BlockChain project

In this example, the `Block` class in Models has the following properties:

- `Index`: Represents the position of the block in the blockchain.
- `Timestamp`: Indicates the time when the block was created.
- `PreviousHash`: Stores the hash of the previous block in the chain, ensuring the integrity and continuity of the blockchain.
- `Transactions`: An array or list of transactions included in the block.
- `Hash`: The hash value of the current block, which is calculated based on the other properties using the SHA256 hashing algorithm.

The `CalculateHash()` method generates the hash value by combining the block's properties and hashing the resulting string

In this updated version, we've added a `Nonce` property to the `Block` class, which represents a value that miners increment to find the desired hash with a specific prefix (e.g., "0000"). The `MineBlock()` method uses a simple proof-of-work algorithm by repeatedly calculating the hash until a hash with the required prefix is found.

The `MineBlock()` method generates hashes using the `CalculateHash()` method, which now includes the `Nonce` value in the hash calculation. The `Nonce` value is incremented until a valid hash is obtained.

- The `chain` variable is a list that stores the blocks in the blockchain.
- The `currentTransactions` variable is a list that holds pending transactions before they are included in a block.
- The `CreateGenesisBlock()` method initializes the blockchain with a genesis block, which is the first block in the chain.
- The `AddTransaction()` method adds a new transaction to the list of pending transactions.
- The `MineBlock()` method creates a new block by mining it using the proof-of-work mechanism. It adds the block to the blockchain and clears the list of pending transactions.
- The `IsValid()` method validates the integrity of the blockchain by checking the hashes and previous hash references of each block in the chain.