Solidity Coding for Smart Contract

Here is our coding used in remix.ethereum.org for deploying our smart contract of student savings account –

```
//SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.6.0;

contract StudentSavingsAccount {

    uint totalBalance = 0;
    uint ContractStartTime;


    constructor() public {
    ContractStartTime = block.timestamp;
    }


    // define Student
    struct student{
        string firstName;
        string lastName;
        string id;
        string nationalId;
    }


    mapping(address => uint) balances;
    mapping(address => uint) depositTimestamps;
```

```
// This function allows user to deposit amount into this smart contract
function addmoney() public payable {
    balances[msg.sender] = msg.value;
    totalBalance = totalBalance + msg.value;

}


function getBalance(address walletAddress) public view returns(uint) {
    uint principal = balances[walletAddress];
    //Calculate seconds
    uint timeElapsed = block.timestamp - depositTimestamps[walletAddress];
    //simple interest of 5%  per year
    return principal + uint((principal * 5 * timeElapsed) / (100 * 365 * 24 * 60 * 60));
}


function secondyearBalance(address walletAddress) public view returns(uint) {
    uint principal = balances[walletAddress];
    //Calculate seconds
    uint timeElapsed = ContractStartTime - depositTimestamps[walletAddress]; //seconds
    //simple interest of 8%  for 2nd year
    return principal + uint((principal * 8 * timeElapsed) / (100 * 730 * 24 * 60 * 60));
}
```

```
    //Student can invest money to any marketplace
    function investmoney() external payable {
        if (msg.value < 2 ether) {
            revert();
        }
        balances[msg.sender] += msg.value;
    }


    //Owner can transfer 2 ether from this cotract to receipent address
    function sendEther (address payable recipentaddress) external {
        recipentaddress.transfer (2 ether);
    }


    // Owner can also withdraw money
   function withdraw() public payable returns (bool) {
        address payable withdrawTo = payable(msg.sender);
        uint amountToTransfer = getBalance(msg.sender);
        balances[msg.sender] = 0;
        totalBalance = totalBalance - amountToTransfer;
        (bool sent,) = withdrawTo.call{value: amountToTransfer}("");
        require(sent, "transfer failed");

        return true;
    }

    receive() external payable {
    }
}
```