99starzXapeDAO Updates

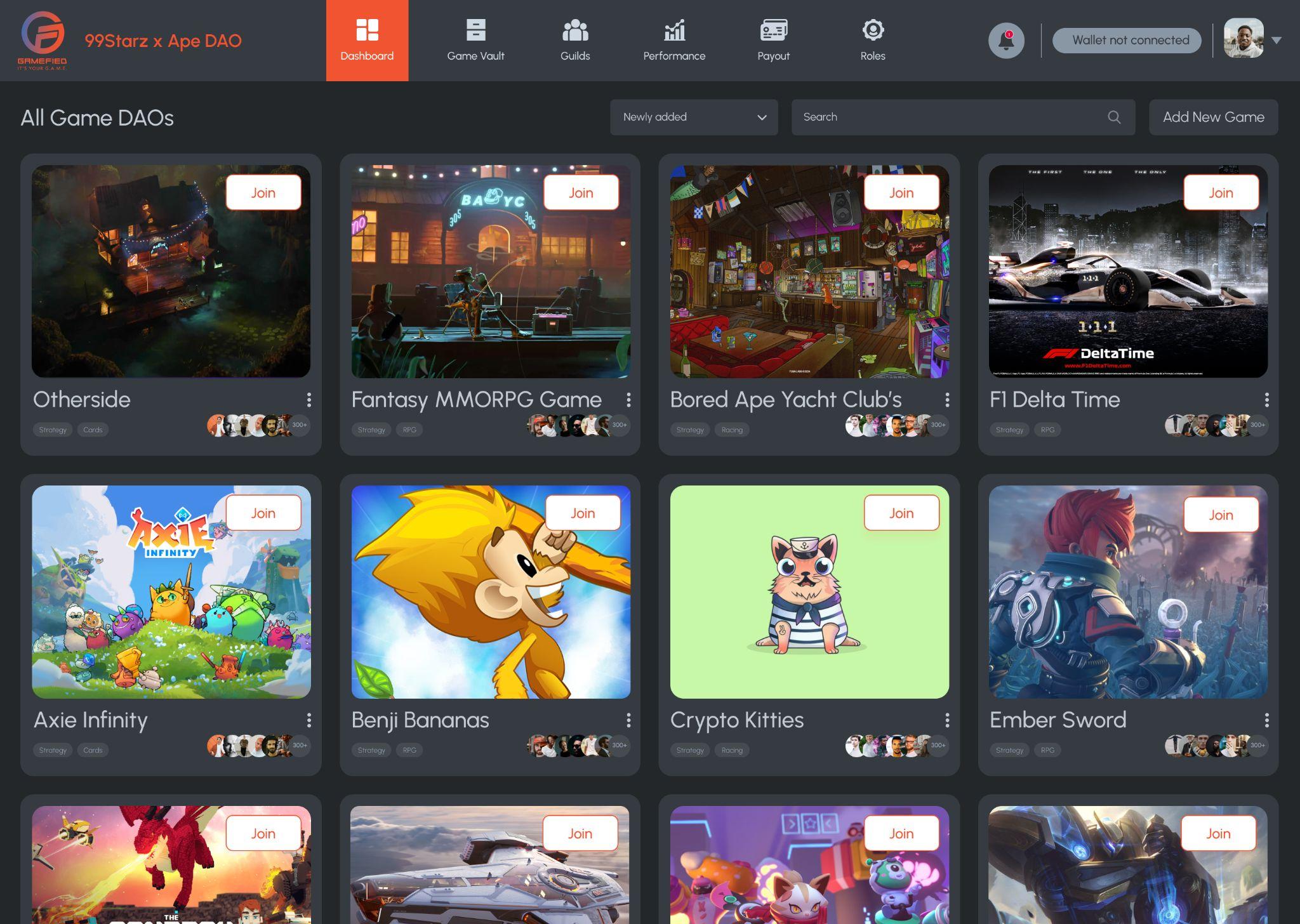
The current progress of the 99starzXapeDAO project specifically focuses on the development of core smart contracts and the UI/UX. Below is a detailed summary of the work completed and ongoing tasks.

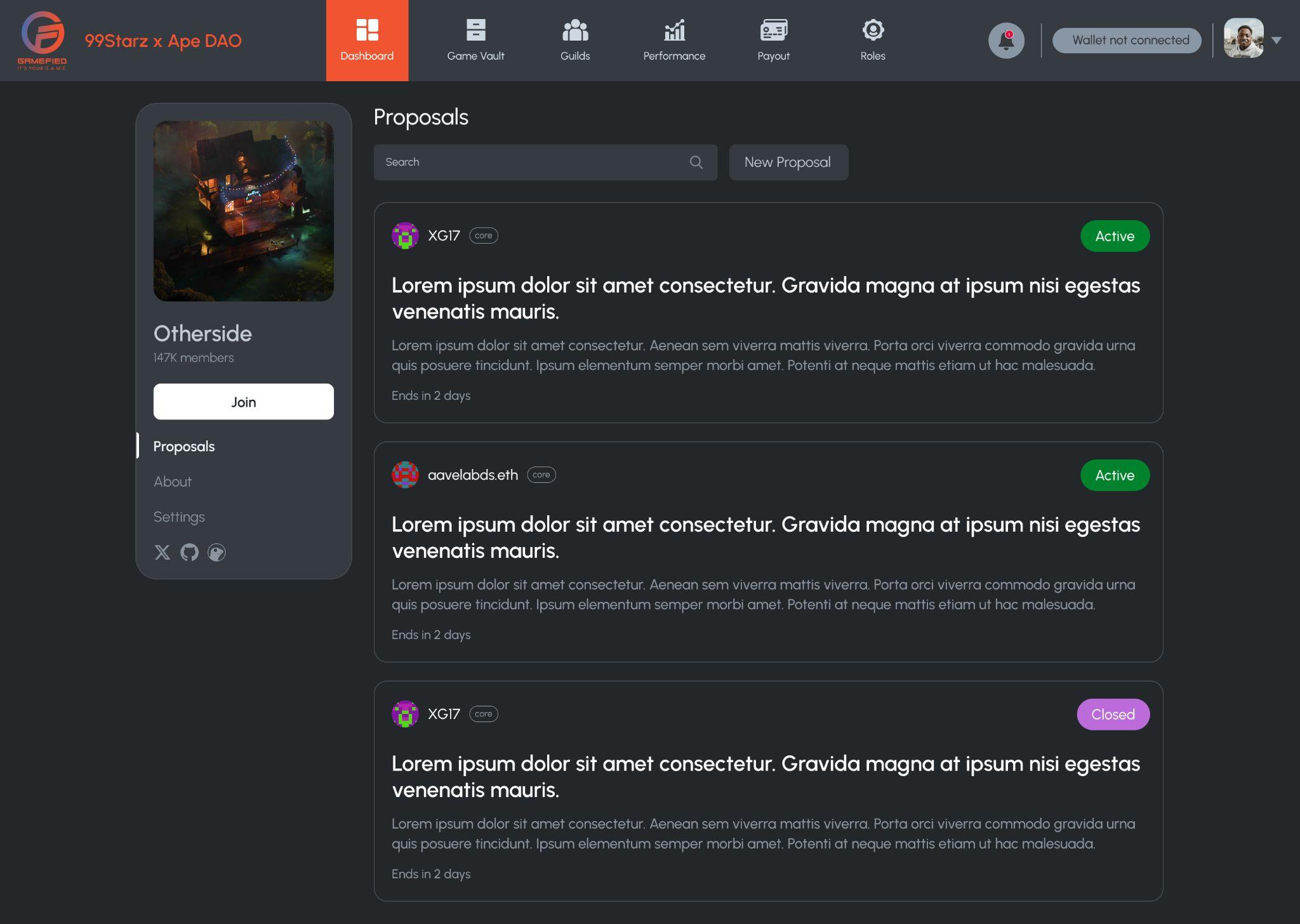
### **UI/UX Development**

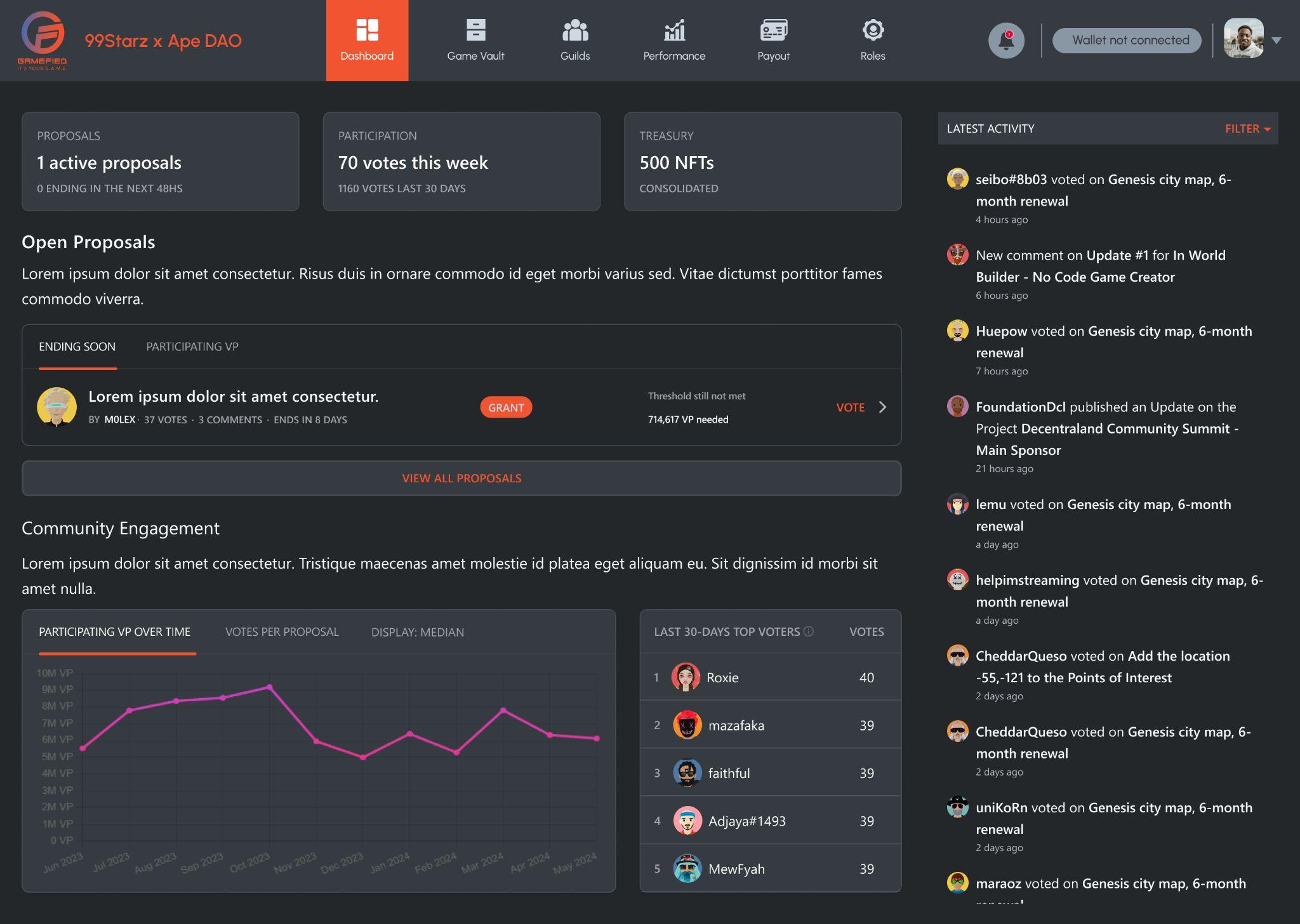
Our design team has been actively working on the user interface and user experience aspects of the project to ensure an intuitive and seamless experience for end-users.

**Key Activities:**

* Wireframing and prototyping of the main DAO and sub-DAO interfaces
* User flow design for proposal creation, voting, staking, and reward distribution
* High-fidelity mockups for key screens and interactions
* Usability testing and feedback incorporation







### 

### **Smart Contract Development**

#### **1. Main DAO Contract**

**Status**: In Progress

We have designed and implemented the core decentralized autonomous organization (DAO) contract, which handles key functions such as membership management, proposal creation, voting, and execution.

| struct Proposal {  uint256 id;  string description;  uint256 votesFor;  uint256 votesAgainst;  uint256 endTime;  bool executed;  mapping(address => bool) voted;  }   address public owner;  uint256 public proposalCount;  mapping(uint256 => Proposal) public proposals;  mapping(address => bool) public members;   modifier onlyOwner() {  require(msg.sender == owner, "Not the owner");  \_;  }   modifier onlyMember() {  require(members[msg.sender], "Not a member");  \_;  }   event NewProposal(uint256 id, string description, uint256 endTime);  event Voted(uint256 id, address voter, bool support);  event Executed(uint256 id); |
| --- |

**Features:**

* Membership addition and removal
* Proposal creation with description and duration
* Voting mechanism for proposals
* Execution of approved proposals

#### **2. Sub-DAO Proxy Contracts**

**Status**: In Progress

Proxy contracts have been developed to facilitate the functionality of sub-DAOs. These contracts delegate calls to an implementation contract, allowing for flexibility and upgradability.

| contract SubDAOProxy is ERC1967Upgrade, Initializable {   function updateImplementation(address \_newImplementation) external onlyAdmin {  \_upgradeTo(\_newImplementation);  }   function \_delegate(address implementation) internal virtual {  assembly {  // Copy msg.data. We take full control of memory in this inline assembly  // block because it will not return to Solidity code. We overwrite the  // Solidity scratch pad at memory position 0.  calldatacopy(0, 0, calldatasize())   // Call the implementation.  // out and outsize are 0 because we don't know the size yet.  let result := delegatecall(gas(), implementation, 0, calldatasize(), 0, 0)   // Copy the returned data.  returndatacopy(0, 0, returndatasize())   switch result  // delegatecall returns 0 on error.  case 0 { revert(0, returndatasize()) }  default { return(0, returndatasize()) }  }  } |
| --- |

**Features:**

* Implementation address management
* Fallback function for delegated calls
* Owner management for implementation updates

#### **3. Wrapper 4097 Contract**

**Status**: In Progress

The Wrapper 4097 contract is designed to enable NFT rentals. This contract allows for the wrapping and unwrapping of NFTs, as well as managing rental agreements.

**Features:**

* NFT wrapping and unwrapping
* Rental agreements with duration
* Transfer restrictions during rental periods