Project Report

Objective

The primary goal was to create a Decentralized Marketplace that uses Ethereum smart contracts for item listing and purchasing, providing users with a decentralized platform to conduct transactions without traditional centralized systems. This approach enhances security and ensures transparency in the buying and selling process. For the marketplace, I have chosen a theme of selling Lego's and named my application 'Lego Land'.

Project Design

The application consists of three main components:

1. **Smart Contract:** Written in Solidity (using Remix IDE), it handles the business logic on the Ethereum blockchain, including functions for listing items, purchasing, and querying item data.
2. **Frontend:** Developed using Next.js, it provides a robust and scalable user interface for interacting with the smart contract. It allows users to list items, view item details, and purchase items.
3. **Hosting and Deployment:** The frontend is hosted on Netlify, leveraging its CI/CD capabilities for seamless updates. The smart contract is deployed on the Sepolia Ethereum testnet.

Smart Contract

The 'Marketplace' contract includes several key functionalities:

1. **listItem:** Allows a seller to list an item by specifying details such as title, description, price, and an IPFS hash for the image.
2. **buyItem:** Enables purchasing of items where transaction validity checks are performed to ensure security.
3. **getAllItems** and **getItemById:** Provide read access to the listed items' data.

Frontend

The frontend utilizes Next.js, a React framework, which is particularly suited for applications requiring server-side rendering or static site generation. Components like 'buying', 'item_details', and 'listingpage' interact with the smart contract via 'ethers.js', facilitating operations such as connecting to Ethereum wallets, listing new items, and purchasing available items. The 'axios' library is used for handling IPFS image uploads, ensuring that user-generated content is stored on a decentralized network.

Implementation

Technologies & Libraries Used

1. **Solidity:** Smart contract programming language.
2. **Next.js:** React framework used for the frontend, enhancing server-side rendering and static site generation capabilities.
3. **Netlify:** For hosting the frontend.
4. **IPFS:** Decentralized storage solution used for image hosting via Pinata Cloud.
5. **Ethers.js:** JavaScript library used for interacting with Ethereum blockchain.
6. **Axios:** JavaScript library used for making HTTP requests to external services like IPFS.

Deployment

1. The smart contract is deployed on the Sepolia testnet using Remix IDE.
2. The frontend is deployed on Netlify with HTTPS enabled to secure the connection.

Challenges and Solutions

Challenge 1: Ethereum Wallet Integration
Problem: Users experienced difficulties in connecting their Ethereum wallets to the DApp.
Solution: Improved the wallet connection mechanism in the frontend with robust error handling and clear user feedback.

Challenge 2: Asynchronous Operations
Problem: Delays and UI blocks during asynchronous operations like blockchain transactions and IPFS uploads.
Solution: Implemented asynchronous JavaScript features and state management to handle operations efficiently and update the UI accordingly.

Challenge 3: Secure and Reliable Image Hosting
Problem: Needed a decentralized and reliable solution for storing item images.
Solution: Utilized IPFS through Pinata for decentralized image storage, ensuring durability and accessibility without relying on central servers. The ‘axios’ library was instrumental in managing these uploads efficiently.

Challenge 4: Handling of Solidity Events
Problem: Managing and responding to Solidity events efficiently to update the frontend in real time.
Solution: Implemented event listeners in the frontend that react to emitted events from the smart contract, ensuring the UI is immediately updated when transaction states change on the blockchain.

Challenge 5: Integrating Contract Data with Frontend
Problem: Difficulty in efficiently integrating live data from the Ethereum smart contract into the Next.js frontend, especially given the asynchronous nature of blockchain transactions.
Solution: Leveraged the powerful capabilities of Next.js and 'ethers.js' to fetch, subscribe to, and manage state changes from the blockchain in real-time. This integration was essential for the seamless operation of the DApp, allowing users to interact with the blockchain without needing to refresh the page.

Challenge 6: User Interface Responsiveness and Accessibility
Problem: Ensuring the DApp is responsive and accessible across different devices and platforms.
Solution: Applied responsive design principles and accessibility standards throughout the frontend development to ensure a seamless and inclusive user experience for all users.

Github Link: https://github.com/sairamp98/Blockchain_decentralized_marketplace_sepoliaeth
Marketplace Link: https://chipper-mermaid-b6aa78.netlify.app/welcome

Limitations of my marketplace:

1. The marketplace only allows you to list and buy items if you have a Metamask extension in your browser.
2. If you are switching between multiple metamask accounts for a purchase, make sure that the accounts are connected to your wallet.
3. The application is very basic in terms of functionality as it does not have a separate page for the user to check his purchases or listings.
4. Due to the usage of Pinata and uploading high quality images, the load times could be longer than usual. Also, the usage of flexboxes and tailwind styling is taking longer load times in machines that have low processing power. (I have observed this issue in a machine that had Intel i5 3rd generation)
5. For the first time users, you could encounter blank page for a split second mainly due to retrieval from Pinata. In case of a persisting blank page, it is always better to refresh the page.
6. If you are switching your wallet after the metamask check button in homepage is clicked, refresh the page again to check the connection of the newly switched wallet.

Conclusion

The project successfully demonstrated the feasibility and effectiveness of a decentralized marketplace built on Ethereum. The DApp provides a secure and transparent platform for users to list and purchase items, showcasing the benefits of blockchain technology in real-world applications. Future enhancements could include features like auction mechanisms, time-limited listings, and more extensive buyer and seller interactions to further enrich the user experience.