迭代三分工

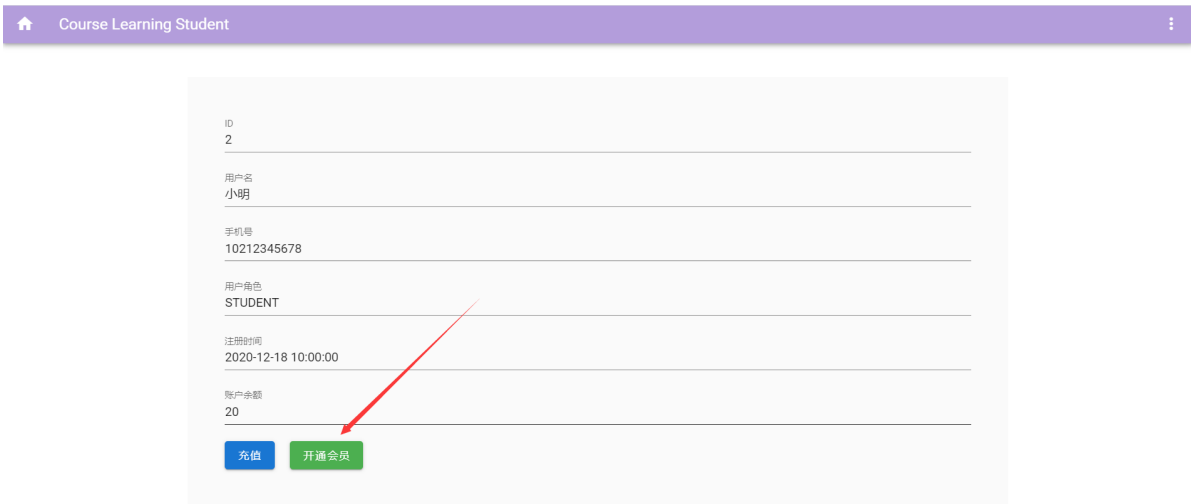
2021-6-17

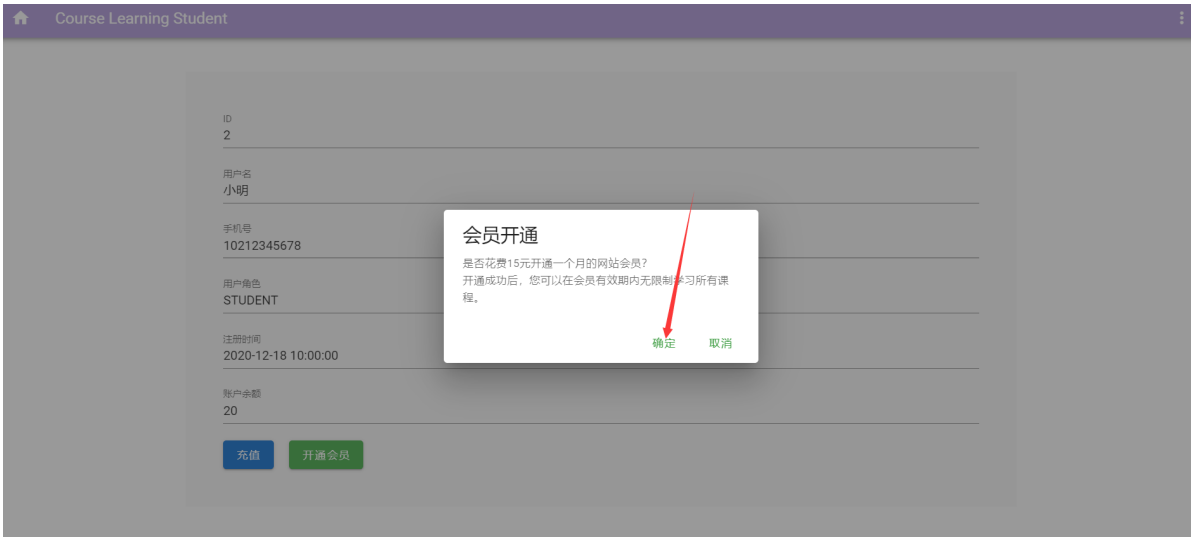
```
export const API_VERSION = "/api";

export const COURSE_MODULE = `${API_VERSION}/course`;
export const ORDER_MODULE = `${API_VERSION}/course_order`;
export const COURSE_WARE = `${API_VERSION}/course_ware`;
export const FILE_MODULE = `${API_VERSION}/file`;
export const USER_MODULE = `${API_VERSION}/user`;
export const RECHARGE_MODULE = `${API_VERSION}/recharge`;
export const COUPON_MODULE = `${API_VERSION}/coupon`;
export const VIP_MODULE = `${API_VERSION}/vip`;
export const QUESTION_MODULE = `${API_VERSION}/question`
export const EXAM_MODULE = `${API_VERSION}/exam`
```

一、开通会员（学生）

学生个人中心页面，点击“开通会员”按钮，可以花费15元开通一个月会员





前端接口：

api/vip.js

```
export const openVip = uid => {
  console.log(uid);
  return axios
    .post( url: `${VIP_MODULE}/`, data: {
      uid
    })
    .then(res => {
      return res.data;
    });
}
```

应在后端controller.order包下添加一个VIPController类，里面的方法映射到前端的api，并在后端完成相关功能需求

功能需求：

1、在user_info表中增加一个字段is_vip，即是否是会员，用户注册时该字段默认为false；增加一个字段vip_deadline，即会员的到期时间，该字段默认为一个比较早的时间（比如1970-01-01 00:00:00）

2、前端会判断用户余额是否足够15元，也会判断用户当前是否已经是会员而避免重复开通，后端不用判断这些逻辑

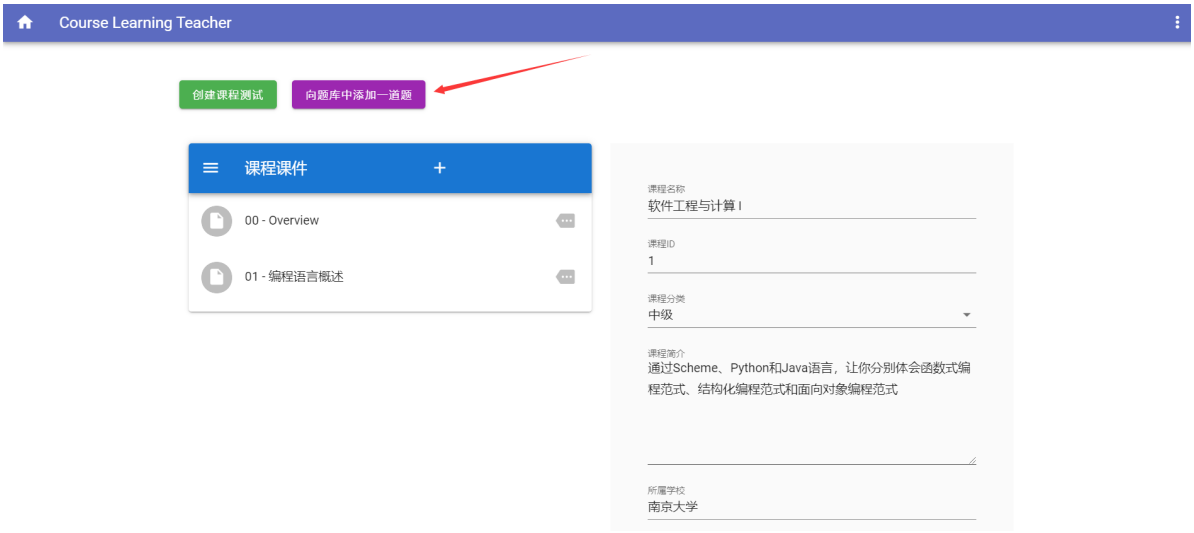
3、若余额足够15元，则调用后端方法，前端会传给后端uid，表示开通会员的用户id

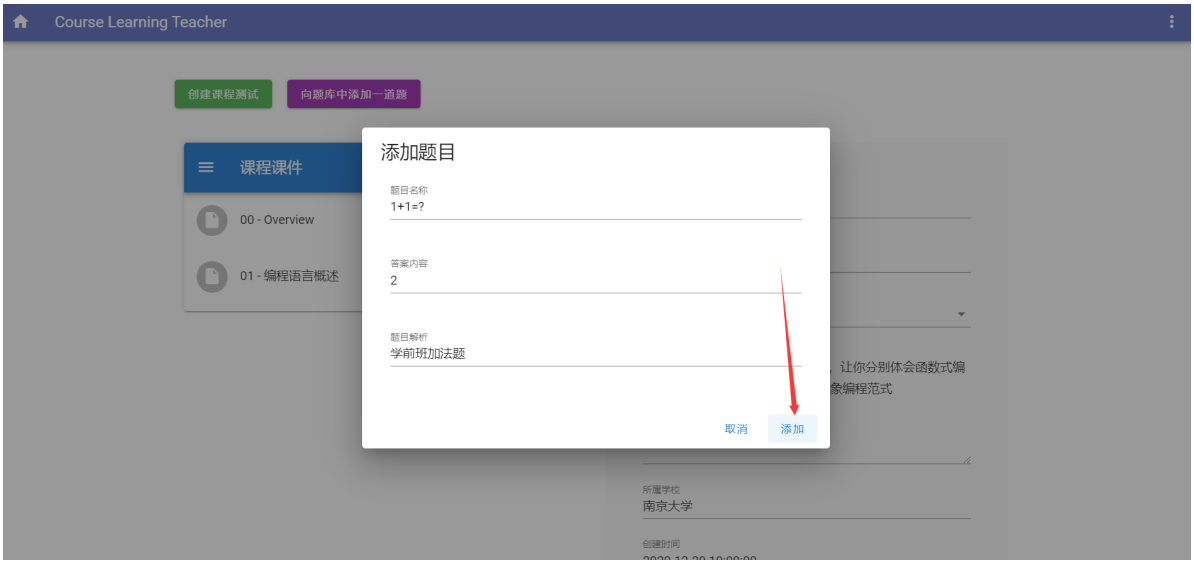
4、后端应在数据库中再添加一个表vip_order，包含uid，开通时间和到期时间

5、后端应把user_info表中对应该用户uid的is_vip字段改为true，把balance字段值减去15，把vip_deadline字段更新为当前时间加上30天；在vip_order表中插入该条订单信息

二、创建题目（教师）

相当于问卷系统的“问题”，且只有填空题





前端接口：

api/question.js

```
/**
 * 添加题目 POST /question
 * @param payload
 * @returns
 */
export const addQuestion = payload => {
  console.log(payload);
  const { courseID, title, answer, analysis } = payload;
  return axios
    .post( url: `${QUESTION_MODULE}/add_question`, data: { courseID, title, answer, analysis })
    .then(res => {
      return res.data;
    });
};
```

前端传给后端数据：{ **course_id, title(String), answer(String), analysis(String)** }

应在后端controller包下添加一个Exam包，即考试包，里面应该有两个Controller类，一个用于添加题目、查找题目的服务；一个用于创建测试、接收学生答卷结果，里面的方法映射到前端的api，并在后端完成相关功能需求

功能需求：

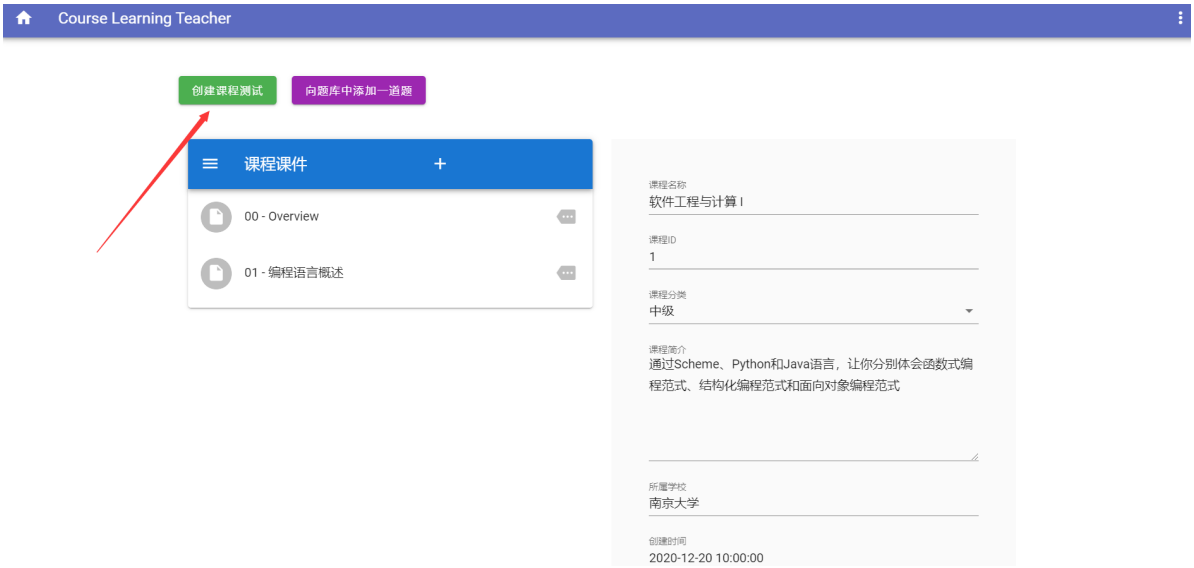
1、需要在数据库中创建一张表question，包含字段：id, course_id, title(varchar), answer(varchar), analysis(varchar，题目的解析)

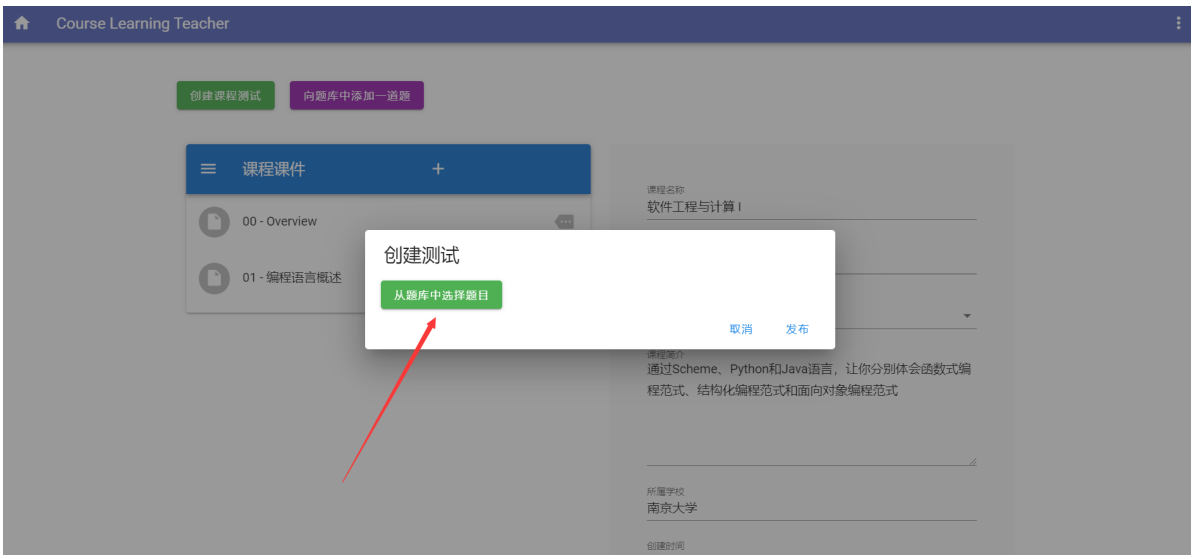
2、后端应新增Question类、QuestionVO类、QuestionPO类等（根据需要创建），用于接收前端传来的question数据，并作数据持久化

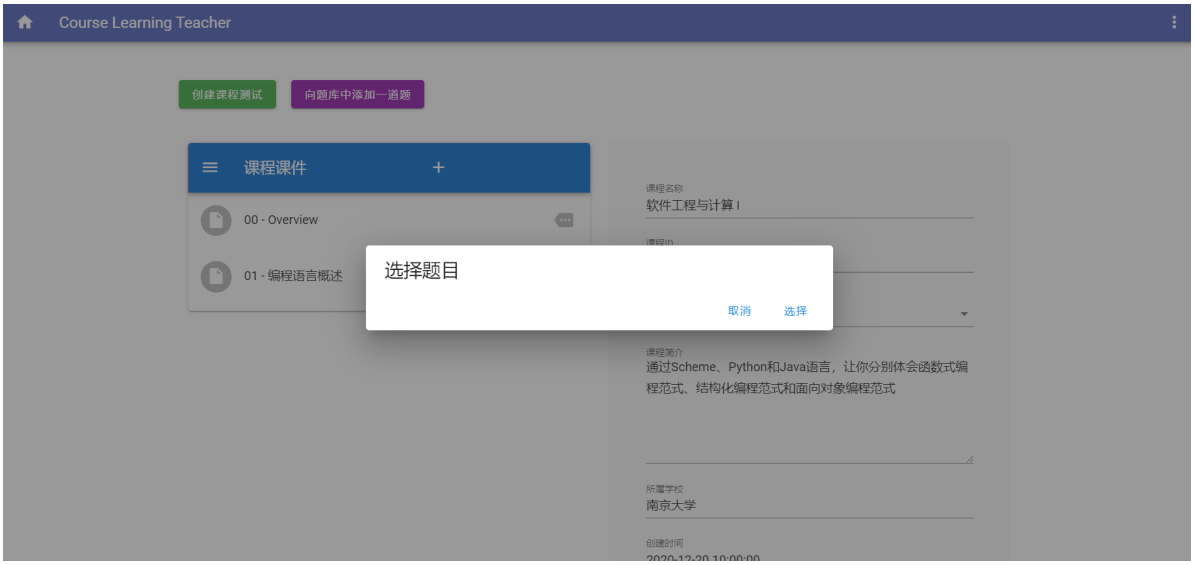
3、将前端传来的question数据插入到question表中，并将结果返回给前端

三、从题库中选题（教师）

教师在某一课程页面中创建课程测试，从该课程的题库中选题







前端接口：

api/question.js

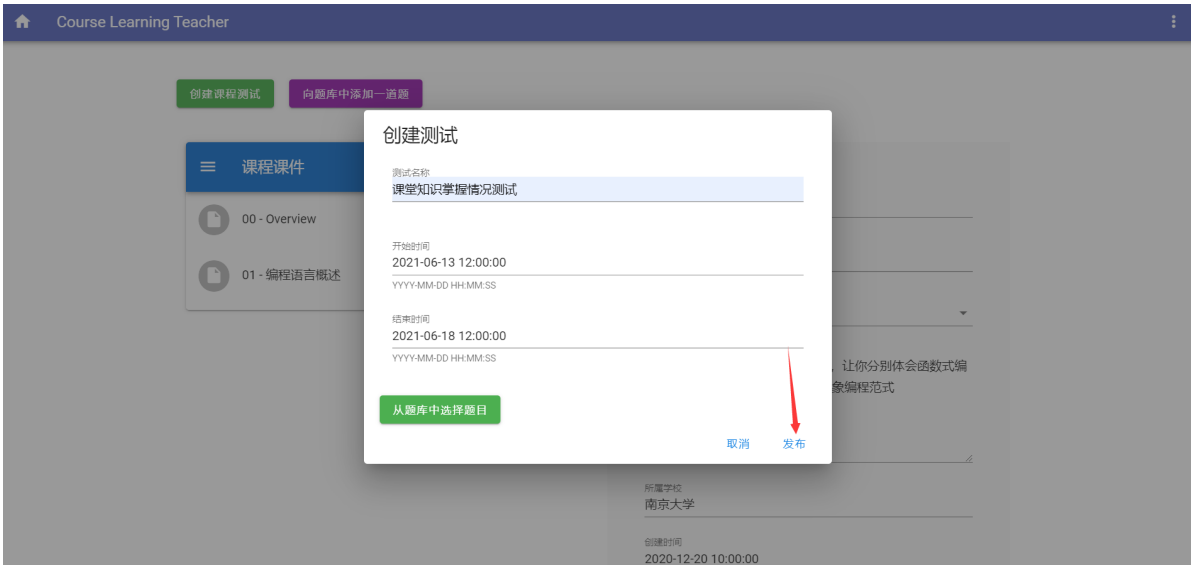
```
/**
 * 展示题库中题目 get /question
 * @param courseID
 * @returns
 */
export const getQuestionsByCourseID = courseID => {
  console.log(courseID);
  return axios
    .get( url: `${QUESTION_MODULE}/${courseID}`)
    .then(res => {
      return res.data;
    });
};
```

功能需求：

前端传给后端course ID，后端返回给前端所有该课程相关的question（List<QuestionVO> res）

四、发布测试卷（教师）

教师选好题目后，点击发布



前端接口：

api/exam.js

```
/**
 * 发布测试 POST /exam
 * @param payload
 * @returns
 */
export const releaseExam = payload => {
  console.log(payload);
  const { courseID, title, questions } = payload;
  return axios
    .post( url: `${EXAM_MODULE}/release`, data: { courseID, title, questions })
    .then(res => {
      return res.data;
    });
};
```

功能需求：

1、数据库中添加exam表，包含字段：id, title(varchar), start_time, end_time，用于存放考试的基本信息，包含id，测试标题，创建时间，截止时间

2、数据库中添加exam_info表，包含字段：exam_id, question_id，用于存放每个考试对应的题目，主键为(exam_id, question_id)

3、在controller包下新建exam包，内含ExamController类，新建ExamVO、ExamPO、Exam类等（根据需要自己设计，我个人认为Exam类中应包含一个成员List<Question> questions，即该试卷中包含的所有问题）

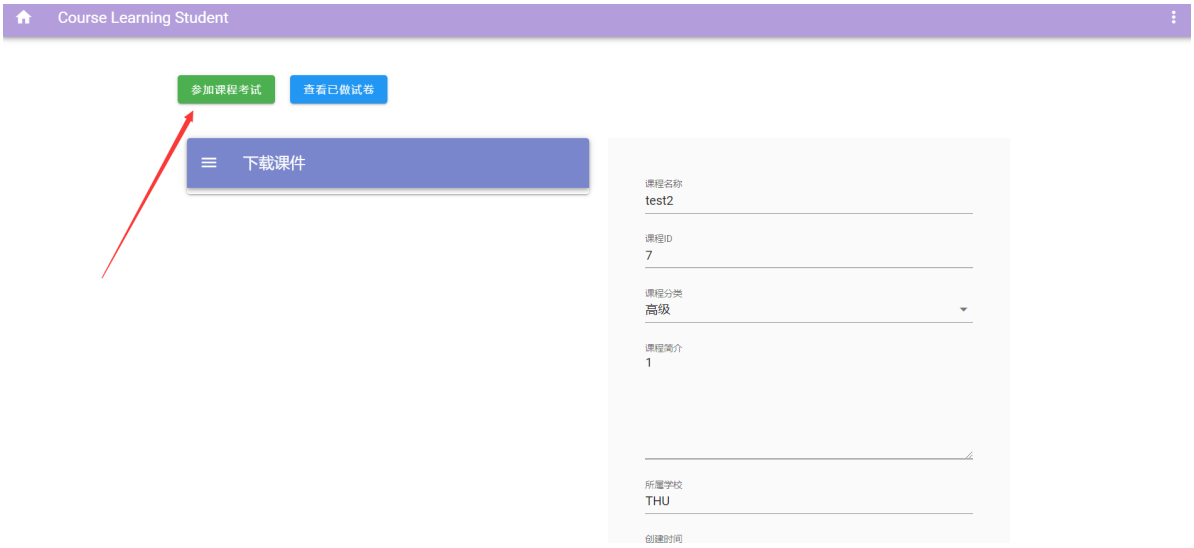
4、前端传给后端{ **courseID, examTitle（考试标题），questions数组（内含多个question对象），startTime(字符串，内容为开始时间，格式为YYYY-MM-DD HH:MM:SS), endTime(字符串，内容为结束时间，格式为YYYY-MM-DD HH:MM:SS)**}

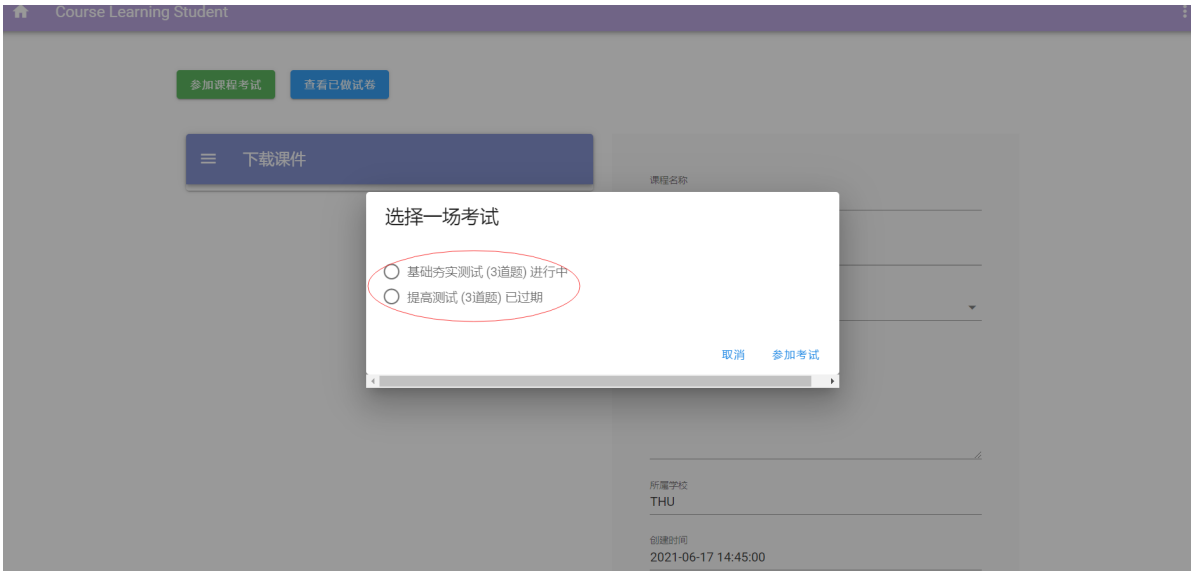
5、后端对exam表进行更新，插入考试的基本信息（标题、发布日期、截止日期）

6、后端对exam_info表进行更新，插入对应考试id的题目

五、获取考试列表（学生）

学生在课程学习页面，点击“参加课程考试”，获取该课程所有的考试





前端接口：

api/exam.js

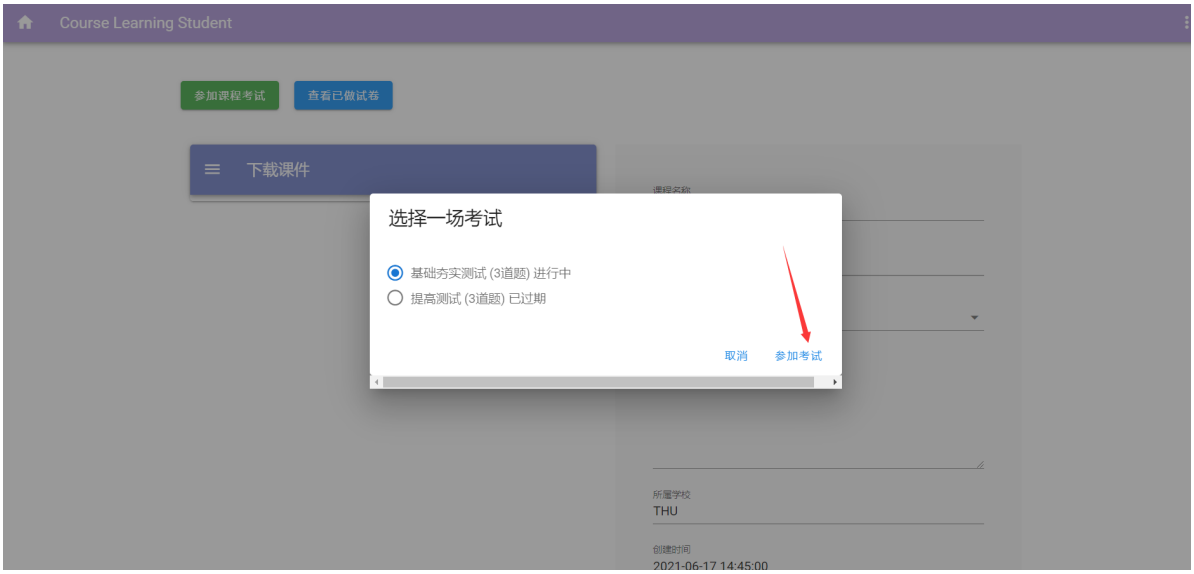
```
/**
 * 查看某课程的所有测试 get /exam
 * @param courseID
 * @returns
 */
export const getExamsByCourseId = courseID => {
  console.log(courseID);
  return axios
    .get( url: `${EXAM_MODULE}/${courseID}` )
    .then(res => {
      return res.data;
    });
};
```

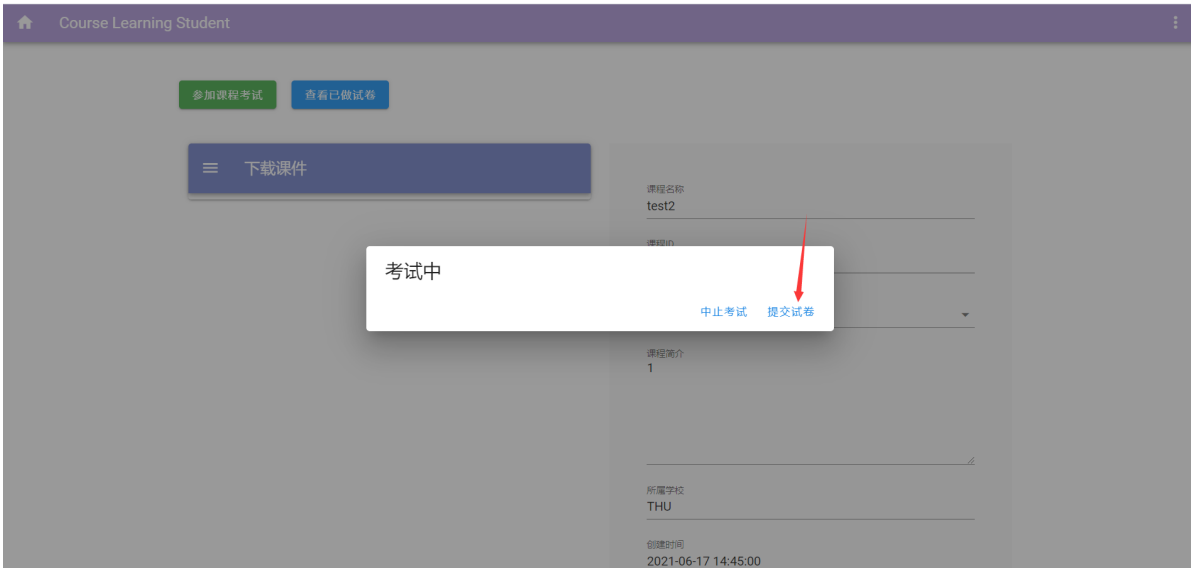
功能需求：

前端传给后端courseID，后端返回该课程对应的所有考试（包括正在进行的和已结束的测试）（List<ExamVO>）

六、学生提交考试结果

学生点击参加考试，进入答题页面，答题完成后提交试卷





前端接口：

api/exam.js

```
/**
 * 提交试卷 POST /exam
 * @param payload
 * @returns
 */
export const submitExam = payload => {
  console.log(payload);
  const { uid, examID, answers } = payload;
  return axios
    .post( url: `${EXAM_MODULE}/submit`, data: { uid, examID, answers })
    .then(res => {
      return res.data;
    });
};
```

前端传给后端数据包括：{ **uid，exam_id，answers数组** }，其中answers数组元素为每道题的答案对象，形式为：

```
answers: [
    {
        questionID: Number,
        answer: String
    },
    ...
]
```

功能需求：

1、后端应在数据库中添加一个**answer表**，用于存放学生的考试答案，包含字段：id, uid, exam_id, question_id, answer(varchar)（根据实际自己设计）

2、后端收到前端传给的数据后，在answer表中作更新

3、给前端返回信息，用于提示学生是否提交试卷成功

4、后端应在数据库中添加一个**scores表**，用于存放学生的考试情况，包含字段：id, uid, exam_id, score(用于存放uid学生在exam_id的考试中取得的分数)

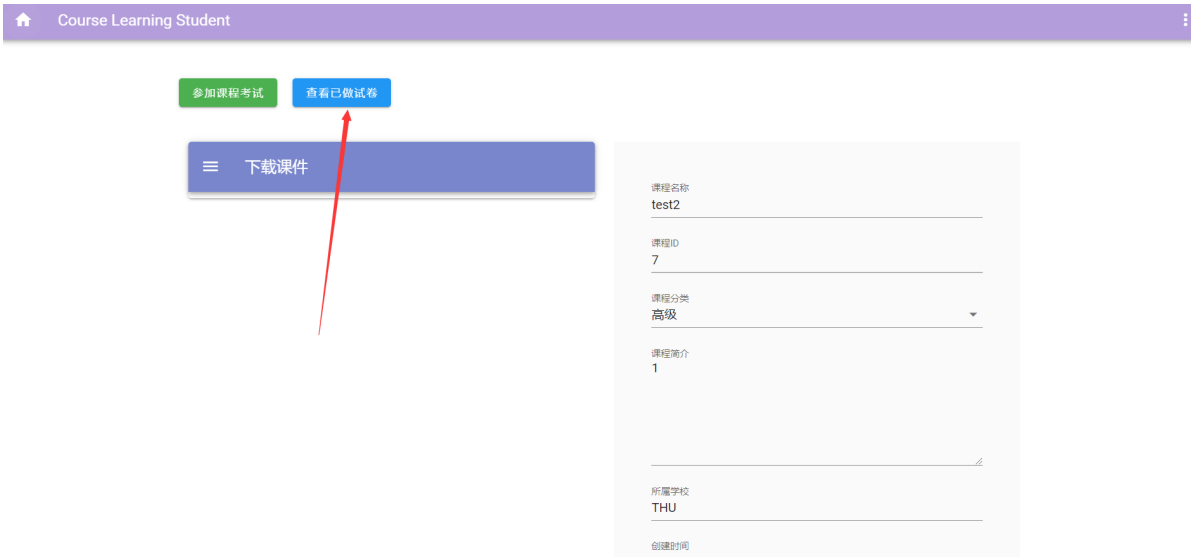
5、后端对前端传来的数据（学生的答案）与数据库中question表中的标准答案进行对比，由于都是字符串形式，只需要比较是否一致就可以了，一致就是答对了，否则答错

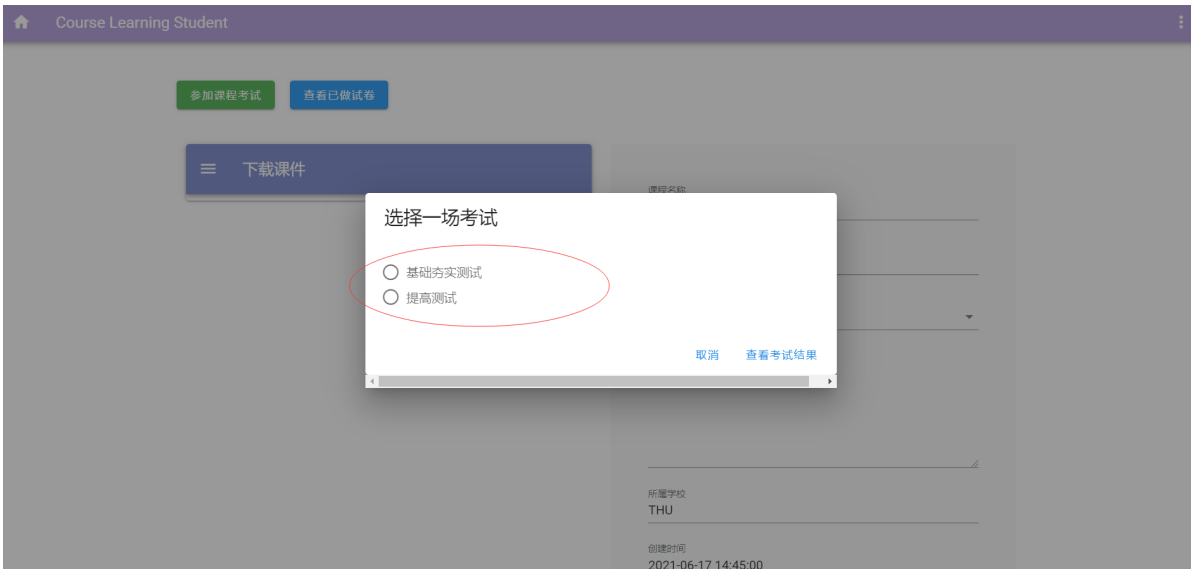
6、记满分为100分，则学生取得的最终分数为：$Final\ Score = \frac{\text{答对的题目数量}}{\text{该试卷所有题目数量}} \times 100(\text{分})$

7、将学生的得分情况存入数据库的**scores表**中

七、学生获取已做试卷列表

学生在课程学习页面点击查看已做试卷





前端接口：

api/exam.js

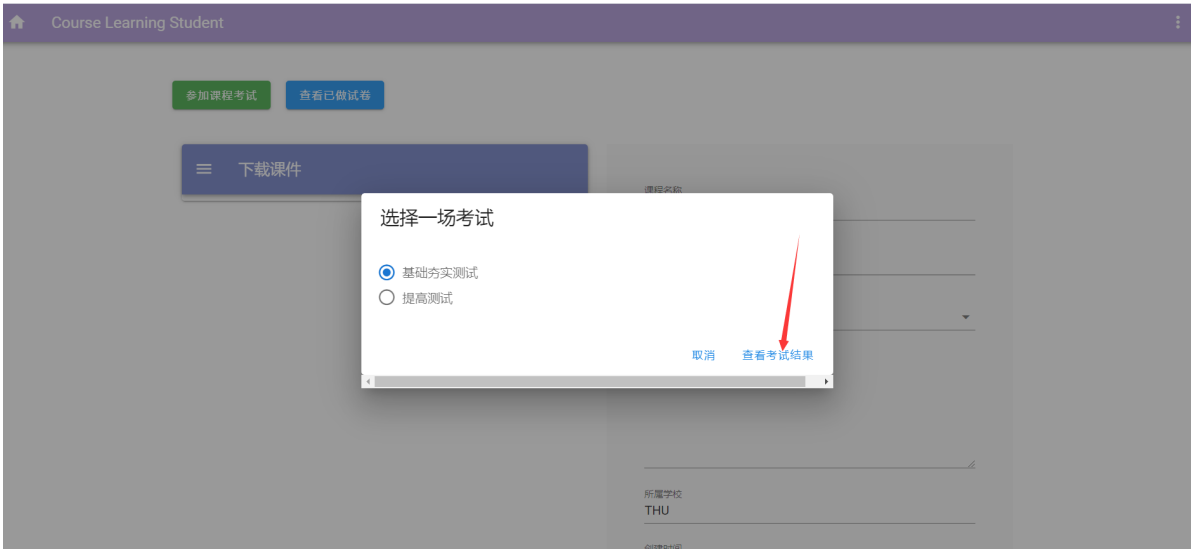
```
/**
 * 查看某学生做完的所有测试 get /exam
 * @param uid
 * @returns
 */
export const getExamsByUserID = uid => {
  console.log(uid);
  return axios
    .get( url: `${EXAM_MODULE}/${uid}`)
    .then(res => {
      return res.data;
    });
};
```

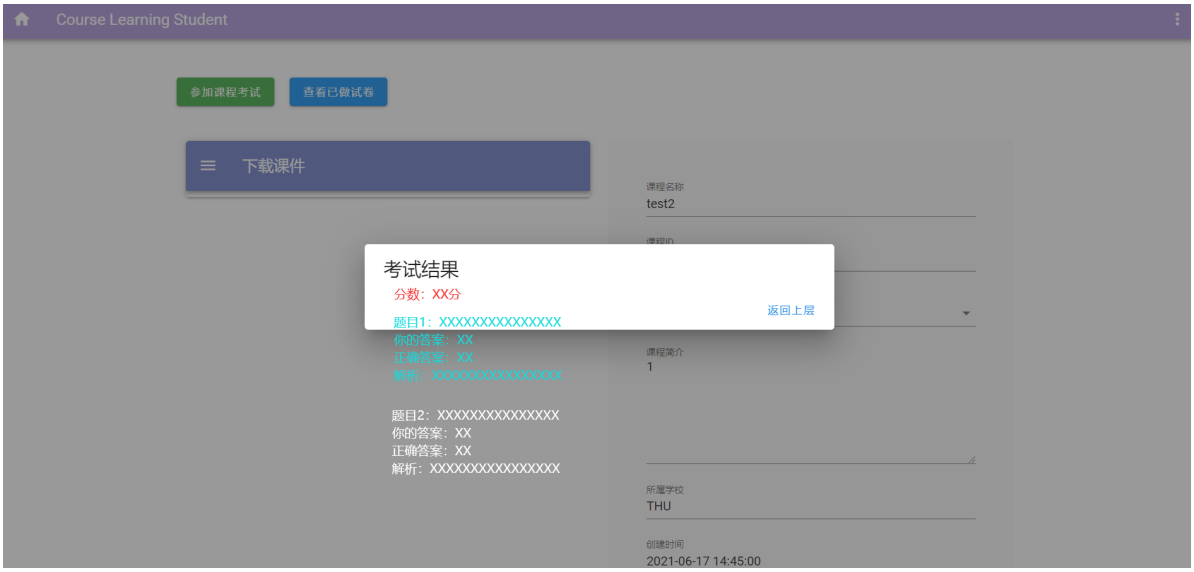
功能需求：

前端传给后端uid，后端返回该学生完成的所有考试（List<ExamVO>）

八、获取考试结果

学生在查看已做试卷时，选择一场已完成的考试并点击查看考试结果，





前端接口：

api/exam.js

```
/**
 * 查看学生在某测试中的答题得分情况 get /exam
 * @param payload
 * @returns
 */
export const getExamResults = payload => {
  console.log(payload);
  const { uid, examID } = payload;
  return axios
    .get( url: `${EXAM_MODULE}/${uid}?exam_id=${examID}` )
    .then(res => {
      return res.data;
    });
};
```

前端传给后端 { uid, examID }

功能需求：

1、接收前端数据，返回给前端该uid学生在该exam_id考试中的 **情况**

2、**情况**包含：

- 考试得分（从score表中获取）
- 该考试中每道题的题目、标准答案、题目解析，以及该学生这道题的答案
- 每道题该学生是否做对了（也可以不包含这一数据，既然后端给了前端每道题目的标准答案和学生的答案，是否做对可由前端进行判断）