Source Code

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.5.0;

contract UniversityTracker{
//---Initialization
        // University Admin
        address private _universityAdmin;
        address private _collegeAdmin;

//---Attributes
        //Structure for College
        struct College{
        address _colHash;
        string _colRegNo;
        string _colName;
        bool _isBlocked;
        uint256 _totalStudents;
        }
        //Mapping Address as the primary key to structure College.
        mapping(address => College) _college;

        //Structure for Student
        struct Students{
        string _studentName;
        uint256 _studentPhone;
        string _studentCourse;
        address _collegeAddress;
        }
        //Mapping string(Student Name) as the primary key to Structure Student.
        mapping(string => Students) _student;

//---Constructor
        //To Initialize Univeristy Admin when the contract is deployed.
        constructor() public{
        _universityAdmin = tx.origin;
        }

//---Modifiers
        //To validate if the entity is an University Administrator.
        modifier checkUniversityAdmin(){
        require(tx.origin == _universityAdmin, "University Administrator is required !");
        _;
```

```
            }
            //To validate if the entity is a College Administrator.
            modifier checkCollegeAdmin(address _collegeHash){
            require(tx.origin == _collegeHash, "College Administrator is required !");
            _;
            }
            //Functions that both University and College can access.
    modifier checkCollegeOrUniversityAdmin(address _collegeHash){
            require(tx.origin == _universityAdmin || tx.origin == _collegeHash, "University or College Administrator
is required !");
            _;
            }
            //To validate if University or Student's College is trying to access.
    modifier checkStudentExists(string memory sName){
        require(tx.origin == _universityAdmin || tx.origin == _student[sName]._collegeAddress, "University or
College Administrator is required !");
        _;
    }
//---Functions
    //------College Functions
            //To Add College affiliated to the University
            function addCollege(address cHash, string memory cReg, string memory cName, bool cStatus) public
checkUniversityAdmin{
            require(_college[cHash]._colHash != cHash, "College already exists.");
        _college[cHash]._colHash = cHash;
            _college[cHash]._colRegNo = cReg;
            _college[cHash]._colName = cName;
        _college[cHash]._isBlocked = cStatus;
            }
            //To Add/Block College from Banlist
            function BlockCollege(address cHash) public checkUniversityAdmin{
            require(_college[cHash]._isBlocked != true,"College is already Blocked !");
        _college[cHash]._isBlocked = true;
            }
            //To Remove/Unblock College from Banlist
            function unBlockCollege(address cHash) public checkUniversityAdmin{
            require(_college[cHash]._isBlocked != false,"College is already Unblocked !");
        _college[cHash]._isBlocked = false;
            }
            //To view College Details
            function viewCollgeDetails(address cHash) public checkCollegeOrUniversityAdmin(cHash) view
returns(address, string memory, string memory, bool, uint256){
            return (_college[cHash]._colHash,
            _college[cHash]._colRegNo,
            _college[cHash]._colName,
            _college[cHash]._isBlocked,
            _college[cHash]._totalStudents
            );
            }

//------Student Functions
            //To Add Student into the College
            function addStudent(address cHash, string memory sName, uint256 sPhNo, string memory sCourse)
public checkCollegeAdmin(cHash){
            require(cHash == _college[cHash]._colHash, "College is not affiliated to Unitversity !");
            if(_college[cHash]._isBlocked == true){
               revert("College has been blocked from adding new students !");
            }
            else{
            if(keccak256(abi.encodePacked((_student[sName]._studentName))) ==
keccak256(abi.encodePacked((sName)))){
               revert("Student exists !");
            }
            else{
               _college[cHash]._totalStudents = _college[cHash]._totalStudents + 1;
                    _student[sName]._studentName = sName;
                    _student[sName]._studentPhone = sPhNo;
                    _student[sName]._studentCourse = sCourse;
                    _student[sName]._collegeAddress = cHash;
            }
            }
            }
            //To View Student Details
            function viewStudentDetails(string memory sName) public checkStudentExists(sName) view
returns(string memory, uint256, string memory, address){
        return (
            _student[sName]._studentName,
            _student[sName]._studentPhone,
            _student[sName]._studentCourse,
            _student[sName]._collegeAddress
            );
            }
            //To Change the Student's Course
            function changeStudentCourse(address cHash, string memory sName, string memory sCourse) public
checkCollegeAdmin(cHash){
            require(_student[sName]._collegeAddress == tx.origin, "Student does not belong to this college !");
            if(keccak256(abi.encodePacked((_student[sName]._studentCourse))) ==
keccak256(abi.encodePacked((sCourse)))){
```

```
            revert("Course already selected !");
        }
        else{
            _student[sName]._studentCourse = sCourse;
        }
            }
}
```