**Project Title:** Basic Token (ERC-20)

**Expanded Description:**

This project involves the creation of a basic cryptocurrency token that adheres to the ERC-20 standard. ERC-20 is the most widely used technical standard for fungible tokens on the Ethereum blockchain.

Your token will have the following core features:

* **Fixed Supply:** A predetermined total amount of tokens that cannot be increased.
* **Transfer Functionality:** The ability for token holders to send tokens to other addresses.
* **Approval Mechanisms:** Allowances for third-party accounts (like decentralized exchanges) to manage a specified amount of a user's tokens on their behalf.

**Project Requirements:**

1. **Token Details:**
   * **Name:** A unique name for your token (e.g., "MyToken").
   * **Symbol:** A short ticker symbol (e.g., "MYT").
   * **Decimals:** The number of decimal places the token can be divided into (often 18).
   * **Total Supply:** The fixed number of tokens to be created.
2. **Functions:**
   * totalSupply() : Returns the total token supply.
   * balanceOf(address \_owner) : Returns the token balance of a specific address.
   * transfer(address \_to, uint256 \_value) : Transfers tokens from the sender to a recipient.
   * approve(address \_spender, uint256 \_value) : Approves a spender to spend a specific amount of the sender's tokens.
   * transferFrom(address \_from, address \_to, uint256 \_value) : Transfers tokens from one address to another (on behalf of the sender, after prior approval).
3. **Events:**
   * Transfer(address indexed \_from, address indexed \_to, uint256 \_value): Emitted when tokens are transferred.
   * Approval(address indexed \_owner, address indexed \_spender, uint256 \_value): Emitted when a spender is approved to spend tokens.

**Technical Resources Needed:**

* **IDE (Integrated Development Environment):**
  + Remix: An online Solidity IDE perfect for beginners. It's easy to use and allows you to write, compile, and deploy your contract directly in your browser.
  + Visual Studio Code (with Solidity extensions): A more robust IDE for local development, suitable if you're comfortable with code editors.
* **Ethereum Wallet:** MetaMask is a popular browser extension wallet that integrates well with Remix. You'll need it to interact with your deployed contract.
* **Ethereum Client:** You can use a test network (e.g., Goerli, Sepolia) to deploy your contract without spending real Ether.
* **OpenZeppelin Contracts:** This library provides secure and audited implementations of ERC-20 and other common token standards, saving you development time and reducing the risk of vulnerabilities.

**Timeline (Estimated):**

* **Learning Solidity Basics & ERC-20:** 1-2 days
* **Contract Implementation (with OpenZeppelin):** 1-2 days
* **Testing and Deployment on a Testnet:** 1 day
* **Total:** 3-5 days (depending on prior Solidity experience)

**Additional Tips:**

* **Start Simple:** Begin with a very basic implementation. Once you understand the core concepts, you can gradually add complexity.
* **Use OpenZeppelin:** Leverage OpenZeppelin's ERC-20 implementation to ensure your contract is secure and follows best practices.
* **Test Thoroughly:** Write unit tests to verify your contract's functionality before deploying it to a testnet.

Let me know when you're ready for the next project breakdown!