Dodatok A

Harmonogram práce

A.1 Zimný semester

1st-4th week	Konzultácie & hľadanie potrebného obsahu
5th week	Analyzovanie on-chain riešení
6th week	Analyzovanie off-chain riešení
7th week	Práca na úvode a analyzovanie hlavných problémov
8th week	Nájdenie vhodných hier
9th-10th week	Prepájanie stránky a hry
11th week	Funkčný model backendu a jednej hry
12th week	Finalizácia dokumentu na odovzdanie

A.2 Letný semester

1st-2nd week	Programovanie smart kontraktu a backendu
3rd-4th week	Zapajánie jednotlivých hier do pozície smart kontraktov
5th-6th week	Budovanie a testovanie funkcionality smart konraktu a hier
7th-8th week	Front-end webovej stránky
10th week	Konzultácie
11th-12th week	Finalizácia dokumentu na odovzdanie

Dodatok B

Obsah digitálneho média

Evidenčné číslo práce v informačnom systéme: FIIT-100241-97019

Obsah digitálnej časti práce (archív ZIP):

Folder	Contents
/master	front-end súbory
/.next	
/blockchain	binárne, abi, .sol súbory
/components	react komponenty
/css	css vzhľad
/models	databázové modely
/node_modules	knižnice
/pages	zdrojové kódy jednotlivých stránok
/public	
/styles	css štýly
/utils	databázové súbory
next.config	konfiguračné súbory

/hardhat	základy rinkeby web3
/artifacts	kompilované .sol súbory
/cache	
/contracts	binárne, abi, .sol súbory
/node_modules	knižnice
/scripts	
/test	testy na kontrakt
hardhat.config	konfiguračné súbory
startServer.cmd	spustenie front-end a back-end serveru
printLibraries.cmd	vytvorenie libraries.txt
test.cmd	spustenie hardhat testu
/praca-pdf	pdf verzia záverečnej práce
/praca.pdf	pdf hlavná časť záverečnej práce
/prilohy.pdf	pdf textové prílohy záverečnej práce

Názov odovzdaného archívu: BP_prilohy.zip.

Listing B.1: Smart kontrakt

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.11;

contract GameChannel {
    address public owner;
    address public winner;
    address payable p2_address;
    address payable inactive;
    address payable waiting;
    uint256 public constant TIMEOUT = 1 minutes;
```

```
uint public bet;
uint256 public timeout;

constructor() payable {
    require(msg.value >= 0.02 ether);
    bet = msg.value;
    owner = payable(msg.sender);
}

function returnWinner() public
    view
    returns (address){
        return winner;
    }

function getMessageHash(
    uint[] calldata _num
) public pure returns (bytes32) {
    return keccak256(abi.encode(_num));
    }

function verify(uint[] calldata _num, bytes32 _board,
   bytes memory _signature1, bytes memory _signature2)
   public returns (bool){
    require(recoverSigner(_board, _signature1) == owner
        , "EOAVerify:␣Signed␣mismatch");
    require(recoverSigner(_board, _signature2) ==
    p2_address, "EOAVerify:␣Signed␣mismatch");
require(getMessageHash(_num) == _board, "Sent␣fake␣
    board");
require(winner == address(0));
if (      (_num[0] == 0x00 && _num[1] == 0x00 && _num
    [2] == 0x00 ) ||
            (_num[3] == 0x00 && _num[4] == 0x00 && _num
                [5] == 0x00 ) ||
            (_num[6] == 0x00 && _num[7] == 0x00 && _num
                [8] == 0x00 ) ||
            (_num[0] == 0x00 && _num[4] == 0x00 && _num
                [8] == 0x00 ) ||
            (_num[6] == 0x00 && _num[4] == 0x00 && _num
                [2] == 0x00 ) ||
            (_num[0] == 0x00 && _num[3] == 0x00 && _num
                [6] == 0x00 ) ||
            (_num[1] == 0x00 && _num[4] == 0x00 && _num
                [7] == 0x00 ) ||
            (_num[2] == 0x00 && _num[5] == 0x00 && _num
                [8] == 0x00 )
){
      winner = owner;
      uint amount = address(this).balance;
      (bool success, ) = winner.call{value: amount}("
          ");
      require(success, "Failed␣to␣send␣Ether");
}else if(
                (_num[0] == 0x02 && _num[1] == 0x02 && _num
                    [2] == 0x02 ) ||
                (_num[3] == 0x02 && _num[4] == 0x02 && _num
                    [5] == 0x02 ) ||
                (_num[6] == 0x02 && _num[7] == 0x02 && _num
                    [8] == 0x02 ) ||
                (_num[0] == 0x02 && _num[4] == 0x02 && _num
                    [8] == 0x02 ) ||
                (_num[6] == 0x02 && _num[4] == 0x02 && _num
                    [2] == 0x02 ) ||
                (_num[0] == 0x02 && _num[3] == 0x02 && _num
                    [6] == 0x02 ) ||
                (_num[1] == 0x02 && _num[4] == 0x02 && _num
                    [7] == 0x02 ) ||
                (_num[2] == 0x02 && _num[5] == 0x02 && _num
                    [8] == 0x02 )
        ){
            winner = p2_address;
            uint amount = address(this).balance;
            (bool success, ) = winner.call{value: amount}("
                ");
            require(success, "Failed␣to␣send␣Ether");
        }else {
            return false;
        }
        return true;
    }
```

```
function timeoutChallenge(bytes32 _board, bytes32
   _boardBefore, bytes memory _challenger, bytes memory
    _challenged) public returns (bool){
    require(timeout == 0);
    require(winner == address(0));
     if (msg.sender == owner) {
        require(recoverSigner(_board, _challenger) ==
           owner, "EOAVerify:␣Signed␣mismatch");
        require(recoverSigner(_boardBefore, _challenged
           ) == p2_address, "EOAVerify:␣Signed␣mismatch
           ");
        inactive = p2_address;
        waiting = payable(owner);
    } else if (msg.sender == p2_address){
        require(recoverSigner(_board, _challenger) ==
           p2_address, "EOAVerify:␣Signed␣mismatch");
        require(recoverSigner(_boardBefore, _challenged
           ) == owner, "EOAVerify:␣Signed␣mismatch");
        inactive = payable(owner);
        waiting = p2_address;
    }
    timeout = block.timestamp + TIMEOUT;
    return true;
}

function claimTimeout() external {
        require(timeout <= block.timestamp);
        require(waiting != address(0));
        winner = waiting;
        uint amount = address(this).balance;
        (bool success, ) = waiting.call{value: amount}("");
        require(success, "Failed to send Ether");
    }

    function cancelTimeout() public {
        require(inactive == msg.sender);
        require(timeout > block.timestamp);
        inactive = payable(address(0));
        waiting = payable(address(0));
        timeout = 0;
    }

    function recoverSigner(bytes32 _ethSignedMessageHash,
       bytes memory _signature)
        public
        pure
        returns (address){
            (bytes32 r, bytes32 s, uint8 v) =
               splitSignature(_signature);
            return ecrecover(_ethSignedMessageHash, v, r, s
               );
        }
```

```
function splitSignature(bytes memory sig)
    public
    pure
    returns (
            bytes32 r,
            bytes32 s,
            uint8 v
        ){
        require(sig.length == 65, "Invalid␣signature␣
           length");
        assembly {
            r := mload(add(sig, 32))
            s := mload(add(sig, 64))
            v := byte(0, mload(add(sig, 96)))
        }
    }

function getP2() public view returns (address) {
    return (p2_address);
}

function getP1() public view returns (address) {
    return (owner);
}

function join() public payable {
    require(msg.value == bet);
```

```
        require(p2_address == address(0));
        p2_address = payable(msg.sender);
    }

    function _verifyMerkleProof(
        bytes32 root,
        bytes32 leaf,
        bytes32[] memory proof
    )
        public
        pure
        returns (bool)
    {
        bytes32 computedHash = leaf;

        for (uint256 i = 0; i < proof.length; i++) {
        bytes32 proofElement = proof[i];

        if (computedHash <= proofElement) {
            // Hash(current computed hash + current element
                 of the proof)
            computedHash = keccak256(abi.encodePacked(
               computedHash, proofElement));
        } else {
            // Hash(current element of the proof + current
               computed hash)
            computedHash = keccak256(abi.encodePacked(
                proofElement, computedHash));
        }
        }

        // Check if the computed hash (root) is equal to
           the provided root
        return computedHash == root;
    }
}
```

Dodatok C

Používateľská príručka

Pred spustením jednotlivých skriptov je dôležité nainštalovať si *node.js* zo stránky `https://nodejs.org/en/download/` a taktiež testovacie prostredie Ganache zo stránky `https://trufflesuite.com/ganache/`. Práca bola testovaná na node verzii 8.3.1 a na Ganache verzii 2.5.4.

Práca bola vyhotovená na operačnom systéme Windows 10 a Windows 11 64-bit. Otestované prehliadače sú Google Chrome a Brave s doplnkom Metamask, kde je potrebné vytvoriť si účet a následne importovať účty z Ganache. Vrámci metamask je dôležité povoliť v nastaveniach testovacie siete a následne sa prepnúť na sieť *Rinkeby Test Network*.

Spustenie serveru pre back-end aj front-end, kde môžeme hrať zakladnú hru piškvôrky, vytvárať hry, spustíme pomocou *startServer.cmd* súboru, ktorý stačí otvoriť vo Windows. Ide o bežný windows cmd skript, ktorý vykoná spustenie serveru na adrese `http://localhost:3000/`. Po spustení sa hlavná stránka nachádza na spomínanej `http://localhost:3000/`, avšak v prípade ak chcete deployovať kontrakt, používa sa adresa `http://localhost:3000/deploy`.

Potrebné knižnice zložky /master vypíšeme do súboru *libraries.txt* pomocou spustenia skriptu *printLibraries.cmd* a potrebné knižnice /hardhat vypíšeme do súboru *librariesHardhat.txt* pomocou spustenia skriptu *printHLibraries.cmd*. Prípadná doinštalácia knižníc prebieha pomocou cmd, po spustení cmd, zvolíme *cd cesta* napr. *cd /hardhat* a následne napíšeme príkaz *npm install nazovKniznice*.

```
Knižnice
/master
+-- arrayify@1.0.0
+-- bulma@0.9.3
+-- eth-crypto@2.2.0
+-- ethers@5.6.3
+-- isomorphic-unfetch@3.0.0
+-- keccak256@1.0.6
+-- merkletreejs@0.2.31
+-- mongoose@5.9.6
+-- next@9.5.5
+-- pubnub@5.0.1
+-- react-dom@16.13.1
+-- react@16.13.1
+-- semantic-ui-css@2.4.1
+-- semantic-ui-react@0.88.2
`-- web3@1.7.3
/hardhat
+-- @nomiclabs/hardhat-ethers@2.0.5
+-- @nomiclabs/hardhat-ganache@2.0.1
+-- @nomiclabs/hardhat-waffle@2.0.3
+-- @nomiclabs/hardhat-web3@2.0.0
+-- arraify@1.2.1
+-- arrayify@1.0.0
+-- chai@4.3.6
+-- eth-crypto@2.3.0
+-- ethereum-waffle@3.4.4
+-- ethers@5.6.5
+-- hardhat-gas-reporter@1.0.8
+-- hardhat@2.9.3
+-- keccak256@1.0.6
+-- merkletreejs@0.2.31
+-- solidity-coverage@0.7.21
`-- web3@1.7.3
```

Spustenie testov vykonáme pomocou spustenia skriptu *test.cmd*.