Connected Data Store.

Problem Statement:

- The problem statement revolves around creating a Connected Data Store, similar to a blockchain, with specific requirements such as maintaining data integrity through hashing and ensuring a certain pattern in the hash values.

Solution Approach:

- Python is chosen as the programming language due to its simplicity and versatility.
- Streamlit is used to create a user-friendly web interface for interacting with the Connected Data Store.
- Hashlib, a built-in Python library, is utilized for generating cryptographic hashes.

System Architecture:

- The system consists of two main classes: **DataStoreObject** and **ConnectedDataStore**.
- **DataStoreObject** represents each individual block in the Connected Data Store, while **ConnectedDataStore** manages the collection of blocks.
- Interaction between the user and the system is facilitated through a Streamlit web interface.

Design:

- **DataStoreObject** includes properties for index, data, previous hash, current hash, and correction value.
- The **generate_hash()** method calculates the hash value based on the object's properties.
- A **calculate_correction_value()** method is included to compute the correction value for ensuring the hash pattern.
- The **update_hash_with_correction()** method updates the hash with the newly calculated correction value.

Implementation Details:

- The code leverages object-oriented programming principles to encapsulate the logic related to data store objects and the connected data store.
- Error handling is implemented to catch and display any errors that may occur during the addition of data store objects.
- The Streamlit UI allows users to input data and view the blocks added to the Connected Data Store.