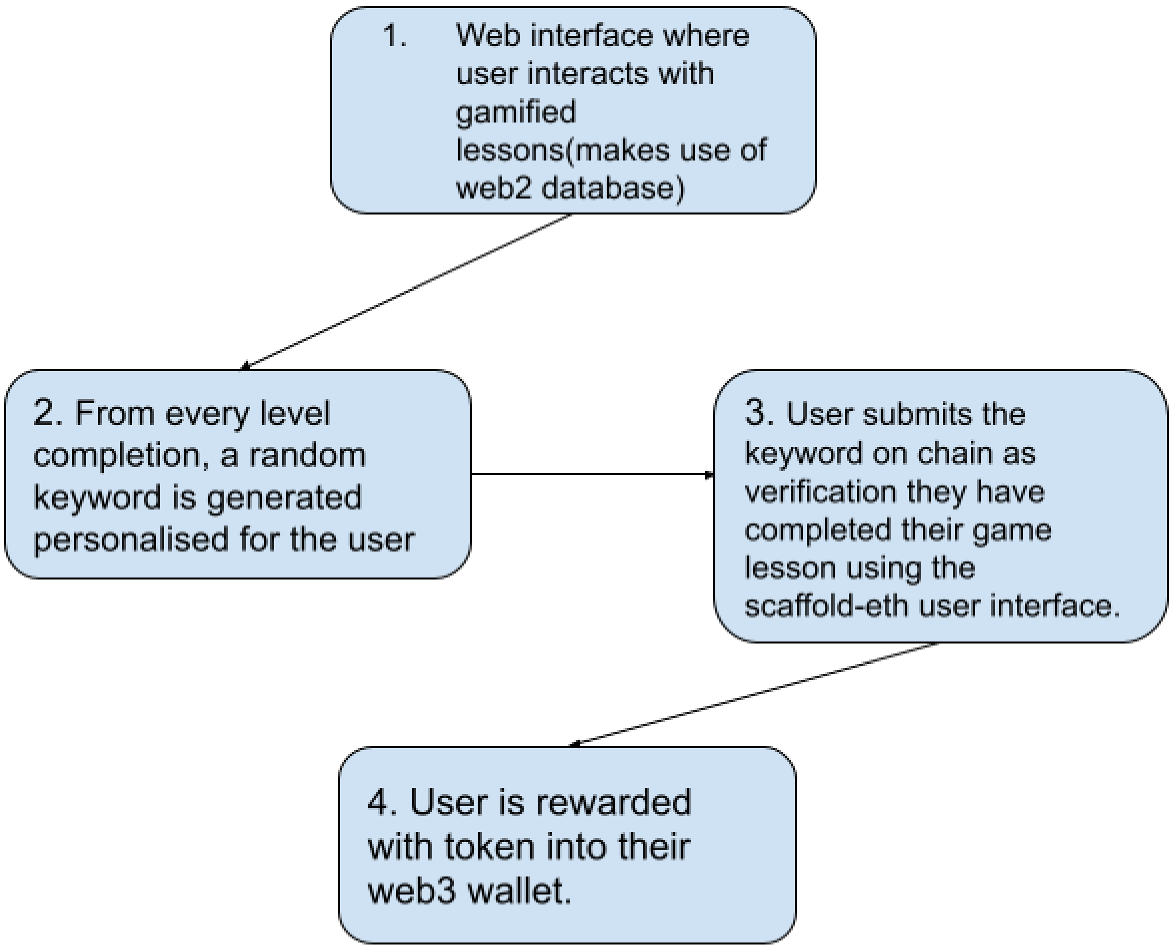
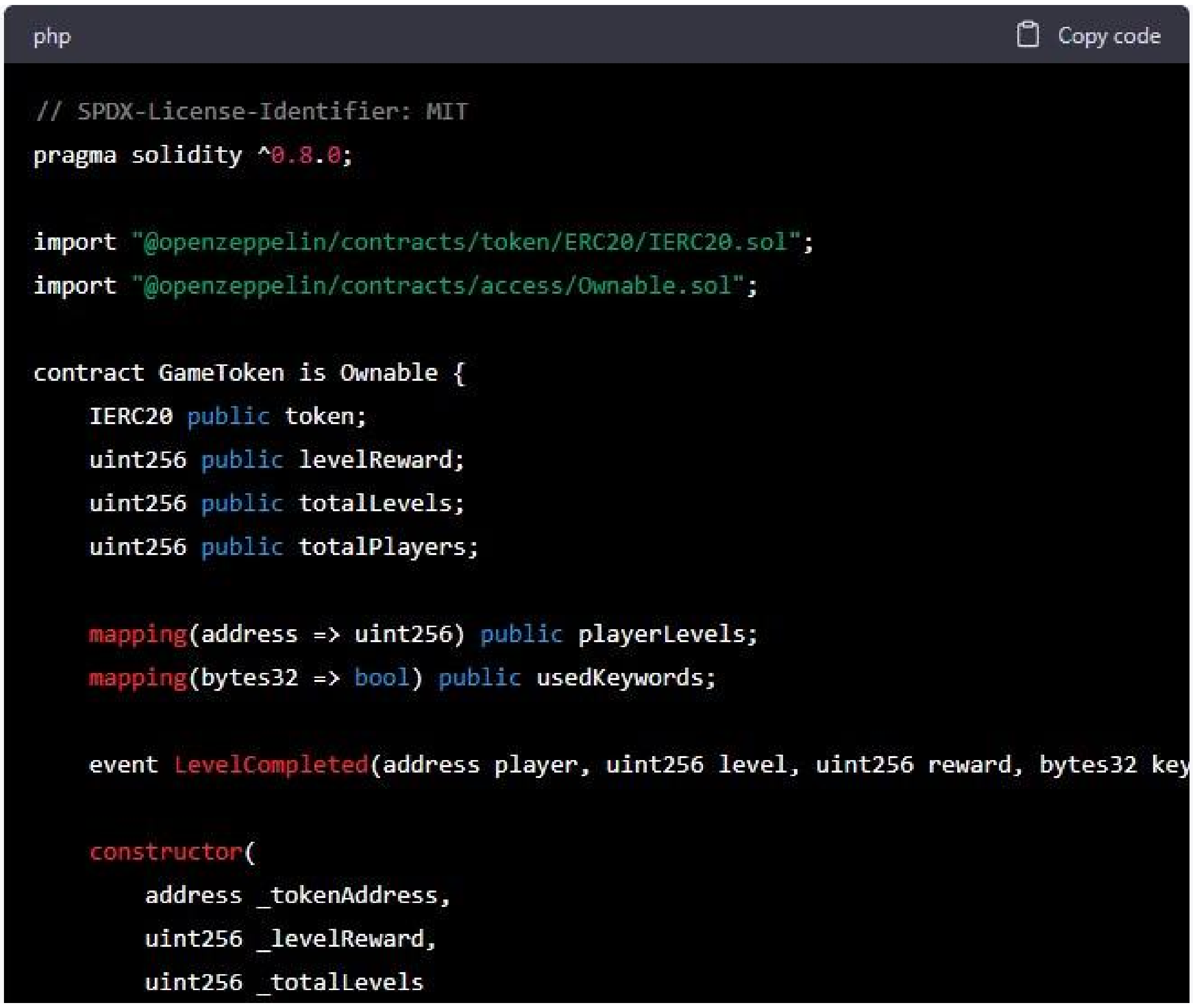
FOXTROT PSEUDOCODE GRAPH


1. Web interface where user interacts with gamified lessons(makes use of web2 database)
2. From every level completion, a random keyword is generated personalised for the user
3. User submits the keyword on chain as verification they have completed their game lesson using the scaffold-eth user interface.
4. User is rewarded with token into their web3 wallet.

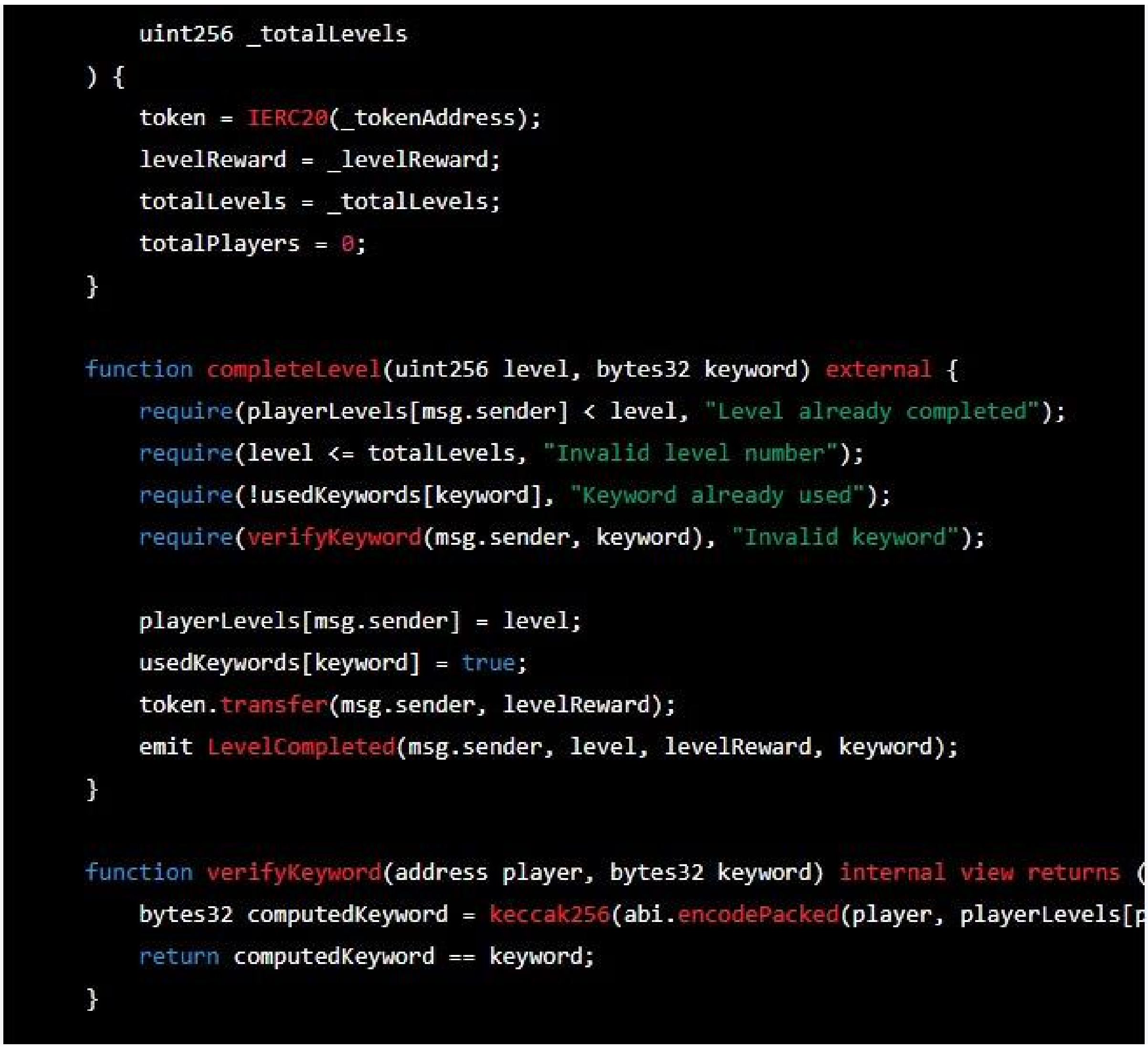
Functions needed:

- Mapping function
- LevelCompleted function
- VerifyKeyword function
- WithdrawTokens function

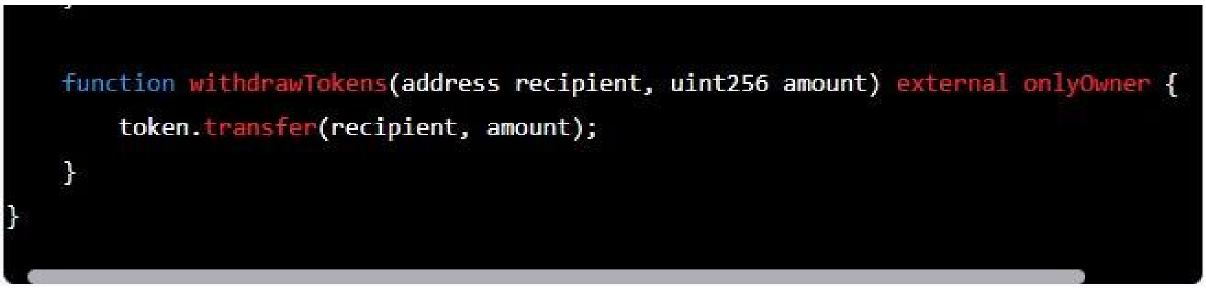
Sample Solidity Code-Submit generated keyword on Scaffold-Eth and Reward Token:



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract GameToken is Ownable {
    IERC20 public token;
    uint256 public levelReward;
    uint256 public totalLevels;
    uint256 public totalPlayers;

    mapping(address => uint256) public playerLevels;
    mapping(bytes32 => bool) public usedKeywords;

    event LevelCompleted(address player, uint256 level, uint256 reward, bytes32 key

    constructor(
        address _tokenAddress,
        uint256 _levelReward,
        uint256 _totalLevels
```

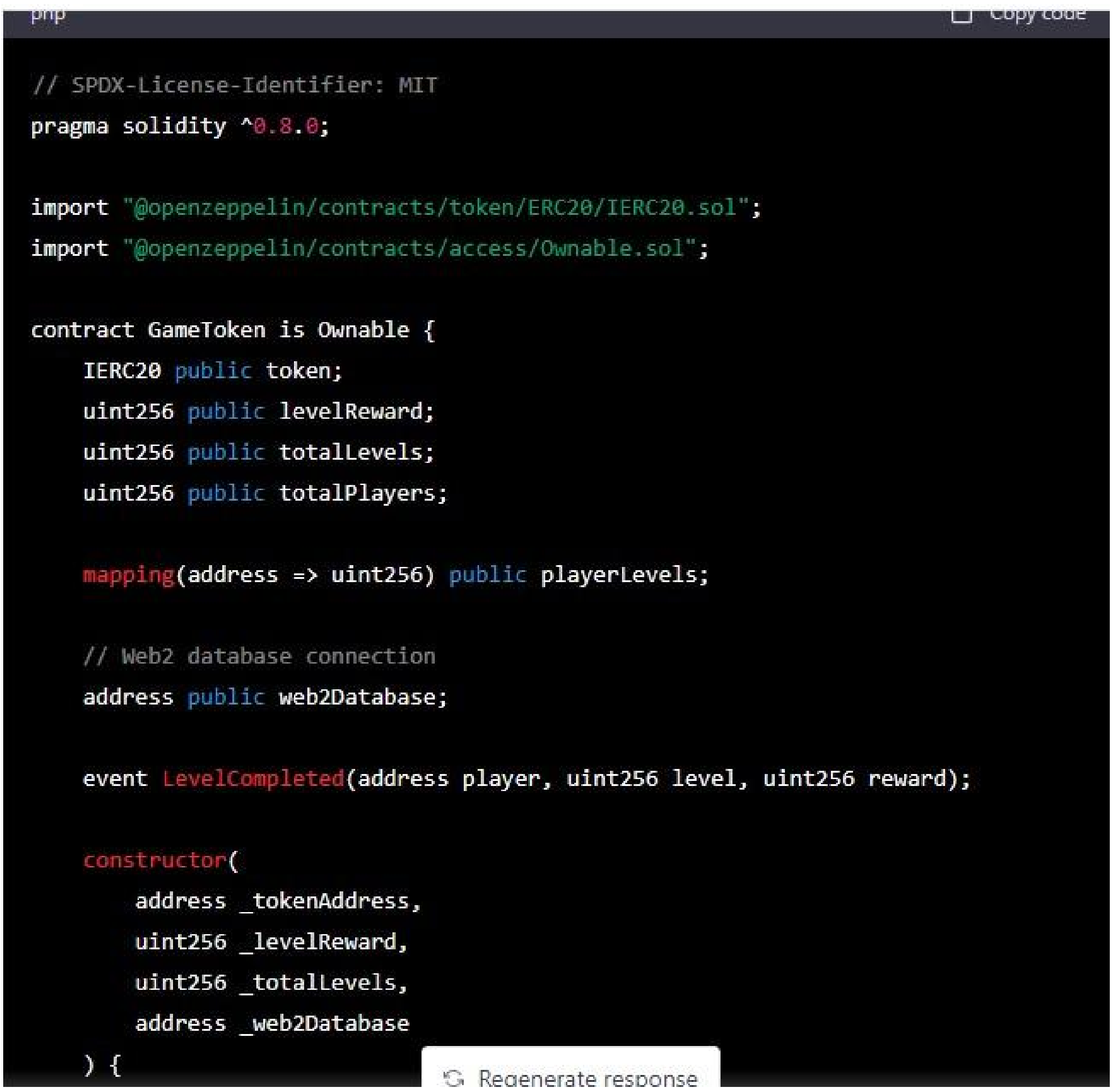


```
        uint256 _totalLevels
    ) {
        token = IERC20(_tokenAddress);
        levelReward = _levelReward;
        totalLevels = _totalLevels;
        totalPlayers = 0;
    }

    function completeLevel(uint256 level, bytes32 keyword) external {
        require(playerLevels[msg.sender] < level, "Level already completed");
        require(level <= totalLevels, "Invalid level number");
        require(!usedKeywords[keyword], "Keyword already used");
        require(verifyKeyword(msg.sender, keyword), "Invalid keyword");

        playerLevels[msg.sender] = level;
        usedKeywords[keyword] = true;
        token.transfer(msg.sender, levelReward);
        emit LevelCompleted(msg.sender, level, levelReward, keyword);
    }

    function verifyKeyword(address player, bytes32 keyword) internal view returns (
        bytes32 computedKeyword = keccak256(abi.encodePacked(player, playerLevels[p
        return computedKeyword == keyword;
    }
```

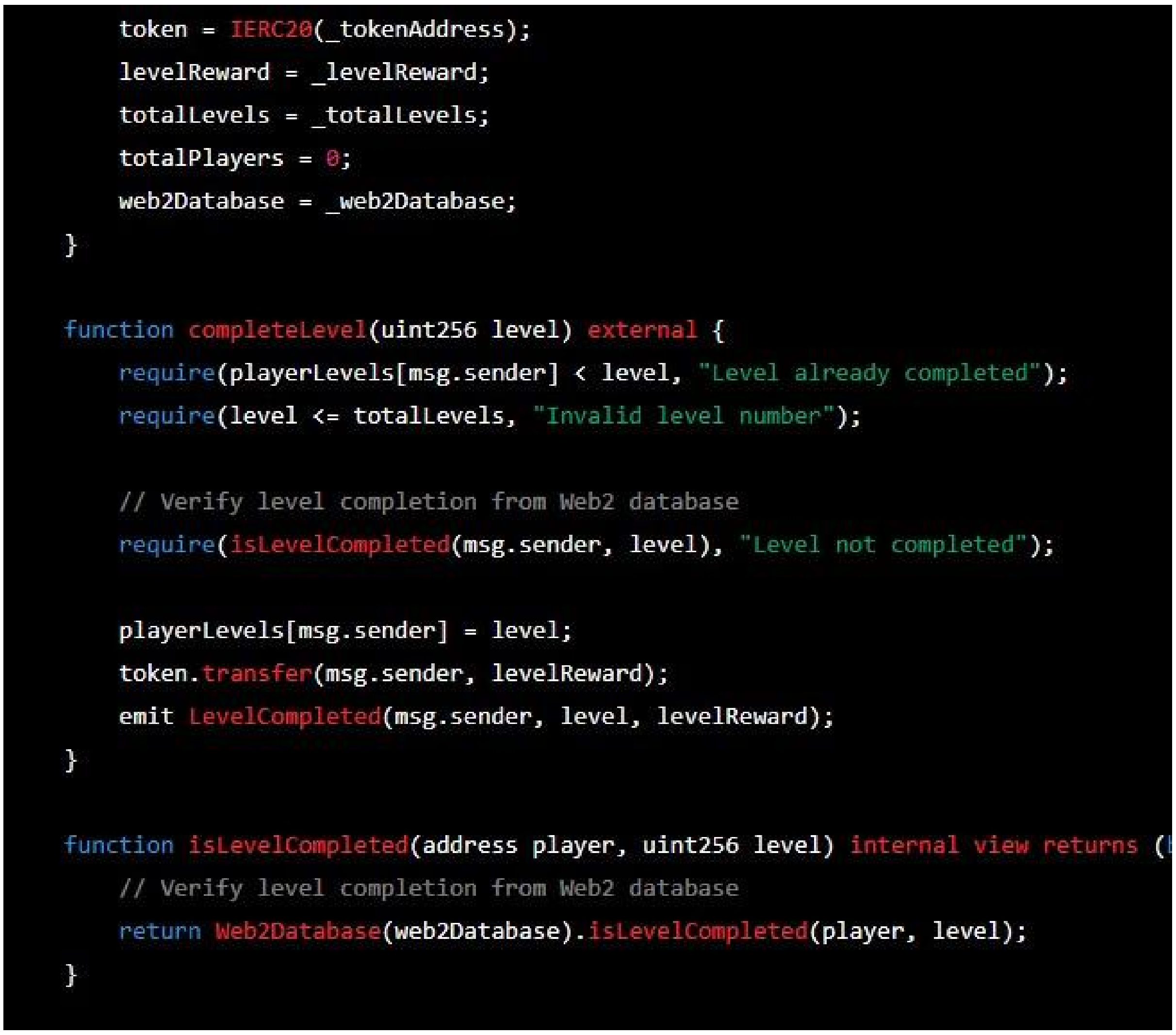


```
    function withdrawTokens(address recipient, uint256 amount) external onlyOwner {
        token.transfer(recipient, amount);
    }
}
```

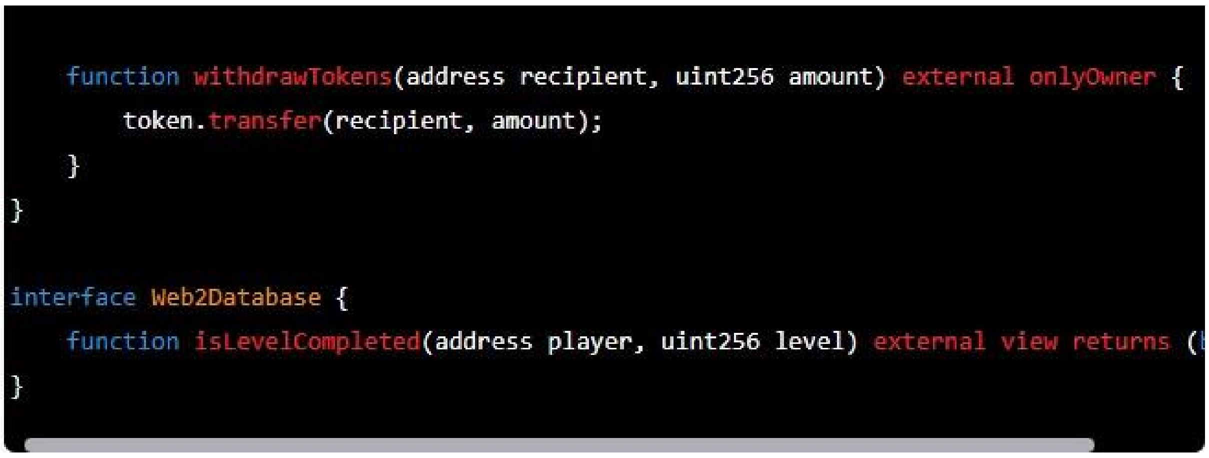
Solidity code example- link web2 database to smart contract and web3 ethereum wallet to verify that wallet address has completed a level on the game to be rewarded with a token:



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract GameToken is Ownable {
    IERC20 public token;
    uint256 public levelReward;
    uint256 public totalLevels;
    uint256 public totalPlayers;

    mapping(address => uint256) public playerLevels;

    // Web2 database connection
    address public web2Database;

    event LevelCompleted(address player, uint256 level, uint256 reward);

    constructor(
        address _tokenAddress,
        uint256 _levelReward,
        uint256 _totalLevels,
        address _web2Database
    ) {
```

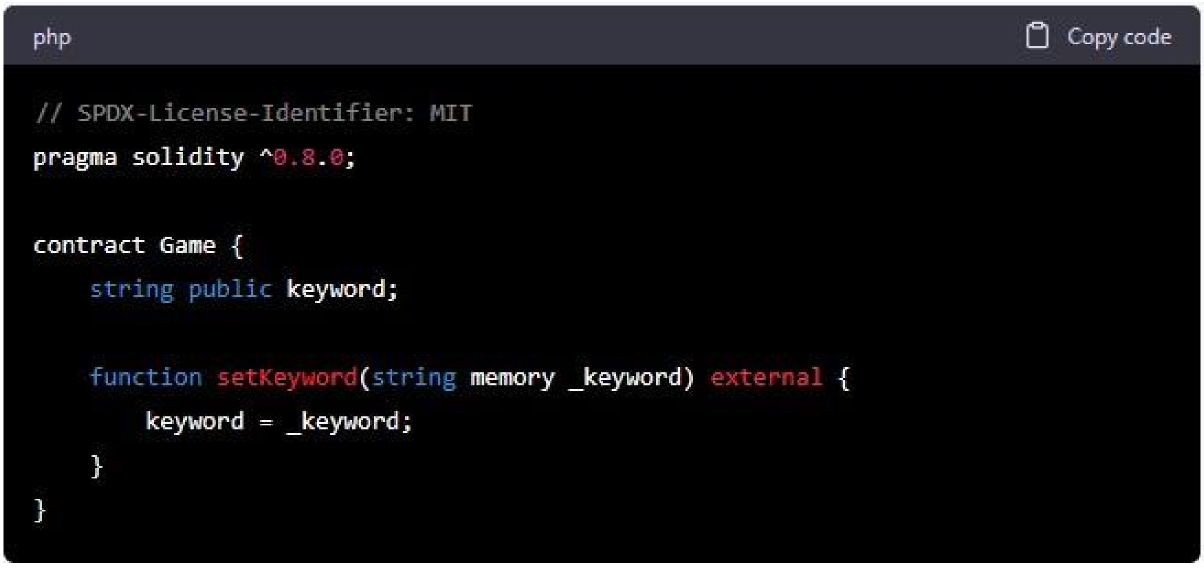


```
        token = IERC20(_tokenAddress);
        levelReward = _levelReward;
        totalLevels = _totalLevels;
        totalPlayers = 0;
        web2Database = _web2Database;
    }

    function completeLevel(uint256 level) external {
        require(playerLevels[msg.sender] < level, "Level already completed");
        require(level <= totalLevels, "Invalid level number");

        // Verify level completion from Web2 database
        require(isLevelCompleted(msg.sender, level), "Level not completed");

        playerLevels[msg.sender] = level;
        token.transfer(msg.sender, levelReward);
        emit LevelCompleted(msg.sender, level, levelReward);
    }

    function isLevelCompleted(address player, uint256 level) internal view returns (
        // Verify level completion from Web2 database
        return Web2Database(web2Database).isLevelCompleted(player, level);
    }
```



```

    function withdrawTokens(address recipient, uint256 amount) external onlyOwner {
        token.transfer(recipient, amount);
    }
}

interface Web2Database {
    function isLevelCompleted(address player, uint256 level) external view returns (
}
```

Sample Solidity code(function)- accepts a keyword input from Scaffold-Eth user interface:


php Copy code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Game {
    string public keyword;

    function setKeyword(string memory _keyword) external {
        keyword = _keyword;
    }
}
```