**Academic E-Certificate Verification System using Algorand Blockchain**

This Dapp is Built on Python with Flask web Framework that interacts with the Algorand blockchain. The application provides functionality for logging in with a mnemonic For a university For uploading Academic E-Certificates on Algorand Blockchain, and anyone Can Retrieve the E-Certificate and verify that the E-Certificate is from that university using the E-Certificate ID Given by the University.

The application uses the `Flask` library for building the web application, `algosdk` for interacting with the Algorand blockchain, `json` for serializing and deserializing JSON data, `base64` for encoding and decoding base64 data, and `flask\_cors` for enabling Cross-Origin Resource Sharing (CORS).

The application has the following endpoints:

* **/:** The home page route. It renders an HTML template that contains a login form and links to other pages.
* **/uc:** The upload E-Certificate page route. It renders an HTML template that contains a form for uploading a E-Certificate.
* **/login:** The login route. It receives a POST request with a mnemonic from a form, derives the private key from the mnemonic, and returns a JSON response with a status indicating whether the mnemonic is valid and owned by a university.
* **/logout:** The logout route. It receives a GET request and removes the user session from the Flask session object.
* **/auth:** The authentication route. It receives a GET request and returns a JSON response with a status indicating whether the user is authenticated.
* **/upload\_certificate:** The upload E-Certificate route. It receives a POST request with a E-Certificate data from a form, signs the E-Certificate data with the user's private key, and submits the signed transaction to the Algorand network. It then returns a JSON response with a status indicating whether the transaction was successful and the transaction ID and block number of the transaction.
* **/get\_certificate:** The get E-Certificate route. It receives a POST request with a transaction ID and searches the Algorand blockchain for a transaction with the specified ID. If the transaction is found and it contains a valid E-Certificate, it returns a JSON response with a status indicating that the E-Certificate was found and the E-Certificate data. Otherwise, it returns a JSON response with a status indicating that the E-Certificate was not found.

**Why This Application Needs The Use Of Algorand Blockchain Instead of Regular DB ?**

Algorand blockchain can provide several benefits over a regular database when it comes to academic E-Certificate verification, including:

1. **Decentralization:** Algorand is a decentralized blockchain, which means that there is no single point of failure. This ensures that the verification process cannot be manipulated or tampered with by any single entity, making it more secure and trustworthy.
2. **Immutability:** Once a E-Certificate is verified and stored on the Algorand blockchain, it cannot be modified or deleted. This ensures that the integrity of the E-Certificate remains intact and cannot be tampered with, providing a more reliable and transparent verification process.
3. **Transparency:** Algorand provides transparency by allowing anyone to access the blockchain and verify the E-Certificates. This eliminates the need for a centralized authority to verify the E-Certificates, making the verification process more efficient and cost-effective.
4. **Speed:** Algorand is designed to process transactions quickly and efficiently, making it ideal for verifying large volumes of E-Certificates in a short period of time.
5. **Cost-effectiveness:** Algorand offers a cost-effective solution for E-Certificate verification compared to traditional methods, such as relying on third-party verification services.

Overall, using Algorand blockchain for academic E-Certificate verification can provide a secure, transparent, efficient, and cost-effective way to verify E-Certificates while ensuring their integrity and immutability.

**Algorand Facilities Used By The Application :**

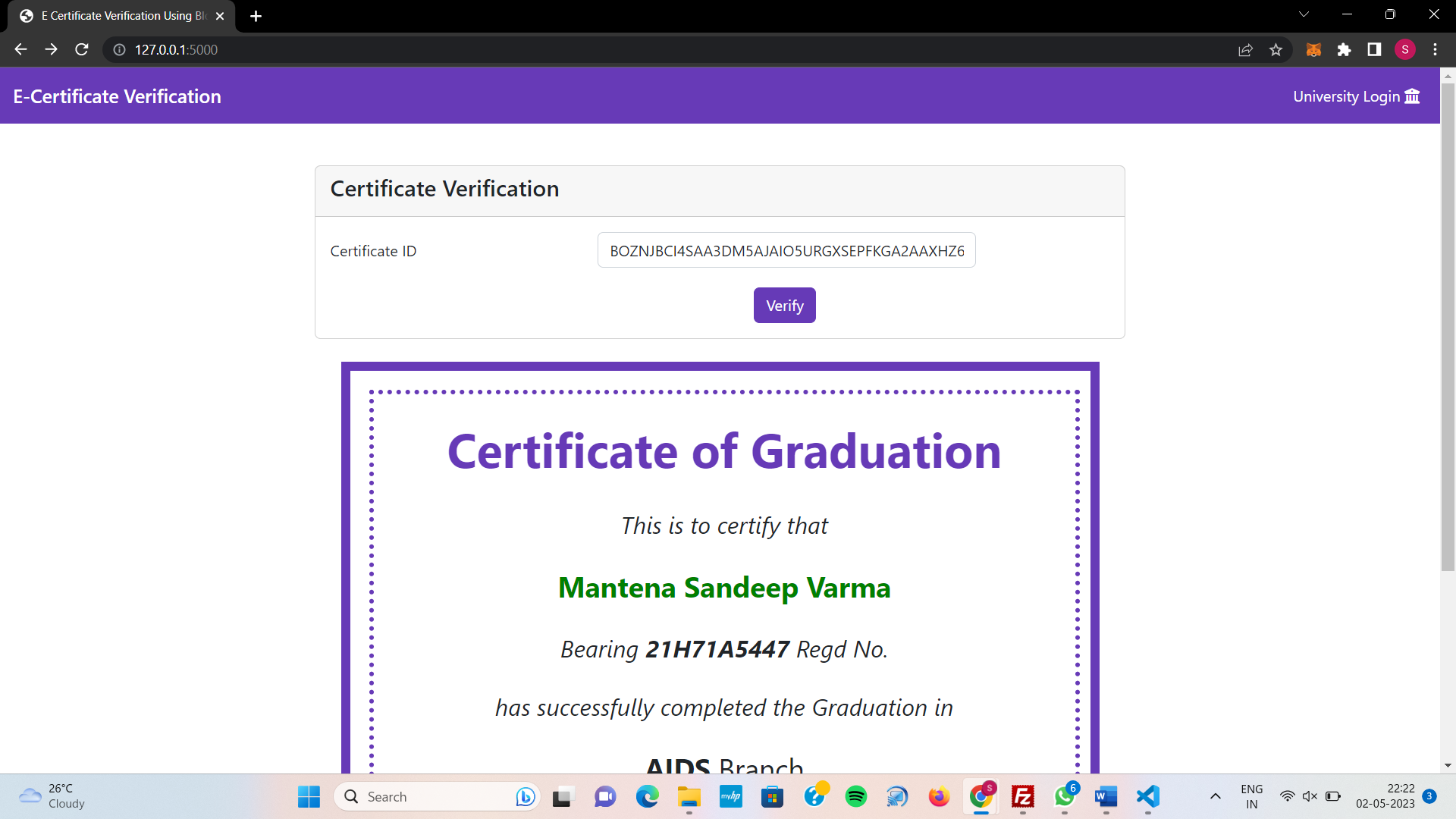
1. Algorand Account : An Algorand Account For the University is used for Uploading the E-Certificates By Making The Transactions with E-Certificate Data to the Algorand Blockchain.
2. PureStake API : An API Key is used in the Application Which is used for Interacting With The Algorand Blockchain.
3. Signatures : While Uploading The E-Certificates to the Blockchain, The Roll No of The Student is Signed With The private key of the university and the signature is also uploaded to the blockchain along with the E-Certificate. This Signature Will Be Verified When The E-Certificate is retrieved from the Blockchain, To Check the Origin of the E-Certificate.
4. AlgoSDK : An Python Development Kit For Building Applications on Algorand Blockchain which contains various Classes & Utility Functions regarding Transactions, Cryptography, Signatures etc. Some of The Functions used in this Application are :

* **AlgodClient**: The AlgodClient is used to create a client that communicates with an Algorand node. It is used to get suggested transaction parameters and send transactions to the Algorand network.
* **IndexerClient**: The IndexerClient is used to create a client that communicates with an Algorand indexer node. It is used to search transactions and retrieve information about them.
* **util.sign\_bytes**: This function is used to sign a byte string with a private key.
* **transaction.PaymentTxn : This class is used to create a payment transaction.**
* **txn.sign:** This method is used to sign a transaction with a private key.
* **algod\_client.send\_transaction:** This method is used to send a signed transaction to the Algorand network.
* **algod\_client.pending\_transaction\_info:** This method is used to get information about a pending transaction, such as the confirmed round.
* **mnemonic.to\_private\_key:** This function is used to derive a private key from a mnemonic.
* **account.address\_from\_private\_key:** This function is used to get the public address corresponding to a private key.

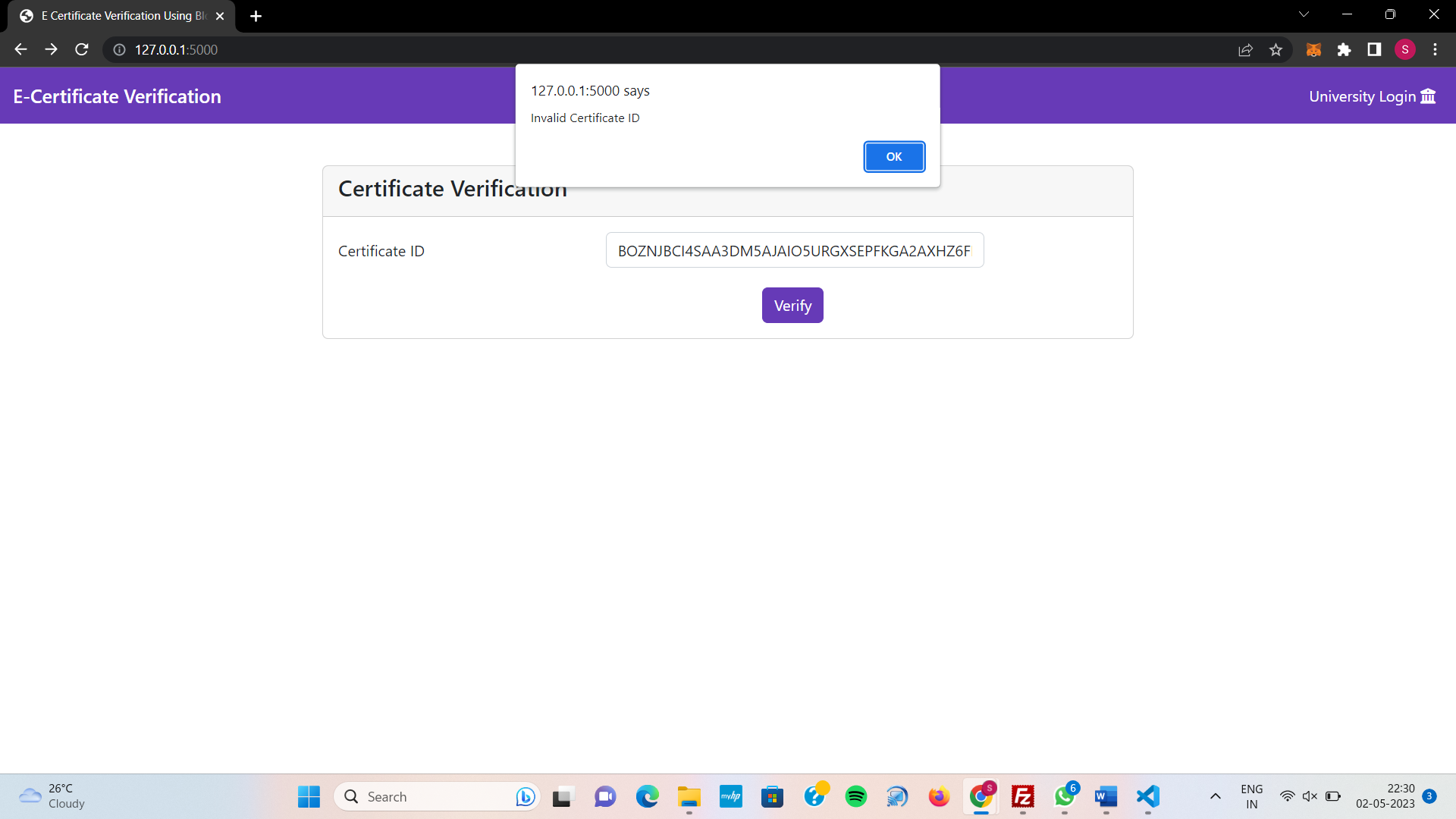
**Usage Of The Application :**

The Application Has 2 Main Functionalities in two Pages :

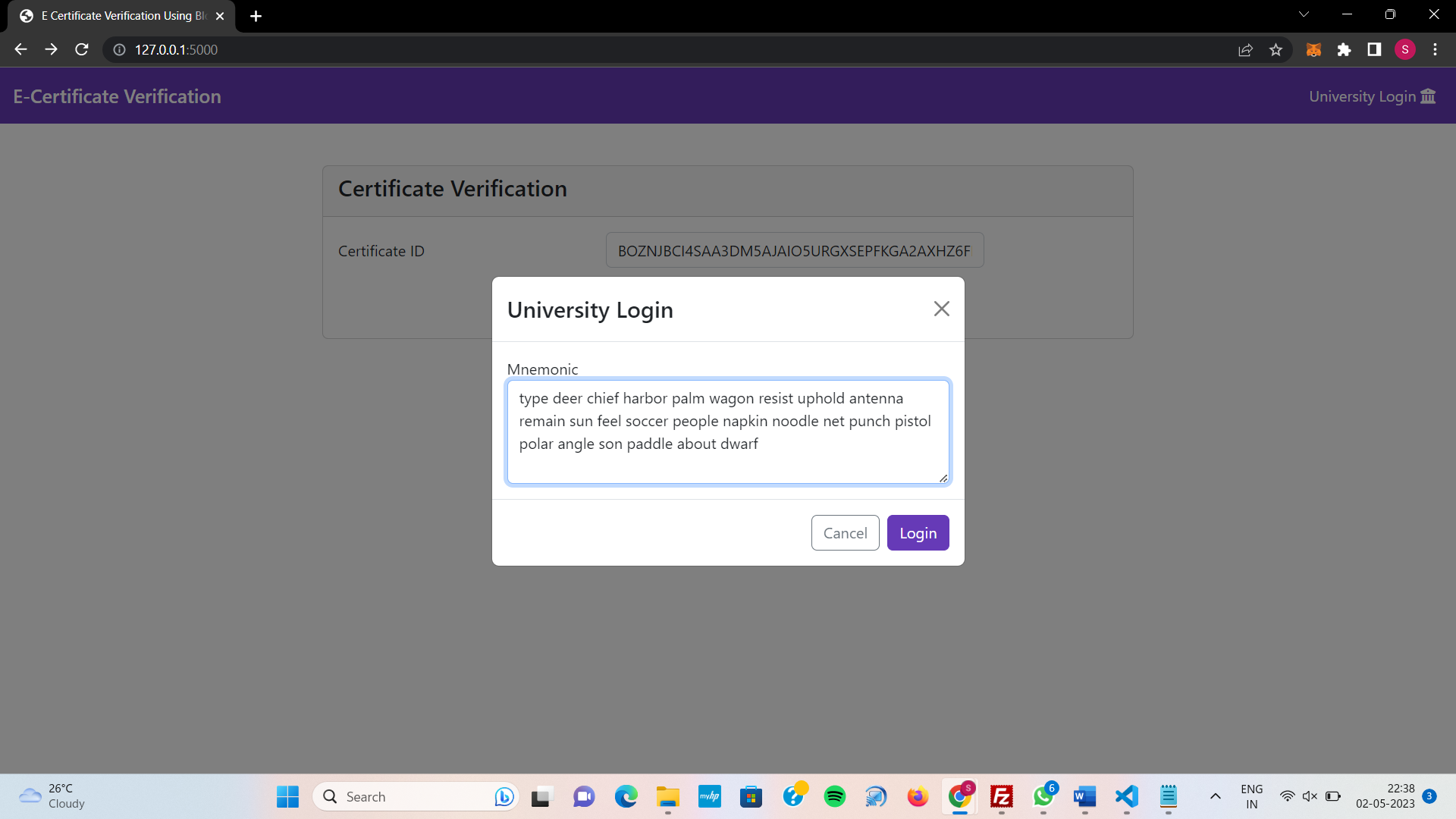
1. E-Certificate Verification ( “/” route)
2. E-Certificate Upload (“/uc” route) - **Only For University**
3. **E-Certificate Verification :** Just Enter The Certificate ID Provided By The University in Certificate ID Field and Enter Verify, If it is a Valid University Issued Certificate, Then the **Certificate will be Displayed** as Below :



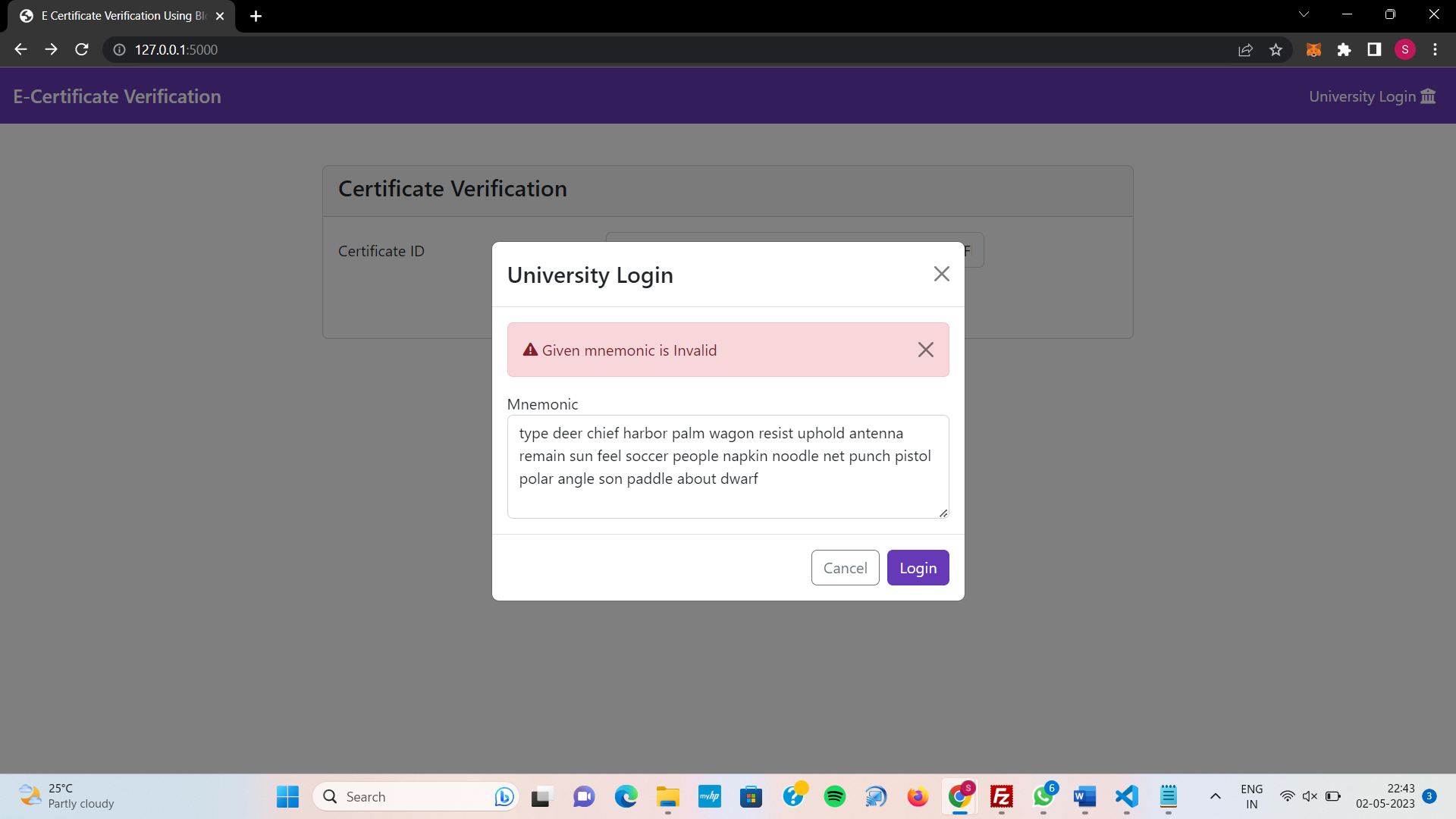
If the Certificate ID is invalid or the Certificate ID is not issued By The University, then the alert with “**Invalid Certificate ID**” will be Displayed as Shown Below :



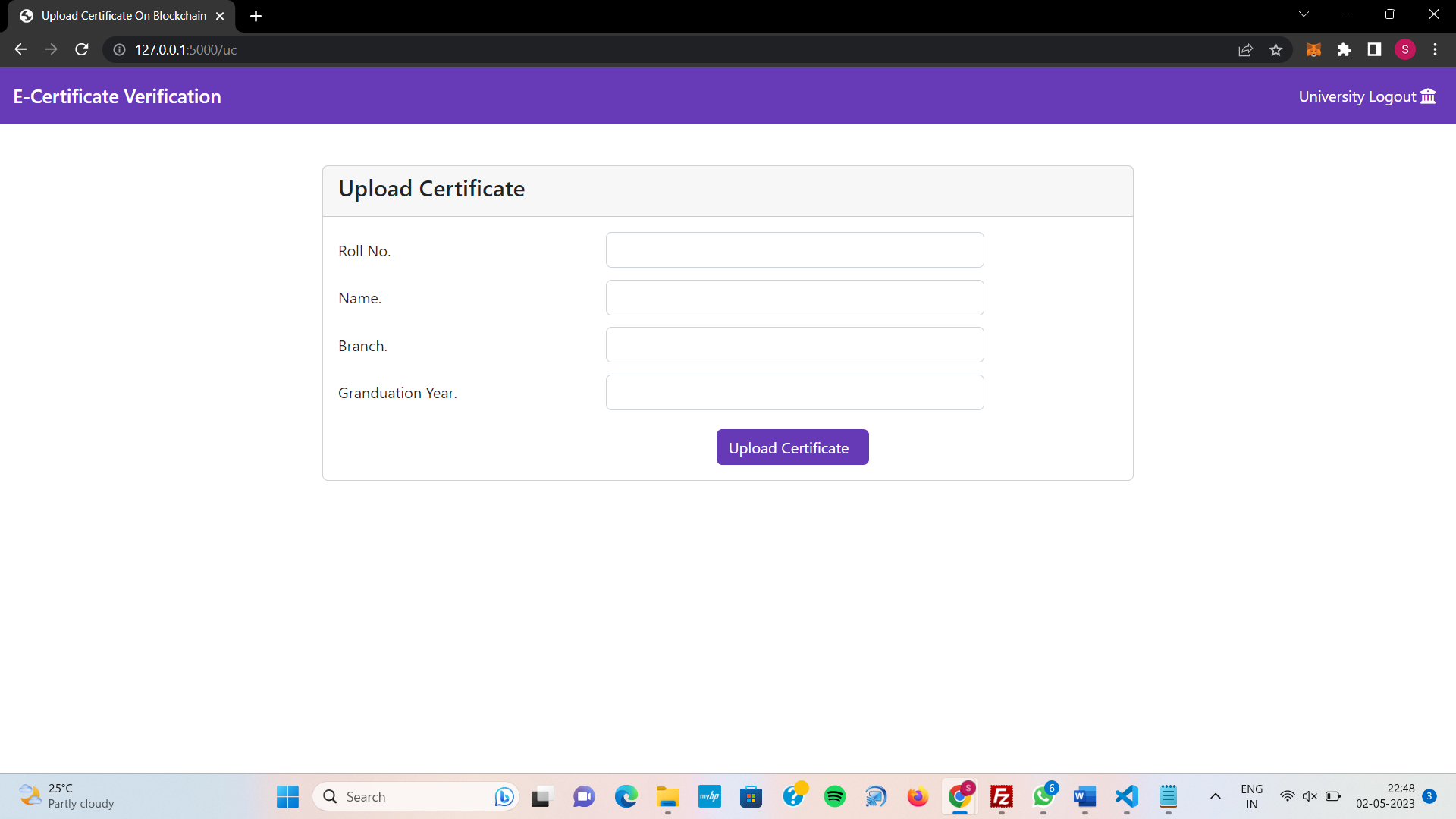
1. **E-Certificate Upload :** This Functionality is Available For The University, So We have To **Login** By Clicking University Login Icon in MenuBar of the Dapp and Entering The University Algorand Account Mnemonic as Shown Below :



Now Click Login Button To Continue To The Upload Certificate Page, If Anything Wrong in the Mnemonic Key Leads To **Login Failure** Which Is Indicated By a Alert Shown Below :

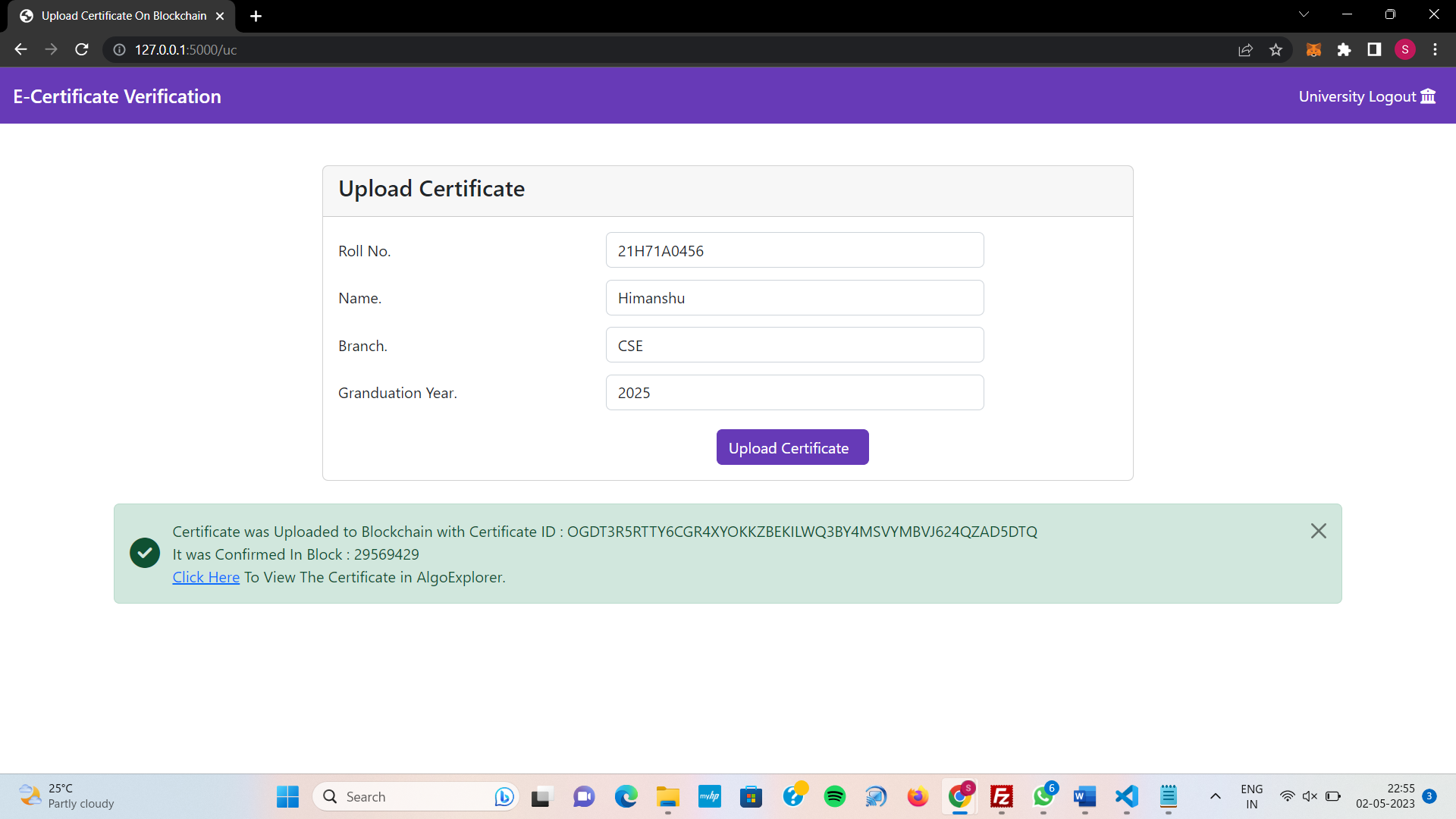


If The Login is Successful Then You Will be **Redirect to the Page** With a Form To Upload E-Certificates To The Blockchain as Shown Below :



Now, If We Fill the Form For The E-Certificate Details And hit Upload Certificate Button, the Data With Signature of University is Placed On The Blockchain and Displays The Certificate ID and Block Number (In Which Block The Data is Stored) After The Block Confirmation in the Blockchain, This Proccess Usually Takes 4-5 Seconds.

After Successful Upload of E-Certificate, The Below Success Alert will be Displayed :



**Note :** If We Click on Click Here Hyperlink in the Alert , We Can **View The Transaction** of E-certificate Data Which Was Signed By The University in the **AlgoExplorer**.