emergency Access Control System as described in this work. The main goal of the present document is to provide a straightforward, accurate, and repeatable procedure of preparing the development environment, deploying the Ethereum smart contract on the Sepolia test network, the integration of the same with the backend, which is developed with the Django framework, and the setting up of AWS S3 to store and retrieve patient information in a secure manner. Reading this guidance will result in a working example of the system comprising:

Registration of doctors on the blockchain, including the granting of credits on level of the assigned dollars. The access request is checked by means of smart contract. The creation of the temporary tokens having rights to the genuine requests. Secure patient report access was in the AWS S3. In this document, it is already known that the operator has privileges to the source code repository of the project, AWS, Infura, and Ethereum wallet settings.

Prerequisites:

- Prerequisites To start off, ensure that the software listed below is installed on your local machine to such an extent that it can be accessed via the command-line:
- Git - To use version control as well as to clone the project on GitHub.
- Currently Technology: Html DOM Nomination backend language Html backend language Python (v3.10+).
- Pip, python package installer and uninstaller.
- Django (right now stable) Back finish logic web framework.
- AWS CLI (v2) - To enable the utilization of AWS S3.
- MetaMask Wallet-To control the blockchain account.
- Infura Account- to connect to Ethereum Sepolia test net.
- Web3.py is Python blockchain interaction library.

Development of Python Virtual Environment:

Isolate dependency by making a project-specific Python virtual environment.

python -m venv venv

Start the virtual environment

venv\Scripts\activate

Adding Dependencies:

When the virtual environment is activated, install the project requirements found in requirements.txt file:

pip install requirements.txt

This will install all necessary **Python module such as Django, boto3, and web3.**

Environmental Variables setup:

In the root folder of the project, create a file named. env, and put the contents below:

```
INFURA_URL=https://sepolia.infura.io/v3/< your ip project id >
CONTRACT_ADDRESS=<your-deployed-contract-address>
PRIVATE_KEY=<your-private-key>
AWS_ACCESS_KEY_ID=<your-aws-access-key>
AWS_SECRET_ACCESS_KEY=<your-aws-secret-key>
AWS_REGION=us-east-1
S3_BUCKET_NAME=access-bucket-sneha-project
```

Blockchain Deployment:

This section outlines the steps to deploy Ethereum smart contract deployed in the Credit-Level Based Emergency Access Control System. The deployment may be done through Remix IDE and MetaMask or by web3.py and Infura programmatically**.**

Compilation of Smart Contracts

- In Remix IDE, open the Solidity smart contract file AccessControl.sol Create a new file called AccessControl.sol Enter the following code in the file Open the Solidity SFTP file called Solidity-3-smart-contracts-function-in-access-control. sftp

- Make sure that the version of the compiler corresponds to the version reflected in the pragma that is present in the contract (e.g., pragma solidity ^0.8.0;).

- Go to Compile Access Control. sol and make sure there are no compilation errors.

Deployment to Ethereum sepolia:

- Transferred to Ethereum Sepolia
- Remix + MetaMask
- In Remix, Select Deploy & Run Transactions.
- meta- Select Injected Provider - MetaMask as environment.
- Link MetaMask with Sepolia test network.
- select Deploy and authorize the transaction in MetaMask.
- Take the address of the deployed contract and retain it to configure.

Contract ABI Storage: Once it is deployed, the contract ABI on remix can be saved as contract_abi.json in the project folder. It is this file that the Django backend needs in order to communicate with the smart contract.

AWS S3 Set up

This section outlines the procedure to set up Amazon Simple Storage Service (S3) to storage and retrieve all the patient data utilized in the Credit-Level Based Emergency Access Control System so as to provide secure storage and retrieval of patient information.

Creating S3 bucket

- Visit the AWS Management Console.
- Go to S3 and create bucket.
- Enter Bucket Name:
- pgsql
- access-bucket-sneha-project
- Region:
- us-east-1
- Leave the configuration of buckets to default but set Block all public access to be active to protect an asset.
- Hit Create bucket.
- Uploading Patient Data File
- In the AWS S3 console, open the bucket created.
- Click Upload and choose patients.csv.
- Make the file resides in the root of the bucket.
- Check the upload.

Local Execution of the Application

The following describes how to run the Credit-Level Based Emergency Access Control System in a local development environment.

The following migration Django commands can be used to initialize the schema of the database:

- python manage.py migrate
- Initialising Development Server
- Start the Django development server with:
- python manage.py run server
- By default the application shall be accessible at:
- localhost: 127.0.0.1:8000
- Getting on to the Application
- Start your web browser and point it to http://127.0.0.1:8000.
- Register a new doctor (give a wallet address, name, credit level) by using the Doctor Registration page.

- After registering, the doctor would be redirected to his dashboard where he or she can find the list of patients Read out of the patients.csv file in AWS S3.

- Request Access on a given patient will initiate the process of access control which is blockchain based.

- Once this is accepted, a temporary token is created which allows viewing the details of the patient within five minutes.

Outfitting on AWS EC2

This section explains what procedures this process entails in deploying the Credit-Level Based Emergency Access Control System in Amazon EC2 instance to access it in the cloud.

Logging On the EC2 Instance
Make sure you downloaded the EC2 .pem key file.
Launch a terminal and log on to the instance by:

ssh -i accesscontrol.pem ubuntu@EC2-Public-IP
sudo apt-upgrade
sudo apt update && sudo apt install python3-pip python3-venv git awscli –
python manage.py makemigrations then python manage.py migrate
python manage.py runserver 0.0.0.0:8000

Conclusion:

Using the steps detailed in this manual, an operator is able to configure, deploy, and run the Credit-Level Based Emergency Access Control System both through local and cloud deployments. Configuration process consists of:
Installing the local framework **Python, Django, AWS CLI, and Web3.py.**
Interaction with Ethereum smart contract on Sepolia test network with Infura and MetaMask or Web3.py.Setting-up AWS S3 to encrypt and retrieve information about patients Running the system locally to do a test. Publishing the application to AWS EC2 instance that has access in the cloud. Such a comprehensive configuration allows safe doctor registration including the blockchain confirmation of identity, a limited amount of access to the patient data depending on the credit status, creation and subsequent recuperation of tokens with definite time-related restrictions, and retrieval of data in the AWS S3.The final system is an improved, scalable, and secure way to provide emergency access to a patients record along with preserving data integrity and privacy utilizing integration with blockchain and cloud technologies.