**Things needed to learn for this project**

1. What is the python language ?
2. Basic concept related with the python mostly related with python function .
3. Installation of the python from the official website
4. What is blockchain technology ?
5. What is use of blockchain technology?
6. What is the difference between centralized and decentralized system?
7. **How a Blockchain Works:** Learn how blocks are created, linked together through cryptographic hashes, and validated using Proof of Work (PoW).
8. **Transactions and Blocks**: Understand the role of transactions in a blockchain and how they get added to a block.
9. **Consensus Mechanism**: Understand the process of reaching an agreement in a distributed network, specifically the Proof of Work (PoW) algorithm.
10. **What is genesis block** ?
11. **Flask Basics**: Learn about creating APIs with Flask, routing, and handling HTTP requests (GET, POST, etc.).
12. **Flask CORS**: CORS (Cross-Origin Resource Sharing) is enabled using the flask\_cors package. This allows the web application (running on a different domain or port) to interact with your blockchain API.

**Cryptography (RSA, Hashing, Digital Signatures)**

* **Hash Functions (SHA-256)**: Learn about secure hash functions, particularly SHA-256, which is used in the hash() method to generate unique hashes for each block.
* **RSA Encryption and Digital Signatures**: You are using RSA keys to generate public-private key pairs for users, and digital signatures to sign transactions (self.sign\_transaction()).
  + Learn how public-key cryptography works, how digital signatures verify the authenticity and integrity of a message, and how to implement it in Python using cryptography.hazmat.
* **Serialization of Keys**: Converting keys into a transferable format (e.g., PEM) using serialization for public and private keys.

**Distributed Systems and Consensus**

* **Node Registration**: Understand how nodes (other participants in the network) are registered and how they communicate with each other (register\_node()).
* **Chain Synchronization**: The resolve\_conflicts() method deals with ensuring that nodes agree on the same version of the blockchain, choosing the longest valid chain.
* **Consensus Algorithms**: Learn about the consensus mechanism used (e.g., Proof of Work in this case).

**HTTP/RESTful API**

* **REST APIs**: You are interacting with the blockchain using API endpoints (/mine, /add\_funds, /transactions/new). Understanding how REST APIs work is key to building and using this project.
* **HTTP Methods**: Learn about the different HTTP request methods (GET, POST) and how to handle them in Flask.
* **JSON Data Handling**: How data is passed via JSON in requests, and how Flask parses and responds with JSON (request.get\_json() and jsonify()).

**Error Handling and Debugging**

* **CORS Errors**: You are facing a CORS-related issue (Cross-Origin Resource Sharing), which prevents your frontend (served from one origin) from accessing your backend API. Learn how to handle CORS properly in Flask and how to troubleshoot issues like the one you're encountering.
* **Response Handling**: Understand how responses are generated in Flask (jsonify), HTTP status codes (200, 400, 500), and how they affect client-side behavior.

**Frontend-Backend Interaction**

* **JavaScript Fetch API**: Your frontend is using JavaScript to make HTTP requests to your blockchain backend. Understanding how the Fetch API works (including error handling) will help you debug client-side issues.
* **Cross-Origin Requests**: How browsers enforce security policies for making cross-origin requests and how to solve CORS issues when working with separate front-end and back-end services.

**Blockchain Concepts in Code**

* **Proof of Work**: The function proof\_of\_work() tries to find a valid "proof" for the last block by solving a computational puzzle. Learn the concept of PoW and how it prevents spam and secures the network.
* **Transaction Signing and Verification**: Users sign transactions with their private key, and the blockchain verifies the signature using their public key (verify\_signature()). Learn why this is important for ensuring the integrity and authenticity of transactions.
* **Mining Rewards**: When a block is mined, the miner (e.g., 'Alice' in this case) receives a reward in the form of cryptocurrency.

**Python Programming**

* **Classes and Objects**: Understanding object-oriented programming in Python is essential for understanding how Blockchain, Block, and Transaction are implemented as Python classes.
* **Error Handling**: You should be familiar with Python's try-except blocks, which are used in various parts of the code to handle potential exceptions.
* **Using External Libraries**: You are using third-party libraries like cryptography, requests, and flask. It is important to know how to install, import, and use such libraries in your projects.

**Concurrency and Node Communication**

* **Requests Library**: The requests module is used for node-to-node communication. Understanding how different nodes communicate and sync with each other (e.g., in resolve\_conflicts()).
* **Multithreading/Async Processing**: Though your project may not yet involve concurrent processing, understanding asynchronous or multithreaded execution can help scale the project when multiple users or nodes are interacting simultaneously.