Bank.sol

// SPDX-License-Identifier: MIT

pragma solidity >= 0.7.0;

// Write a smart contract on a test network, for Bank account of a customer for

// following operations: Deposit money | Withdraw Money | Show balance

contract Bank{

mapping(address => uint) public user\_account;

mapping(address => bool) public user\_exist;

function create\_account() public payable returns(string memory){

require(user\_exist[msg.sender] == false, "Account Already created!");

user\_account[msg.sender] = msg.value;

user\_exist[msg.sender] = true;

return "Account created";

}

function deposit(uint amount) public payable returns(string memory){

require(user\_exist[msg.sender] == true, "Account not created!");

require(amount > 0, "Amount should be greater than 0");

user\_account[msg.sender] += amount;

return "Amount deposisted sucessfully";

}

function withdraw(uint amount) public payable returns(string memory){

require(user\_exist[msg.sender] == true, "Account not created!");

require(amount > 0, "Amount should be greater than 0");

require(user\_account[msg.sender] >= amount, "Amount is greater than money deposisted");

user\_account[msg.sender] -= amount;

return "Amount withdrawn sucessfully";

}

function account\_balance() public view returns(uint){

return user\_account[msg.sender];

}

function account\_exists() public view returns(bool){

return user\_exist[msg.sender];

}

}

Contract.sol

// SPDX-License-Identifier: MIT

pragma solidity >=0.7.0 <0.9.0;

contract StudentManagement {

struct Student {

uint stud\_id;

string name;

string department;

}

Student[] public students;

// Event to emit when a student is added - useful for front-end applications

event StudentAdded(uint stud\_id, string name, string department);

// Function to add a new student

function addStudent(uint stud\_id, string memory name, string memory department) public {

Student memory newStudent = Student(stud\_id, name, department);

students.push(newStudent);

emit StudentAdded(stud\_id, name, department);

}

// Function to retrieve student data by ID

function getStudent(uint stud\_id) public view returns (string memory, string memory) {

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

if (students[i].stud\_id == stud\_id) {

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

}

}

revert("Student not found");

}

// The receive function is executed on a call to the contract with empty calldata

// This is the only function that will handle Ether transfers to the contract

receive() external payable {

// Any logic that should be executed when the contract receives Ether

}

}

ishabobo@23

<https://replit.com/@ak47007/Leo>