Project Title: **Hyperlocal Community Exchange Platform**

Problem Statement:

Hyperlocal communities often face issues such as inefficient resource sharing, lack of trust, and limited economic opportunities. This project aims to develop a decentralized community exchange platform to solve these problems and foster collaboration within hyperlocal communities.

Technologies:

- Ethereum blockchain
- Solidity smart contracts
- Truffle framework
- Web3.js for interacting with the blockchain
- React.js for the front-end development
- MetaMask for wallet integration

Project Outline:

- Smart Contract Development:
- Develop smart contracts for the community exchange platform.
- Define data structures for user profiles, listings, transactions, and reputation scores.
- Implement functions for creating listings, making offers, accepting/rejecting offers, and managing transactions.

Blockchain Integration:

- Set up a local development blockchain environment using tools like Ganache.
- Deploy smart contracts on the Ethereum network.
- Establish connections with the Ethereum network using Web3.js.
- Integrate MetaMask to enable users to interact with the platform using their Ethereum wallets.

```
// Example smart contract code

pragma solidity ^0.8.0;

contract CommunityExchangePlatform {
  struct Listing {
    address owner;
    string title;
    uint price;
    bool isSold;
  }

  struct Transaction {
    address buyer;
    address seller;
    uint amount;
    bool isCompleted;
  }

  mapping(uint => Listing) public listings;
  mapping(uint => Transaction) public transactions;
  uint public listingCounter;
  uint public transactionCounter;

  event ListingCreated(uint listingId, address owner, string title, uint price);
```

```
  event OfferMade(uint transactionId, address buyer, address seller, uint amount);
  event TransactionCompleted(uint transactionId, address buyer, address seller, uint amount);

  function createListing(string memory _title, uint _price) external {
    listings[listingCounter] = Listing(msg.sender, _title, _price, false);
    emit ListingCreated(listingCounter, msg.sender, _title, _price);
    listingCounter++;
  }

  function makeOffer(uint _listingId, uint _amount) external payable {
    require(!listings[_listingId].isSold, "Listing is already sold");
    require(_amount >= listings[_listingId].price, "Offered amount is less than the listing price");

    transactions[transactionCounter] = Transaction(msg.sender, listings[_listingId].owner, _amount, false);
    emit OfferMade(transactionCounter, msg.sender, listings[_listingId].owner, _amount);
    transactionCounter++;
  }

  function completeTransaction(uint _transactionId) external payable {
    require(!transactions[_transactionId].isCompleted, "Transaction is already completed");
    require(msg.sender == transactions[_transactionId].seller, "Only the seller can complete the transaction");
```

```
payable(transactions[_transactionId].seller).transfer(transactions[_transactionId]
.amount);
    transactions[_transactionId].isCompleted = true;
    listings[_transactionId].isSold = true;

    emit TransactionCompleted(_transactionId,
transactions[_transactionId].buyer, transactions[_transactionId].seller,
transactions[_transactionId].amount);
  }
}
```

JavaScript: -

```
// Example code to interact with the blockchain using Web3.js

import Web3 from 'web3';
import contractAbi from './contractAbi.json';

const web3 = new Web3(Web3.givenProvider);
const contractAddress = 'YOUR_CONTRACT_ADDRESS';
const
```