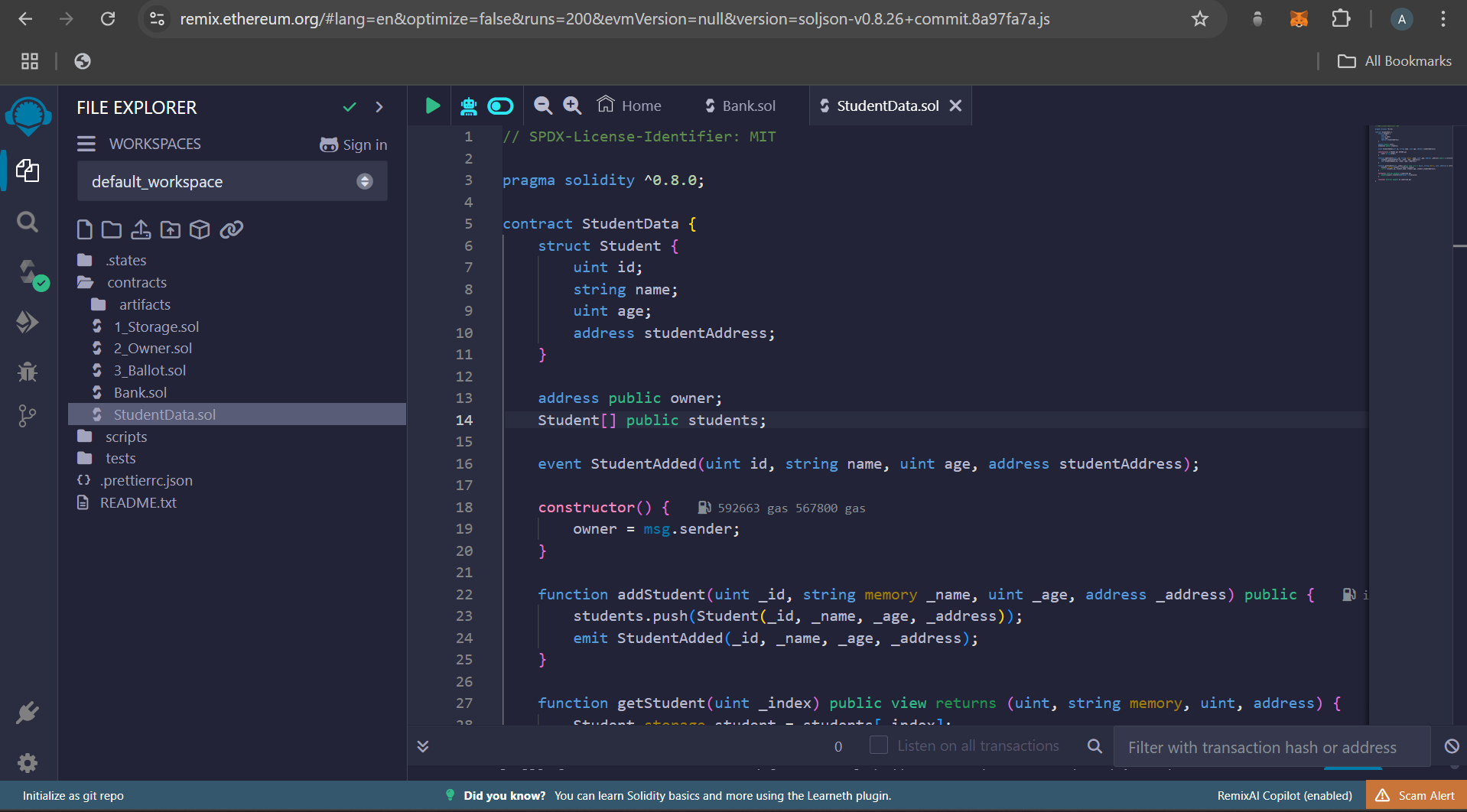
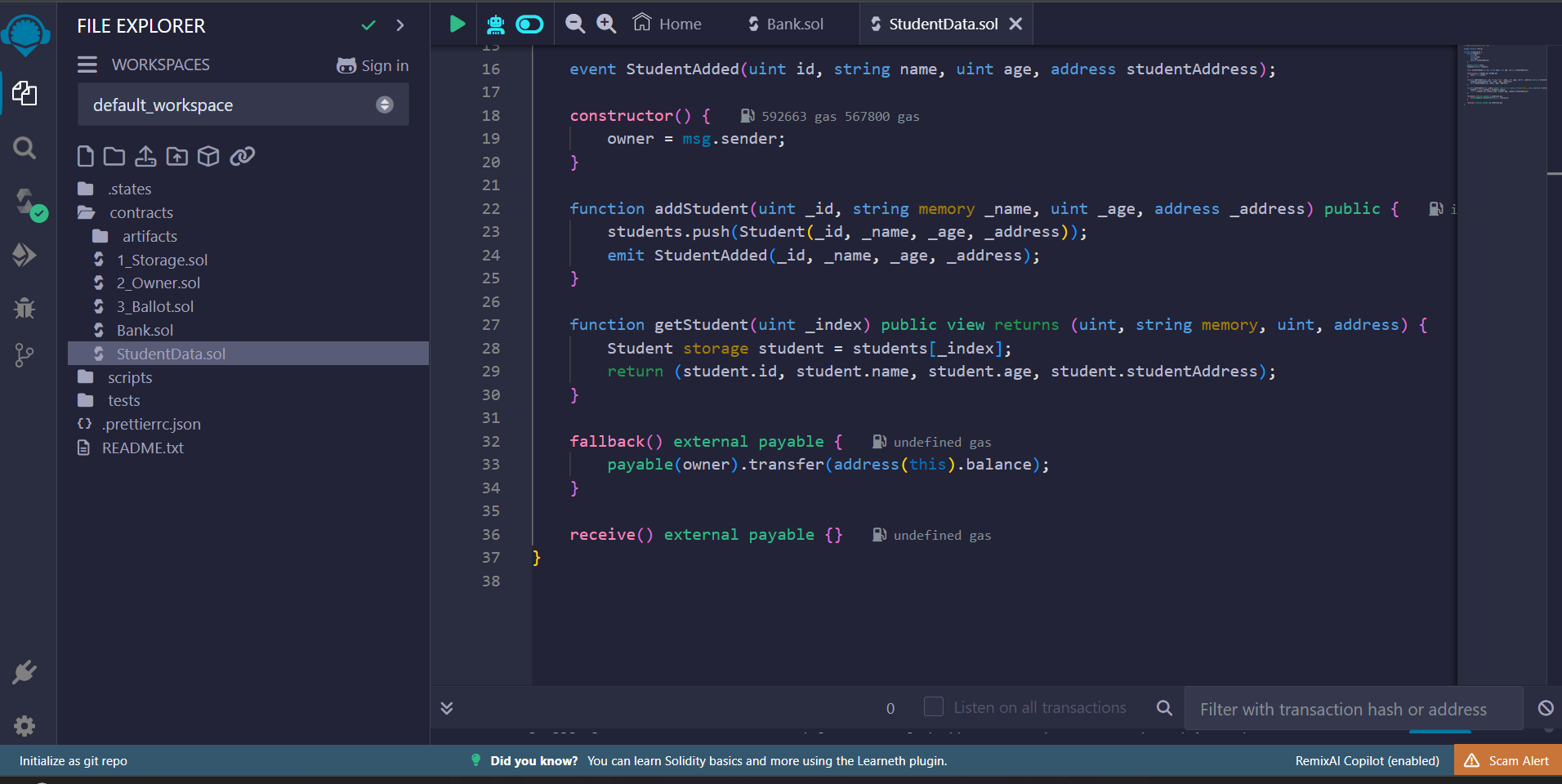
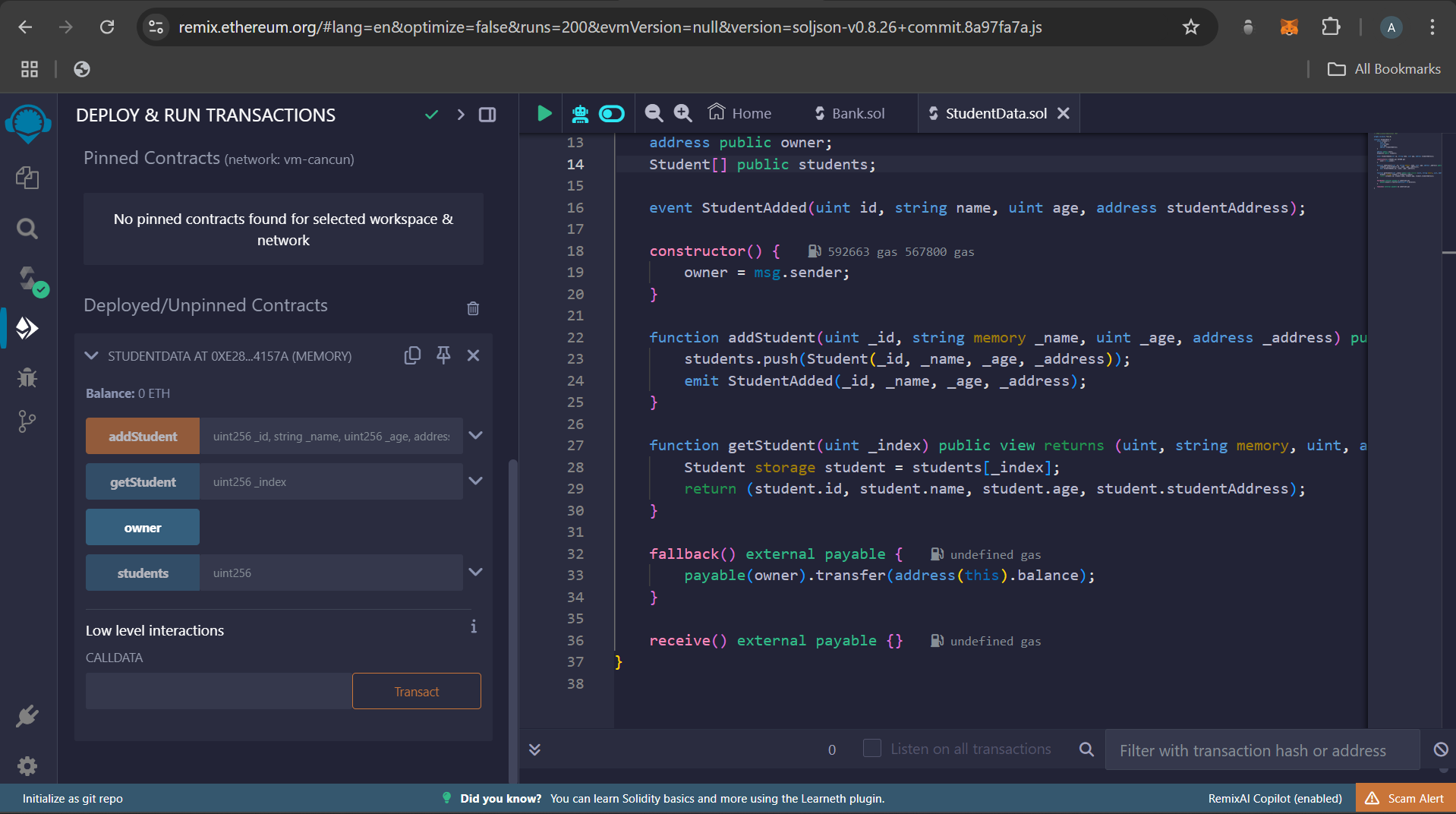
**Student Data**

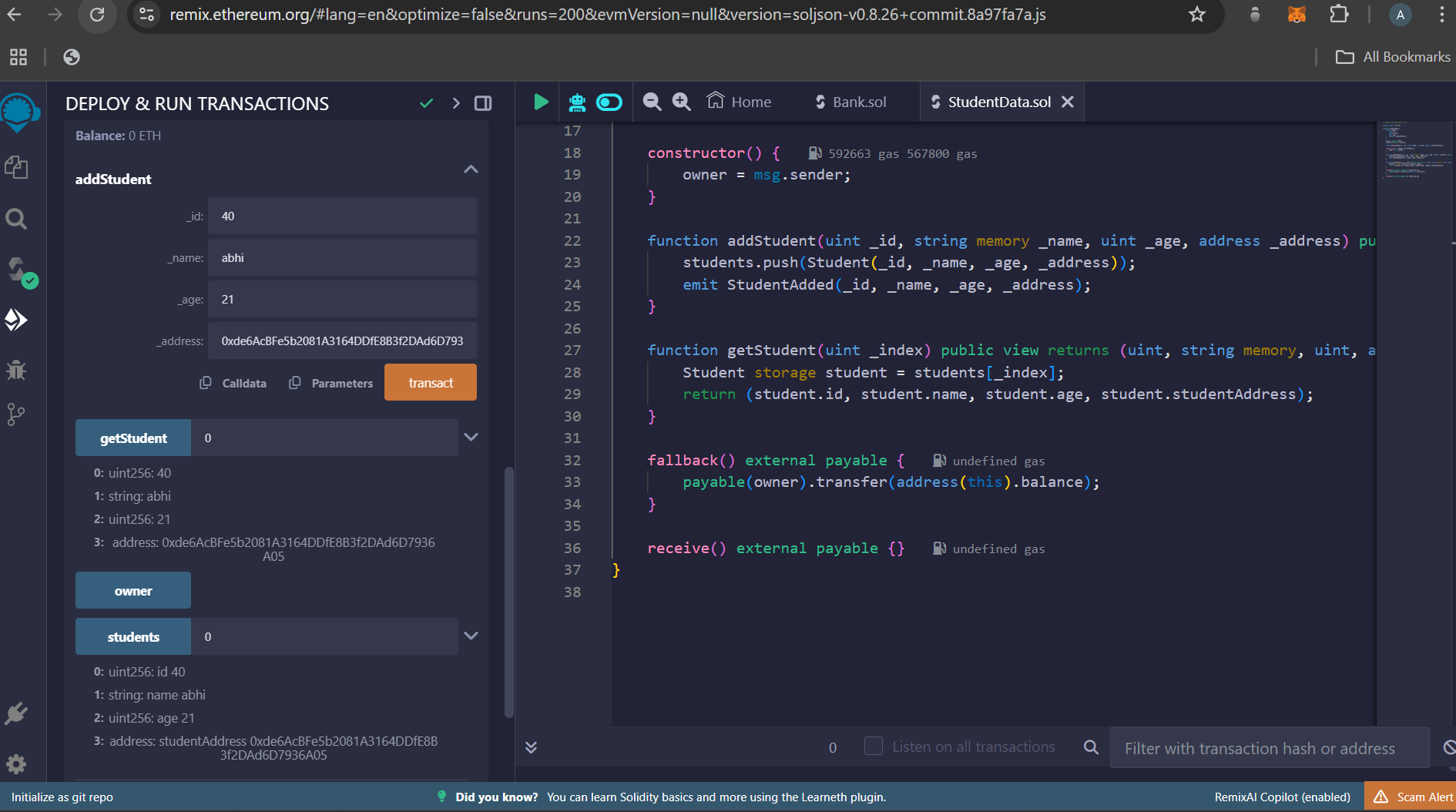
Code for student data





Compile and deploy the code, then test the code





// SPDX-License-Identifier: MIT

pragma solidity >= 0.8.0;

contract StudentData{

struct Student{

uint rollNumber;

string name;

uint balance;

// address block\_address;

}

Student[] students;

constructor(uint[3] memory rolls, string[3] memory names, uint[3] memory balances){

for(uint i=0; i<3; i++){

Student memory student;

student.rollNumber = rolls[i];

student.name = names[i];

student.balance = balances[i];

// students[i].block\_address = msg.sender;

students.push(student);

} // for - i

}

function getStudentByRollNumber(uint rollNumber) external view returns(string memory studentName, uint balance){

for(uint i=0; i<students.length; i++){

if(students[i].rollNumber == rollNumber){

return(students[i].name, students[i].balance);

}

} // for - i

}

function addStudent(uint rollNumber, string memory name, uint balance) external returns(string memory message){

Student memory student = Student(rollNumber, name, balance);

students.push(student);

return("STUDENT ADDED SUCCESSFULLY !");

}

function deductFromStudent(uint rollNumber, uint amount) external returns(string memory message){

for(uint i=0; i<students.length; i++){

if(students[i].rollNumber == rollNumber){

require(students[i].balance > amount, "INSUFFICIENT\_FUNDS\_ERROR : Student does not have enough funds !");

students[i].balance -= amount;

return("DEDUCTED SUCCESSFULLY");

} // if

} // for - i

return("STUDENT DOES NOT EXIST IN RECORDS");

}

function sendToStudent(uint rollNumber, uint amount) external returns(string memory message){

for(uint i=0; i<students.length; i++){

if(students[i].rollNumber == rollNumber){

students[i].balance += amount;

return("CREDITED SUCCESSFULLY");

} // if

} // for - i

return("STUDENT DOES NOT EXIST IN RECORDS");

}

function printAllStudents() external view returns(Student[] memory studs){

return(students);

}

event fallbackEvent(string message, bytes data);

event recieveEvent(string message, uint val);

fallback() external payable{

emit fallbackEvent("FALLBACK TRIGGERED DUE TO DATA IN THE TRANSACTION", msg.data);

}

receive() external payable{

emit recieveEvent("RECIEVE TRIGGERED DUE TO NO DATA IN THE TRANSACTION", msg.value);

}

}