2024 年“中银杯”四川省职业院校技能大赛

“区块链技术应用（学生赛）”赛项样题

任务书

参赛队编号：_____

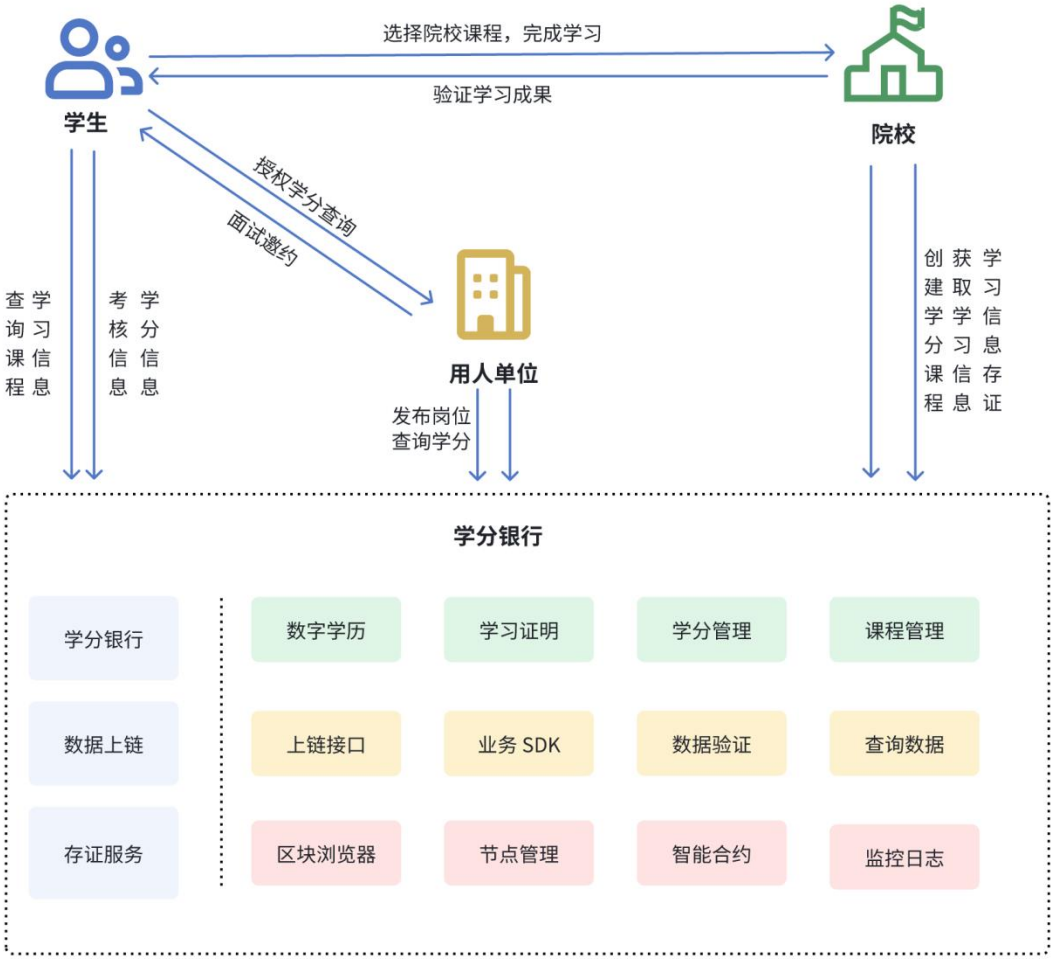
背景描述

截止最新数据，我国共有 16 个省、直辖市进行了学分银行建设，上海市终身教育学分银行已有 376 万学习者开户，9.1 万人转换学分。

学分银行是近年来应用区块链技术发展起来的一种学分管理系统。背景是传统学分管理中存在的问题，包括学分信息分散、数据不一致、难以转移和认证等。而学分银行的建立解决了这些问题，并带来了重要的意义和社会影响。

学分银行提供了高度透明和可信的学分管理系统。通过区块链技术，学分信息被存储在链上的区块中，实现了去中心化和不可篡改的特性。这样，学生、院校和用人单位等各方可以实时查看和核实学分信息，确保了数据的真实性和准确性。这提高了学信认证的可靠性，为院校、用人单位等提供了更准确的评估依据。

基于学分银行的优势，某市教育局计划推广面向全社会的区块链的学分银行平台，将学习者的学分信息以去中心化的方式存储在区块链上，保障学分数据的真实性和准确性。通过推出基于区块链的学分银行平台，学生可以通过注册账号，在区块链平台上进行选课，完成学习考核后获得链上学分；院校可以基于区块链的学分银行平台发布课程，设置课程内容、考核要求及课程学分等；用人单位可以在学分银行平台上发布招聘岗位，授权获取学生真实的学习成果进行面试后符合招聘要求进行录用等；


选择院校课程，完成学习
验证学习成果
学生
院校
授权学分查询
面试邀约
查询学习课程信息
考核学分信息
用人单位
发布岗位
查询学分
创建学习课程
获取学分信息
学习信息存证
学分银行
学分银行
数据上链
存证服务
数字学历
学习证明
学分管理
课程管理
上链接口
业务 SDK
数据验证
查询数据
区块浏览器
节点管理
智能合约
监控日志

模块一：区块链产品方案设计及系统运维（35 分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块一提交结果.docx】中对应的任务序号下。

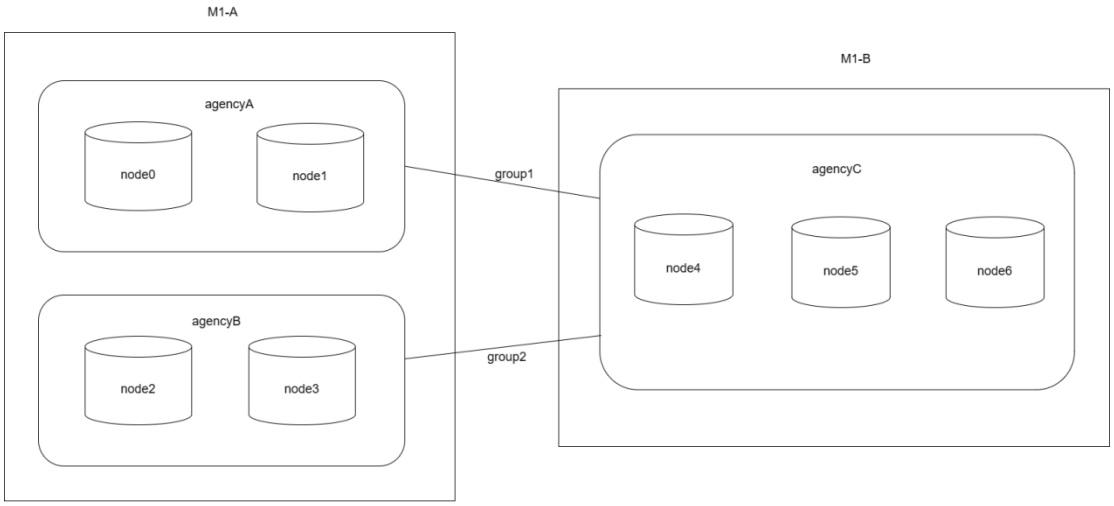
任务 1-1：区块链系统部署与运维

围绕学分银行平台部署与运维需求，进行项目相关系统、节点以及管理工具的部署工作。通过监控工具完成对网络、节点服务的监控。最终利用业务需求规范，完成系统日志、网络参数、节点服务等系统结构的维护，具体要求如下：

1. 根据参数与端口设置要求，部署区块链系统并验证；
2. 根据参数与端口设置要求，部署区块链网络管理平台并验证；
3. 基于区块链系统相关管理平台，按照任务指南实施系统运维工作并验证；
4. 基于区块链系统相关监管工具，按照任务指南对区块链系统进行监管。

子任务 1-1-1： 搭建区块链系统并验证

基于给定服务器环境以及软件（地址“/root/tools”），搭建如下图所示的双机、三机构、二群组、七节点的星形组网拓扑区块链系统。其中，二群组名称分别为 group1、group2，三个机构名称为 agencyA、agencyB、agencyC。p2p_port、channel_port、jsonrpc_port 起始端口分别为 30330、20230、8545，确保搭建的区块链系统能正常运行。



具体工作内容如下：

（1）采用默认配置分别搭建双主机区块链网络；

（2）通过命令分别在 M1-A 和 M1-B 上验证区块链节点进程运行状况；

（3）通过命令分别在 M1-A 和 M1-B 上验证区块链连接状态和共识状态日志输出。

子任务 1-1-2：搭建区块链系统管理平台并验证

基于给定服务器环境以及软件（地址“/root/tools”），搭建区块链控制台并开展相关运维工作，具体工作内容如下：

（1）在 M1-A 主机上面配置控制台，修改配置信息，使用 Console 连接 agencyB 中节点，部署 HelloWorld.sol 智能合约；

（3）使用控制台完成 HelloWorld.sol 智能合约中的 set 与 get 方法操作；

（3）使用机器 M1-A 控制台检查区块链中的当前区块高度以及查看部署合约的交易详情。

子任务 1-1-3：区块链节点运维

基于已完成的区块链系统与管理平台搭建工作，开展区块链节点的加入与退出运维工作，具体内容如下：

（1）基于服务器中的扩容工具，在机器（M1-B）上进行新节点（Node7）扩容并加入群组 Group1；

（2）使用机器（M1-B）检查扩容完成的区块链节点（Node7）的连接状况以及新节点在群组（Group1）中的共识状态；

（3）修改配置文件，指定 node3 输出等级为警告级，并设置日志存储阈值位 50MB。

子任务 1-1-4：区块链网络运维

根据任务描述要求，完成网络配置与管理运维操作，具体内容如下：

（1） 设置区块链系统黑名单，修改 M1-B 的 node7 配置将 node3 设为黑名单，并通过命令检查；

（2） 在 M1-A 上将区块链网络中群组 2 的最大交易数量设为 3000；

（3） 在 M1-A 上通过控制台检查群组 2 的区块最大打包交易数量。

任务 1-2：区块链系统测试

设计对区块链系统的测试流程；结合实际业务需求，调用部署的智能合约进行系统测试、性能测试等；根据业务需求，分析并且修复给定智能合约中的安全漏洞。利用模拟业务和测试工具来完成对区块链系统服务数据的测试。

子任务 1-2-1：系统测试

在 M1-A 登录 linux 服务器，进入指定操作目录（/root/tools）中完成区块链节点管理器的配置部署，并进行节点接口测试，具体操作任务如下：

（1）进入 WeBASE-Node-Manager 目录，完成数据库初始化操作；

（2）修改 application.yml 配置文件，进行 WeBASE-Node-Manager 的服务配置，包括数据库名称，数据库用户，数据库密码等；

（3）使用命令启动 WeBASE-Node-Manager 管理平台服务，并检查节点管理是否正常启动；

（4）进行节点管理服务的 API 接口测试。

子任务 1-2-2：压力测试

参照工程项目（地址：“/root/projects”）使用 Caliper 测试工具对使用 Caliper 测试工具对 HelloWorld.sol 中 set 和 get 功能进行压力测试，具体操作任务如下：

修改 testnw/fisco-bcos.json 文件，同时指定测试 HelloWorld.sol 合约文件配置；

（1） 配置进行压测的 js 信息；

（3）提供 set 功能核心测试代码；

（4）提供 get 功能核心测试代码；

（5）设置 txNumber=100，tps=1，所有测试通过率为 100%。

子任务 1-2-3：智能合约安全漏洞测试。

有如下问题智能合约：

合约一：

```
pragma solidity ^0.8.11;

contract WETH9 {
    string public name   = "Wrapped Ether";
    string public symbol    = "WETH";
    uint8   public decimals = 18;

    event   Approval(address indexed src, address indexed guy, uint wad);
    event   Transfer(address indexed src, address indexed dst, uint wad);
    event   Deposit(address indexed dst, uint wad);
    event   Withdrawal(address indexed src, uint wad);

    mapping (address => uint)   public   balanceOf;
    mapping (address => mapping (address => uint))   public   allowance;


    fallback() external { deposit(); }
    function deposit() public payable {
        balanceOf[msg.sender] += msg.value;

    }
    function withdraw(uint wad) public {
        require(balanceOf[msg.sender] >= wad);
        balanceOf[msg.sender] -= wad;
        payable(msg.sender).transfer(wad);

    }

    function totalSupply() public view returns (uint) {
        return address(this).balance;
    }

    function approve(address guy, uint wad) public returns (bool) {
```

```
            allowance[msg.sender][guy] = wad;

            return true;
        }

        function transfer(address dst, uint wad) public returns (bool) {
            return transferFrom(msg.sender, dst, wad);
        }

        function transferFrom(address src, address dst, uint wad)
            public
            returns (bool)
        {
            require(balanceOf[src] >= wad);

            if (src != msg.sender &&
allowance[src][msg.sender] !=type(uint256).max) {
                require(allowance[src][msg.sender] >= wad);
                allowance[src][msg.sender] -= wad;
            }

            balanceOf[src] -= wad;
            balanceOf[dst] += wad;


            return true;
        }
    }
    合约二：
    // SPDX-License-Identifier: UNLICENSED

    pragma solidity 0.8.11;
    import "./WETH9.sol";

    interface ERC20 {
        function transferFrom(address, address, uint) external;
        function transfer(address, uint) external;
        function balanceOf(address) external view returns (uint);
        function permit(address,address,uint256,uint,uint,bytes32,bytes32)
external;
        function approve(address,address) external;
    }

    contract Vault{
```

```
        constructor() {

        }
        function deposit(address token,uint256 amount) external{
            ERC20(token).transferFrom(msg.sender,address(this),amount);

        }

        function depositwithPermit(address token,address target,uint256 vaule,uint
deadline,uint v,bytes32 r, bytes32 s)external {
            ERC20(token).permit(target,address(this),vaule,deadline,v,r,s);
            ERC20(token).transferFrom(target,address(this),vaule);
        }

    }
    contract Setup {
        WETH9 public weth9;
        Vault public vault;
        constructor() payable {
            uint256 amount = msg.value;
            weth9 = new WETH9();
            weth9.deposit{value: amount}();
            vault = new Vault();
            weth9.approve(address(vault),type(uint256).max);
            vault.deposit(address(weth9),amount/2);

        }
        function withdraw(address token,address target,uint256 vaule,uint
deadline,uint v,bytes32 r, bytes32 s) external {
            vault.depositwithPermit(token, target, vaule, deadline, v, r, s);
        }
        function isSolved() external view returns (bool) {
            uint256 balance = weth9.balanceOf(address(this));
            return balance == 0;
        }
    }
```

（1）分析以上智能合约中存在的漏洞，并说明其可能造成的危害；

（2）编写测试用例，复现智能合约中存在的漏洞；

（3）创建修复后的智能合约，命名为 setupRepair.sol，并编写测试用例修复结果，并说明修复内容。

模块二：智能合约开发与测试（30 分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块二提交结果.docx】中对应的任务序号下。

任务 2-1：智能合约开发

使用 Solidity 语言进行智能合约开发，根据需求功能介绍在待补充源码中完成程序接口功能的编码，解决代码错误和警告，正确编译合约，功能调试正确，并正确进行相关业务功能的验证，具体要求如下：

子任务 2-1-1：院校发布课程的接口编码

根据需求功能介绍在待补充源码中完成院校发布课程功能的编码，解决代码错误和警告，正确编译合约，功能调试正确。

（1）编写院校注册上链接口，完成只有院校的角色可以完成注册，输入院校地址、学校名称、学校位置等信息，该院校账户地址必须不存在，则可以完成院校信息上链，代码截图保存；

```
function registerSchool(address schoolId, string    name, string    location) public {
        Table table = tableFactory.openTable(SCHOOL_TABLE_NAME);
        Entry entry =  选手填写部分;

        entry.set(选手填写部分);
        entry.set(选手填写部分);
        entry.set(选手填写部分);

        table.insert(选手填写部分);
    }
```

（2）编写院校发布课程接口，完成只有院校可以发布课程的接口，输入课程 Id、课程名称、学分等信息，完成发布操作，代码截图保存。

```
function registerCourse(uint256 courseId, string   name, uint256 credits,uint256 period,address schoolId) public {
        Table table = tableFactory.openTable(COURSE_TABLE_NAME);
        Entry entry = table.newEntry();

        选手填写部分
```

```
        table.insert(选手填写部分);
    }
```

子任务 2-1-2：学生完成课程学分上链的接口编码

根据需求功能介绍在待补充源码中完成学生之间进行课程学习学分上链功能功能的编码，解决代码错误和警告，正确编译合约，功能调试正确。

（1）编写学生注册上链的接口，完成只有学生的角色可以完成注册，输入学生地址、学校地址等信息，该学生账户地址必须不存在，则可以完成学生信息上链，代码截图保存；

```
function registerStudent(address studentId, string  name, uint256 age,string gender,string major,string class,uint256 isEnrolled,address schoolId) public {
        Table table = tableFactory.openTable(STUDENT_TABLE_NAME);
        Entry entry = table.newEntry();

        // 设置学生信息
        选手填写部分
        // 插入学生信息到表中
        table.insert(选手填写部分);
    }
```

（2）编写学生选课的接口，完成学生选课前提需要学生已注册，课程已发布，输入学生地址、课程 Id 等信息完成学生选课，代码截图保存；

```
function enrollInCourse(address studentId, uint256 courseId) internal {
        // 验证学生是否存在
        require(选手填写部分, "Student does not exist");

        // 验证课程是否存在
        require(选手填写部分, "Course does not exist");

        // 验证学生是否已选修该课程
        require(选手填写部分, "Student is already enrolled in the course");

        // 添加关联关系到关联表
        Table enrollmentTable = tableFactory.openTable(选手填写部分);
        Entry entry = enrollmentTable.newEntry();
       选手填写部分



        int256 count =
 enrollmentTable.insert(选手填写部分);
        require(count > 0, "Failed to enroll in the course");
    }
```

任务 2-2：智能合约测试

子任务 2-2-1：基于 Web 前置平台的合约测试

1.解决代码错误和警告，正确编译并部署合约，成功获取部署的合约地址和 abi；

2. 调用学分银行智能合约的接口，完整验证业务流程；

3. 通过 console 控制台查看对应操作的结果。

模块三：区块链应用系统开发（30 分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块三提交结果.docx】中对应的任务序号下。

任务 3-1：区块链应用前端功能开发

任务 3-1-1：院校课程管理接口

使用 Vue 调用院校课程管理接口,将获取的课程代码、课程名称、课程学分、课程老师等信息传递给前端模板，要求如下：

（1）在 CourseList.vue 文件中完成功能并按照课程管理原型图的长度、宽度、行高、间距、文字样式、颜色等，完成课程管理页面的开发，将 Web 页面和代码截图保存；

```
<el-table :data="list">
            <el-table-column
              type="选手填写部分"
              label="编号"
              width="120px"
            ></el-table-column>
            <el-table-column
              prop="选手填写部分"
              label="课程代码"
              min-width="120px"
            ></el-table-column>
            <el-table-column
              prop="选手填写部分"
              label="课程名称"
              min-width="120px"
            ></el-table-column>
            <el-table-column
              prop="选手填写部分"
              label="课程学分"
              min-width="120px"
            ></el-table-column>
            <el-table-column
              prop="选手填写部分"
              label="课程老师"
```

```
            min-width="120px"
          ></el-table-column>
          <el-table-column
            prop="选手填写部分"
            label="课程类型"
            min-width="120px"
          ></el-table-column>
          <el-table-column
            prop="选手填写部分"
            label="课程时长"
            min-width="120px"
          ></el-table-column>
          <el-table-column
            prop="选手填写部分"
            label="课程类型"
            min-width="120px"
          ></el-table-column>
          <el-table-column
            prop="选手填写部分"
            label="课程简介"
            min-width="120px"
          ></el-table-column>
          <el-table-column
            prop="选手填写部分"
            label="报名人数"
            min-width="120px"
          ></el-table-column>
        </el-table>
```

（2）完成 Vue 调用课程管理接口 API，获取接口返回的课程代码、课程名称、课程学分、课程老师等信息，填充至 Vue 页面中，完成添加课程功能接口，添加成功后刷新刷新课程管理页面，将 Web 页面和代码截图保存。

```
queryAll () {
      this.axios.get(选手填写部分).then((response) => {
        if (选手填写部分  === 200) {
          this.list =  选手填写部分
        } else {
          this.$message({
            type: 'warning',
            message: '请求内容有误'
          })
        }
```

```
      })
    }
```

任务 3-1-2：学生课程管理接口

使用 Vue 调用学生管理接口，将获取的课程编号、课程名称、课程类型、课程课时、课程学分、授课老师等信息传递给前端模板，要求如下：

（1）在 Selection.vue 文件中完成功能并按照学生管理原型图的长度、宽度、行高、间距、文字样式、颜色等，完成学生管理页面的开发，将 Web 页面和代码截图保存；

```
<el-table :data="选手填写部分">
            <el-table-column prop="选手填写部分" label="课程编号" width="200px"></el-table-column>
            <el-table-column prop="选手填写部分" label="课程名称" width="200px"></el-table-column>
            <el-table-column prop="选手填写部分" label="课程类型" width="200px"></el-table-column>
            <el-table-column prop="选手填写部分" label="课程课时" width="200px"></el-table-column>
            <el-table-column prop="选手填写部分" label="课程学分" width="200px"></el-table-column>
            <el-table-column label="操作">
              <template slot-scope="选手填写部分">
                <el-popconfirm
                  title="确认选这门课程吗？"
                  cancel-button-type="primary"
                  @confirm="选手填写部分"
                  @cancel="选手填写部分"
                >
                  <el-button type="primary" slot="reference">选课</el-button>
                </el-popconfirm>
              </template>
            </el-table-column>
          </el-table>
```

（2）完成 Vue 调用课程管理接口 API，获取接口返回的课程编号、课程名称、课程类型、课程课时、课程学分、授课老师等信息，填充至 Vue 页面中，完成选课、学习课程、投简历等功能接口，将 Web 页面和代码截图保存。

```
query () {
      this.axios.get(选手填写部分).then((response) => {
```

```
        if (选手填写部分 === 200) {
          this.list = 选手填写部分
        } else {
          this.$message({
            type: 'error',
            message: '请求出错'
          })
        }
      })
    }
```

任务 3-2：区块链应用后端功能开发

在区块链学分银行后端开发中,使用 Java 完成在区块链中实现院校、学生、用人单位项目的管理，完善 MySql 数据库表结构创建，完善后端中注册院校、学生、用人单位接口，发布课程、选择课程、发布招聘等功能，与前端页面交互，形成完整的区块链学分银行系统。

任务 3-2-1：区块链学分银行数据库设计

（1）根据提供的区块链学分银行数据库脚本文件，完善数据库的创建表语句，截图保存；

（2）打开 mysql 数据库客户端,并连接数据库,创建 credit_bank 数据库，并执行第 1 步补全的 sql 脚本，将运行过程截图保存。

任务 3-2-2：编写调用院校课程管理的课程列表接口

使用 Java 语言完成 getCourse 接口，完成院校的查询接口，并将注册结果标识信息传递给前端模板，要求如下：

（1）接收 Web 端传递的对应实体类参数；

（2）调用 service 业务层，执行查询课程操作，将查询结果传递给前端页面；

（3）将获取到的课程信息进行解析，并通过数据库依赖包（mysql-connector-java-bin.jar）存储到数据库中；

（4）使用 swagger-ui 测试功能完整性，测试参数和结果截 图,getCourse 接口部分代码截图。

```
@GetMapping("/getCourse")
    @ApiOperation(value = "查询课程")
    public 选手填写部分 getCourse(选手填写部分) {
        String token = request.getHeader(选手填写部分);
        return schoolService.getCourse(选手填写部分);
    }
```

```
@Override
    public AjaxResult getCourse(选手填写部分) {
        List<Course> coursesBySchoolId = courseMapper.选手填写部分(选手填写部分);

        return AjaxResult.me().setResultObj(选手填写部分);
    }
```

任务 3-2-3：编写调用学生学分管理的学分上链接口

使用 Java 语言完成 completeLearningTasks 接口调用智能合约，完成学分上链接口，通过 fisco sdk 与企业管理接口进行交互，并将学分上链信息传递给前端模板，要求如下：

（1）接收 Web 端传递的对应实体类参数；

（2）调用 service 业务层，执行学分上链操作，将发放结果传递给前端页面；

（3）信息上链之后，将获取到的发放信息进行解析，并通过数据库依赖包（mysql-connector-java-bin.jar）存储到数据库中；

（4）使用 swagger-ui 测试功能完整性，测试参数和结果截图,completeLearningTasks 接口部分代码截图。

```
@PostMapping("/completeLearningTasks")
    @ApiOperation(value = "完成学习任务,记录学分")
    public AjaxResult completeLearningTasks(选手填写部分,@选手填写部分 Integer courseId) {
        String token = request.getHeader(选手填写部分);
        return studentService.completeLearningTasks(选手填写部分);
    }
```

```
public AjaxResult completeLearningTasks(String token, Integer courseId) {
        StudentCourse studentCourse = studentCourseMapper.选手填写部分;
        studentCourse.setState("1");
        studentCourseMapper.save(选手填写部分);



        List<Object> funcParams = new ArrayList<>();
```

```
        funcParams.add(选手填写部分);
        funcParams.add(选手填写部分);

        String res = HttpUtils.commonReq(
                webaseDomain.getURL(),
                webaseDomain.getCONTRACT_ADDRESS(),
                webaseDomain.getCONTRACT_NAME(),
                webaseDomain.getCONTRACT_ABI(),
                选手填写部分,
                选手填写部分,
                选手填写部分
        );


        JSONObject _obj = JSON.parseObject(res);
        if (_obj.getIntValue("code") > 0 || !_obj.get("status").equals("0x0")) {
            System.out.println(_obj);
            return null;
        }
        return AjaxResult.me();
    }
```