**在线课堂系统课程报告**

张泽繁,120L022429

**摘要:**在疫情期间,为了满足“停课不停学”的要求,同时解决在线学习平台不够成熟的问题,运用业务流程管理的方法对系统进行业务流程创新, 为学员构建了一个全方位的在线学习平台。

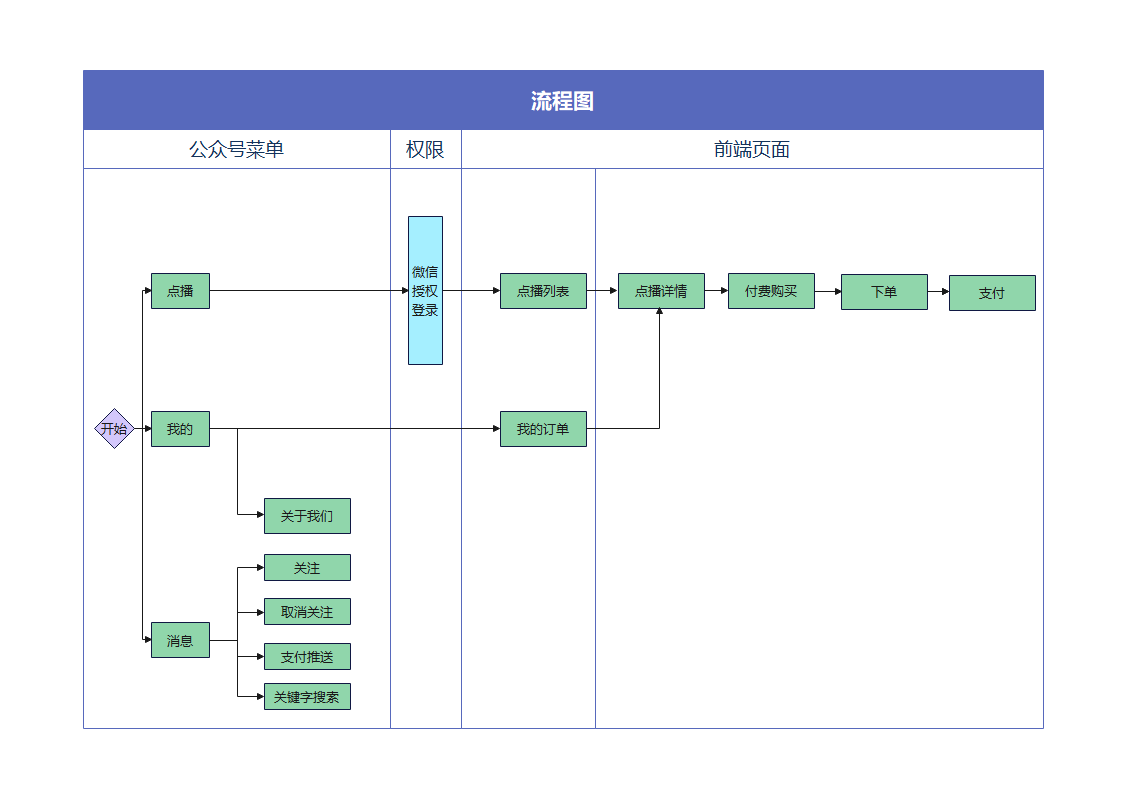
**关键词:**在线课堂,As-is,SOMA,ESB,微服务

1. 系统背景介绍

2020年初，新冠疫情的突然出现和快速传播，让世人始料未及又措手不及，虽然我国很快控制了疫情的蔓延，并争分夺秒进行复工复产，但考虑到教学的实际情况，国家发出了停课不停学的号召，于是线上教学便成为了一种新鲜而且受人期待的教学方式。而且，多年来世界网络信息技术飞速发展，特别是直播平台技术的成熟与完善，让线上教学具备了可能性。



系统背景介绍

****

传统业务流程 as-is

1. 服务模式创新

2.1问题分析

通过分析可以发现,当前的业务结构存在一些问题，主要体现在缺乏促销与推广的功能以及灵活的身份绑定机制。这些问题可能导致以下一些困扰：

限制访问渠道：目前仅依赖微信进行客户访问，如果由于某种原因客户无法使用微信号，将无法访问已购买的课程或其他服务。这种单一访问渠道的限制可能导致客户流失和不满意。

缺乏分享功能：当前业务结构没有提供分享功能，这意味着客户无法通过推荐或分享课程等内容来吸引更多潜在客户。分享功能是有效的口碑营销手段，能够扩大品牌知名度和吸引更多用户。

优惠措施的缺失：在现有的业务结构中，缺乏优惠券、免费试看等促销措施，这使得吸引新客户和提高客户黏性的效果受到限制。通过提供优惠券和免费试看等活动，可以激励用户试用课程并增加他们的购买欲望。

2.2基于用户偏好的支付推送服务创新:

个性化推荐：为了满足用户的个性化需求，推荐系统可以基于用户过往的学习记录、课程评价和搜索记录等信息，利用机器学习算法分析用户兴趣和偏好，向用户推荐最适合他们的课程。

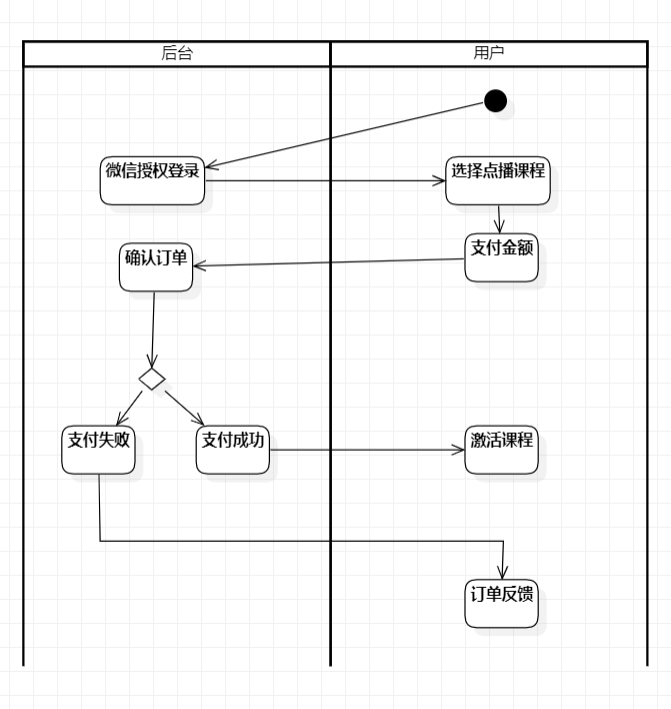
社交化推荐：在线课堂推荐服务可以通过社交网络，将用户与志同道合的学习者联系起来。这可以让用户更容易地了解其他学习者的学习经验和意见，从而找到更符合自己兴趣和需要的课程。

增加互动性：为了提高用户参与度和学习效果，推荐系统可以向用户推荐互动性更高的课程。这些课程可能包括在线辅导、小组讨论和实践项目等，可以让用户更加深入地了解课程内容，并与其他学习者交流和分享经验。

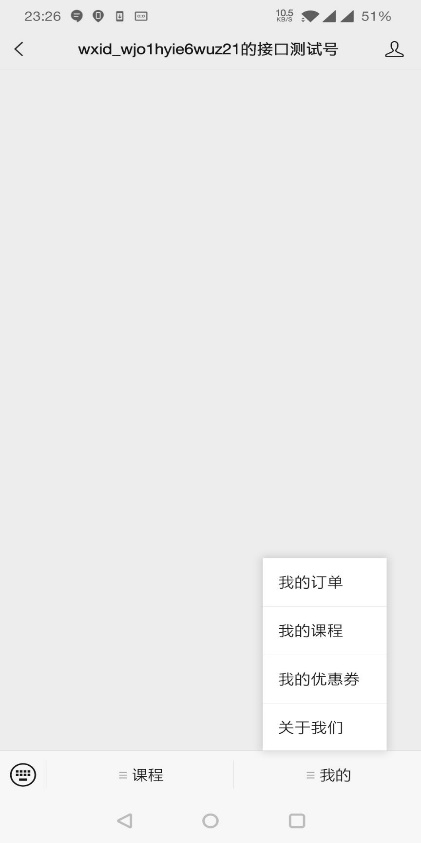
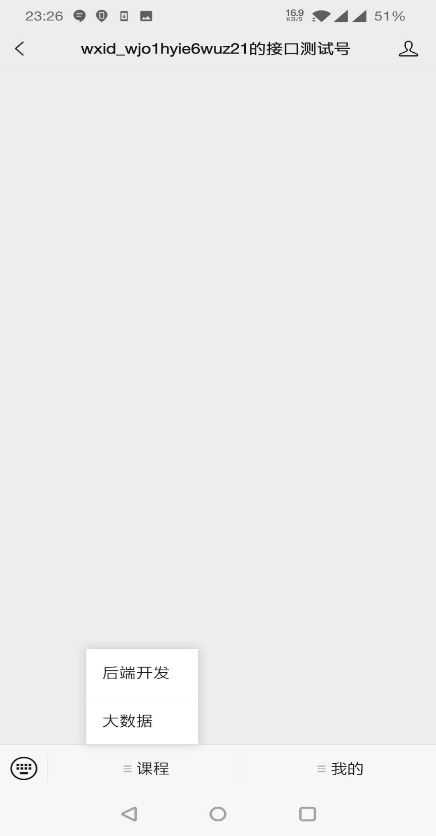
课程多样性：推荐系统应该提供丰富的课程类型和主题，以满足不同用户的需求。除了基础课程和专业课程外，还可以推荐兴趣爱好相关的课程、职业发展相关的课程和实用技能课程等，以满足用户不同的学习目的。

实时性：推荐系统应该及时更新和推荐新的课程，以反映市场和行业的变化。这样可以让用户更好地了解最新的知识和技能，保持竞争优势。

用户反馈：推荐系统应该充分利用用户反馈来优化推荐算法和课程选择。用户可以通过课程评价、投票和评论等方式向系统提供反馈，以改进推荐系统的准确性和个性化程度。



业务流程图



公众号测试号

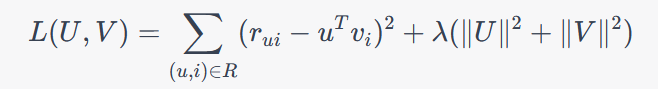
用户兴趣建模是推荐系统中的一个关键任务，它旨在根据用户的历史行为数据来捕捉和表示用户的兴趣和偏好。我们采用协同过滤算法,具体如下:

假设用户具有相似的兴趣会对相似的物品有相似的偏好。基于协同过滤的用户兴趣建模算法使用矩阵分解技术， Alternating Least Squares (ALS)。

假设有一个用户-物品矩阵 R，其中每个元素 rui 表示用户 u 对物品 i 的评分或行为。

ALS算法的目标是找到两个低维矩阵 U 和 V，使得 R≈UVT，其中 U 是用户兴趣矩阵，V 是物品表示矩阵。

ALS算法通过交替固定 U 或 V 来最小化以下损失函数：



其中，λ是正则化参数，用于控制模型的复杂度。

2.3基于已购课程的精准投放:

精准投放优惠券可以提高优惠券使用率，增加销售量和用户忠诚度。

收集用户数据：为了能够根据已购课程精准投放优惠券，需要先收集用户的购买记录和行为数据。这些数据可以包括用户购买的课程名称、购买时间、购买金额、购买方式等信息。

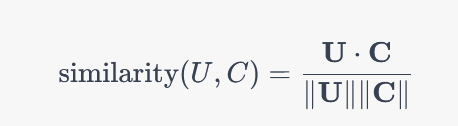
分析用户购买行为：根据收集的用户数据，可以利用数据分析工具进行分析和挖掘，了解用户购买行为和偏好。例如，可以分析用户购买的课程类型、购买频率、购买金额等指标，以识别用户的购买习惯和特征。

制定优惠券策略：基于分析结果，可以制定针对不同用户群体的优惠券策略。例如，对于经常购买某一类型课程的用户，可以针对该类型课程推出优惠券，以激励用户继续购买该类型的课程。

精准投放优惠券：在制定优惠券策略后，需要将优惠券投放给相应的用户。可以通过多种方式进行投放，例如电子邮件、短信、站内信等。需要确保优惠券的投放方式和内容能够吸引用户的注意力和促进消费。

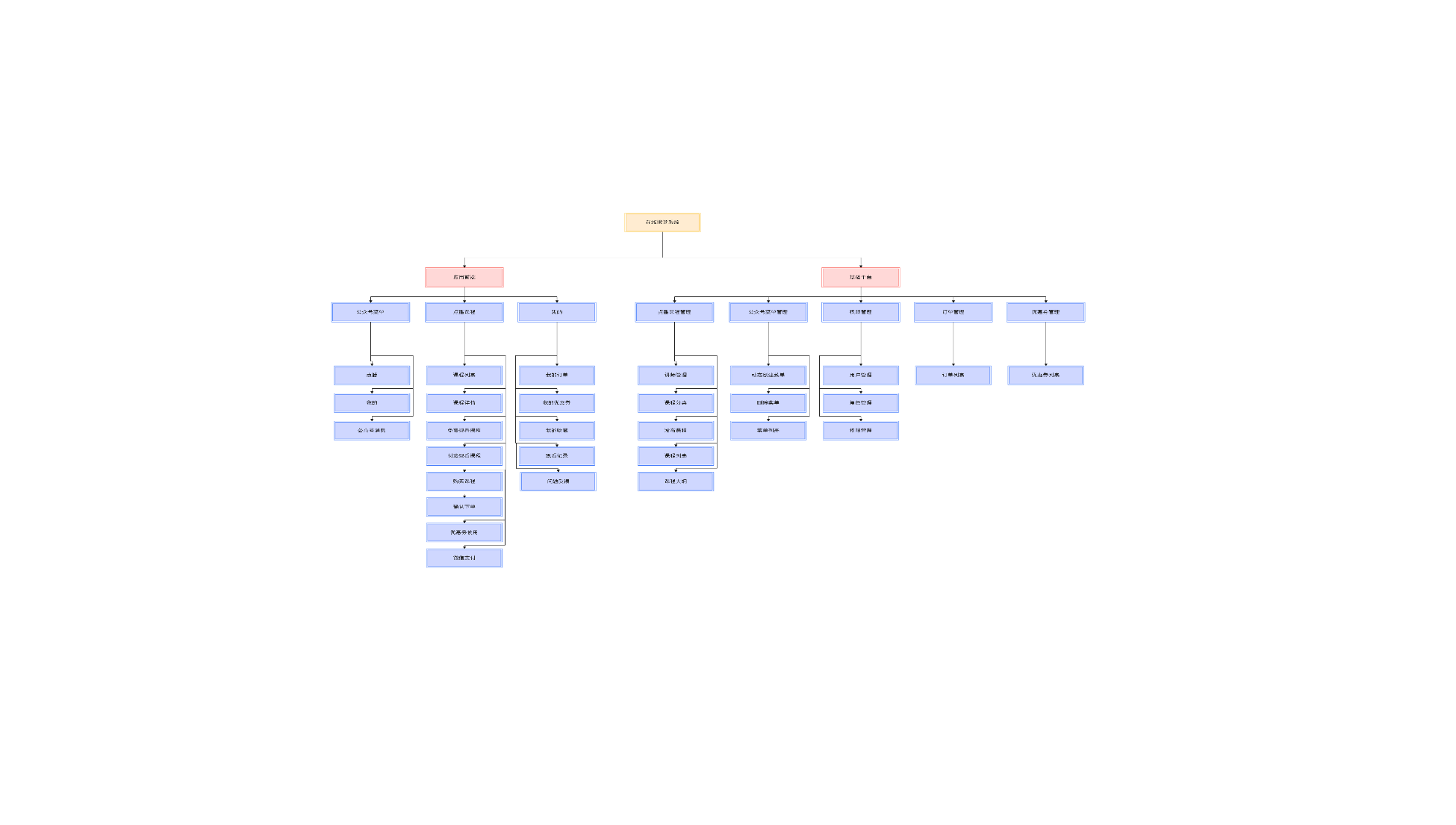
监测优惠券使用情况：投放优惠券后，需要不断监测和评估优惠券的使用情况和效果。例如，可以跟踪用户的使用行为，分析优惠券的使用率和转化率等指标，以不断优化优惠券策略和投放方式。

我们使用余弦相似度（Cosine Similarity）来度量广告与用户已购课程之间的相关性。余弦相似度可以计算两个向量之间的夹角，值越接近1表示两个向量越相似，值越接近0表示两个向量越不相似。

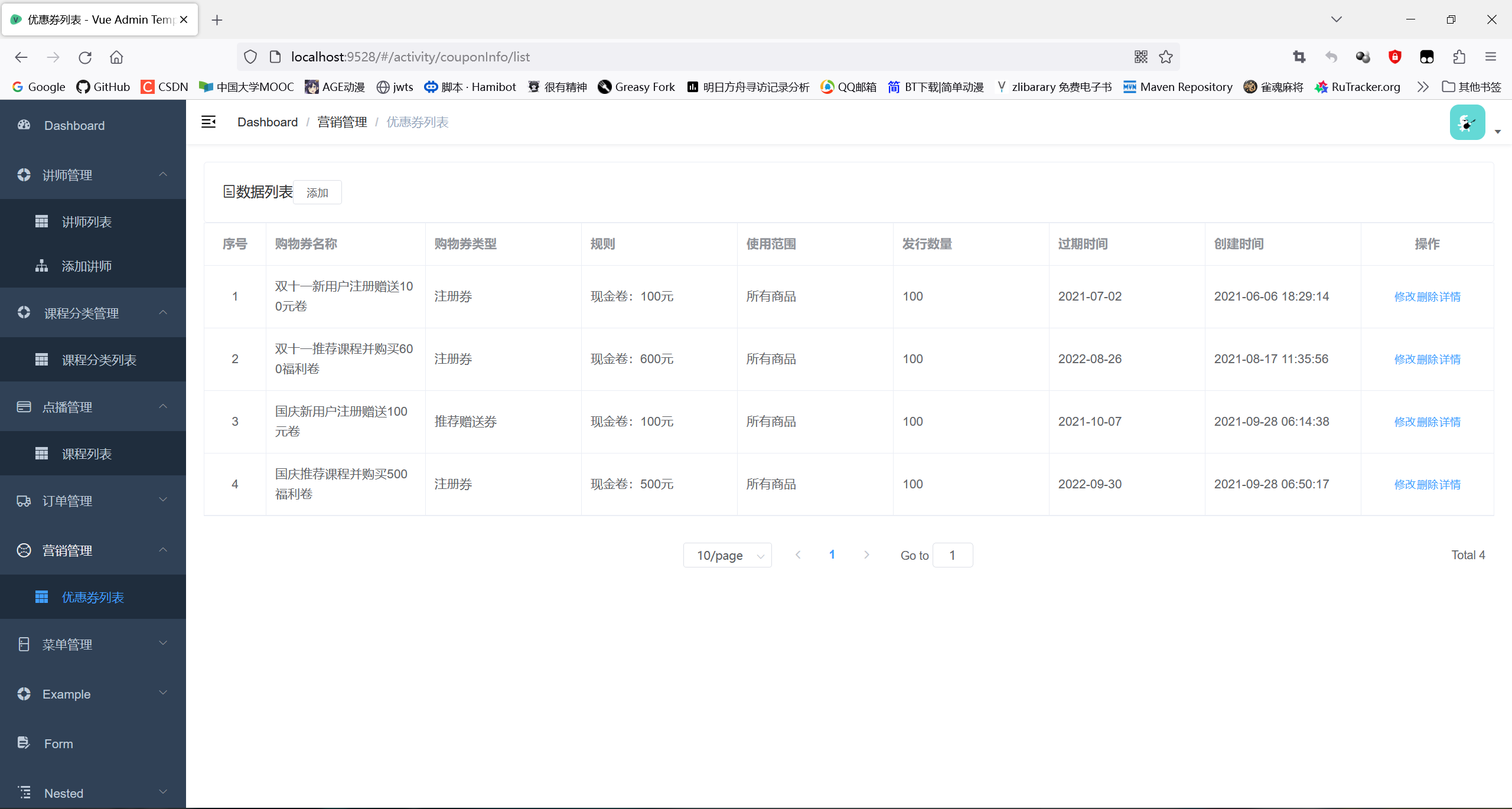


cosine\_similarity = dot\_product(U, A) / (norm(U) \* norm(A))

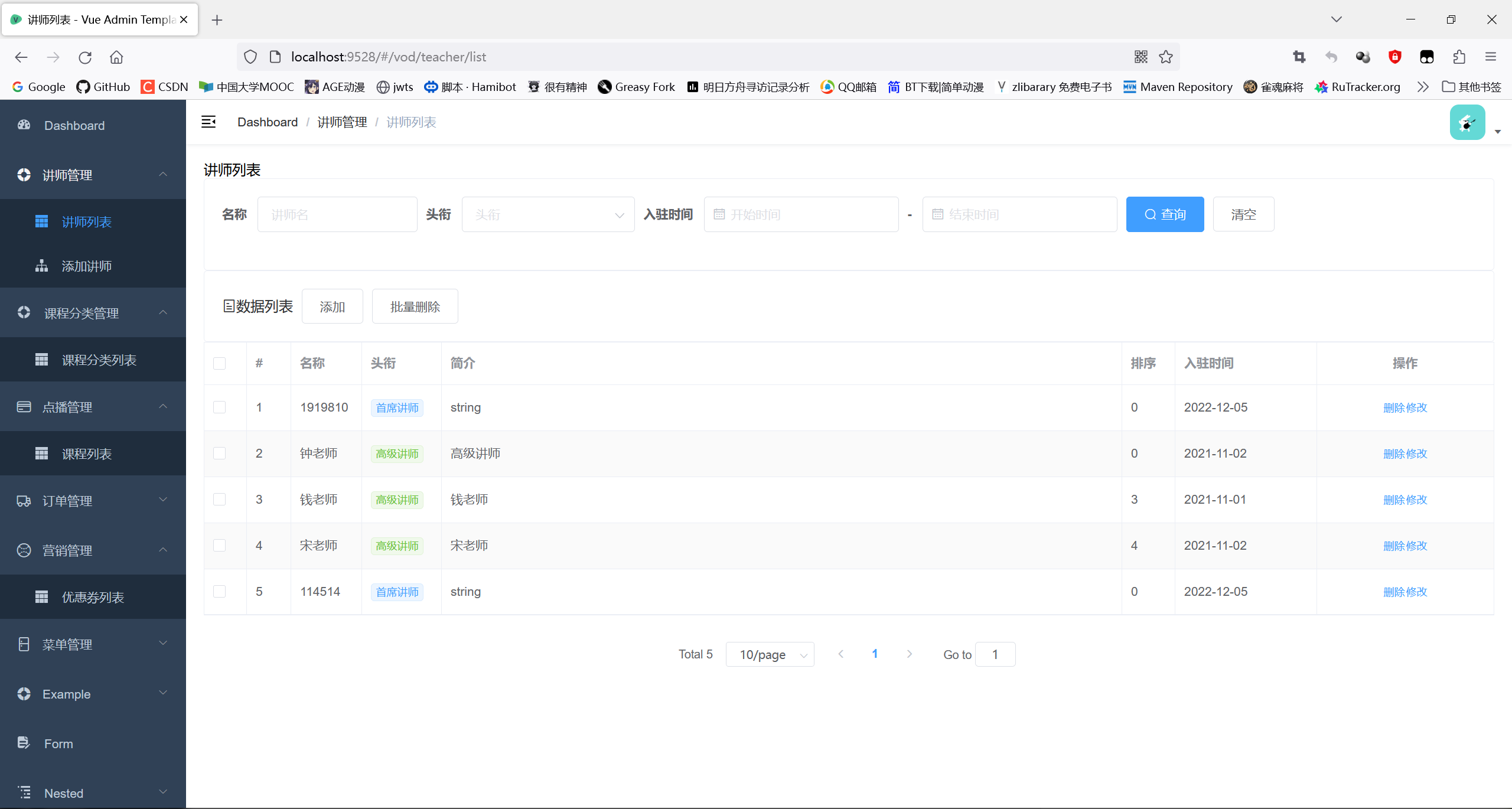
其中，dot\_product(U, A)表示已购课程向量 U 和广告向量 A 的点积（元素对应相乘后相加），norm(U)表示已购课程向量 U 的范数（即向量的长度），norm(A)表示广告向量 A 的范数。



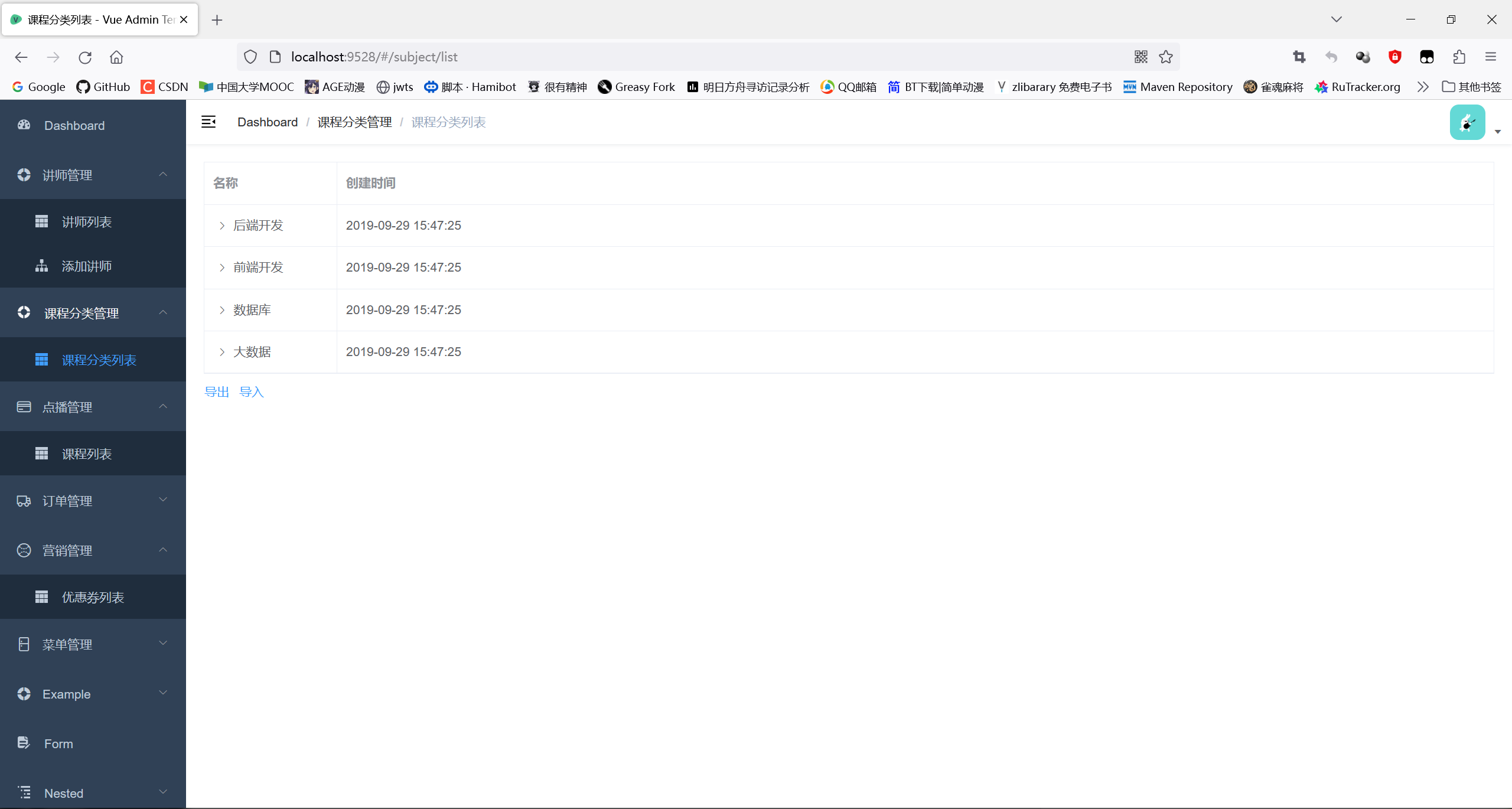
整体框架和设计原型



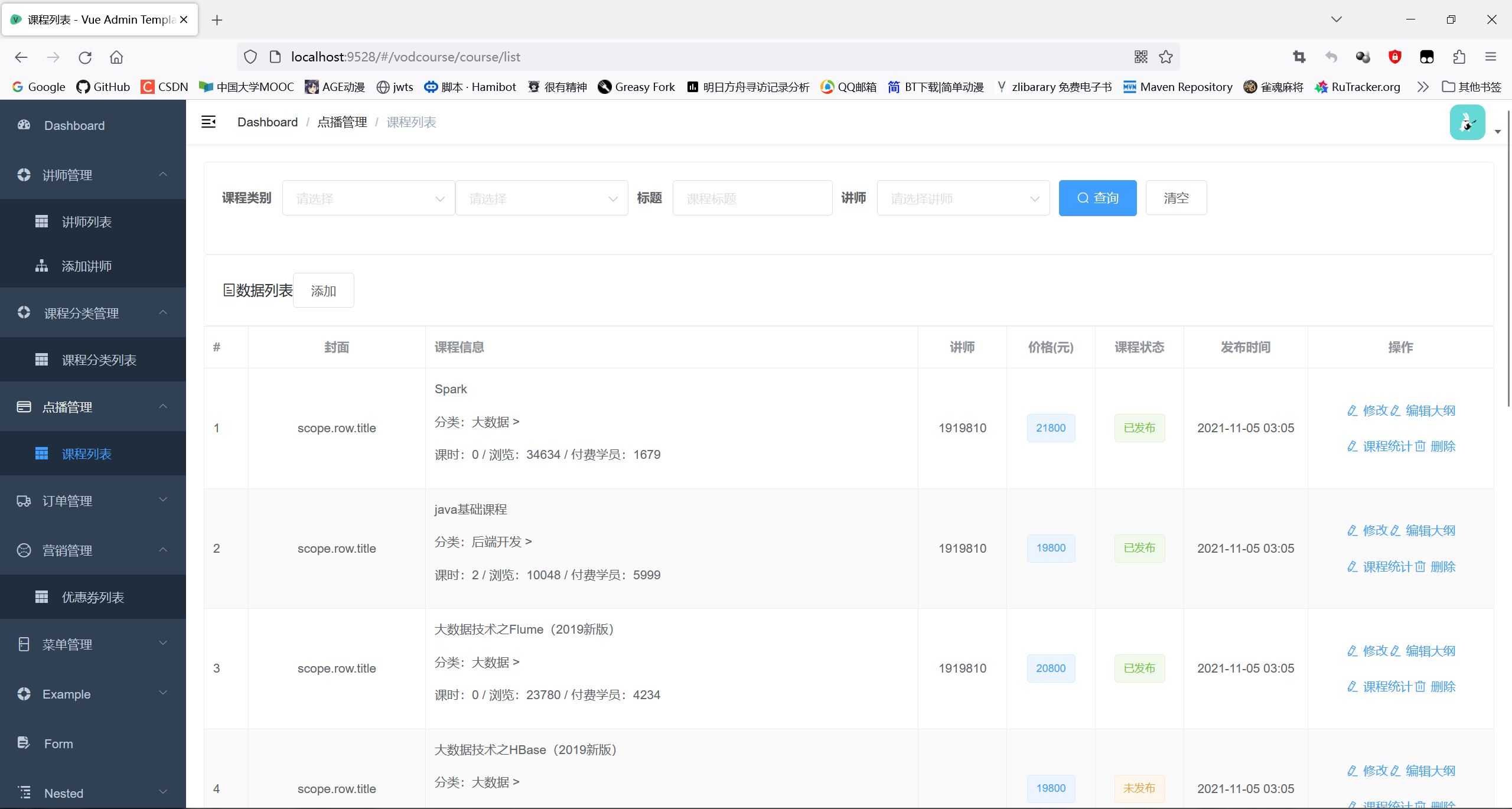
优惠券后台管理



讲师管理



课程分类列表

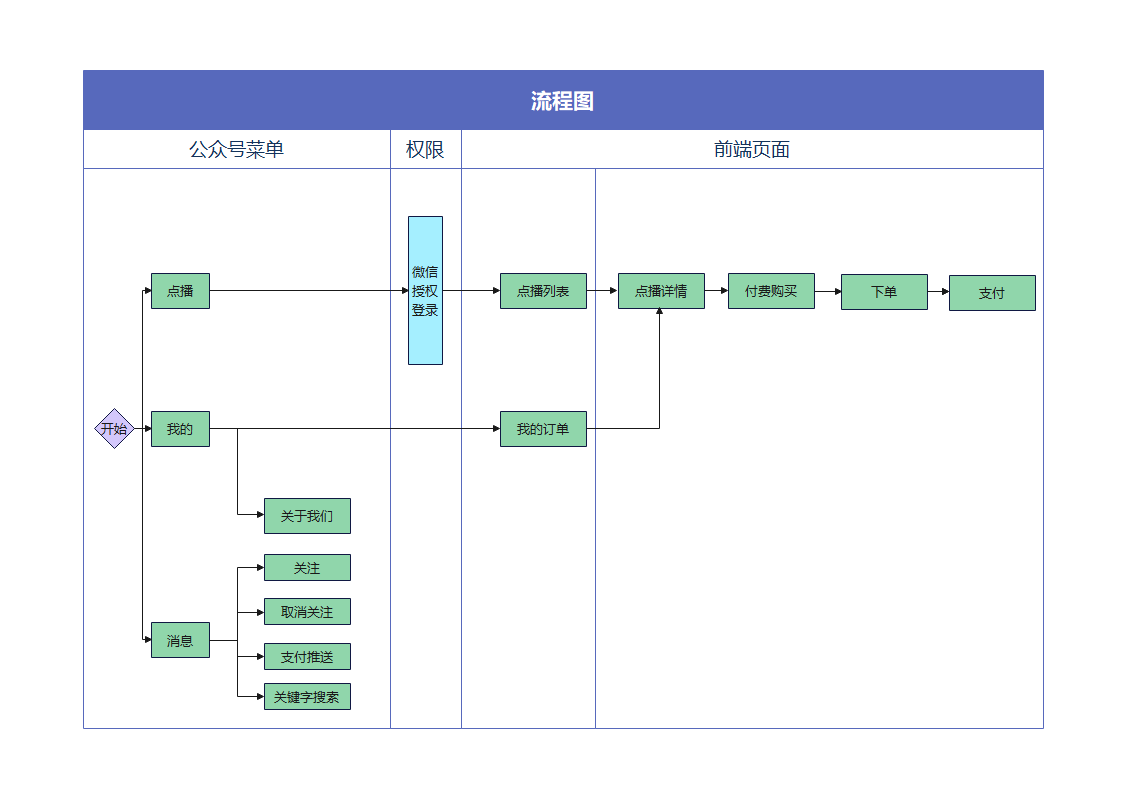


课程列表

1. 服务模式创新

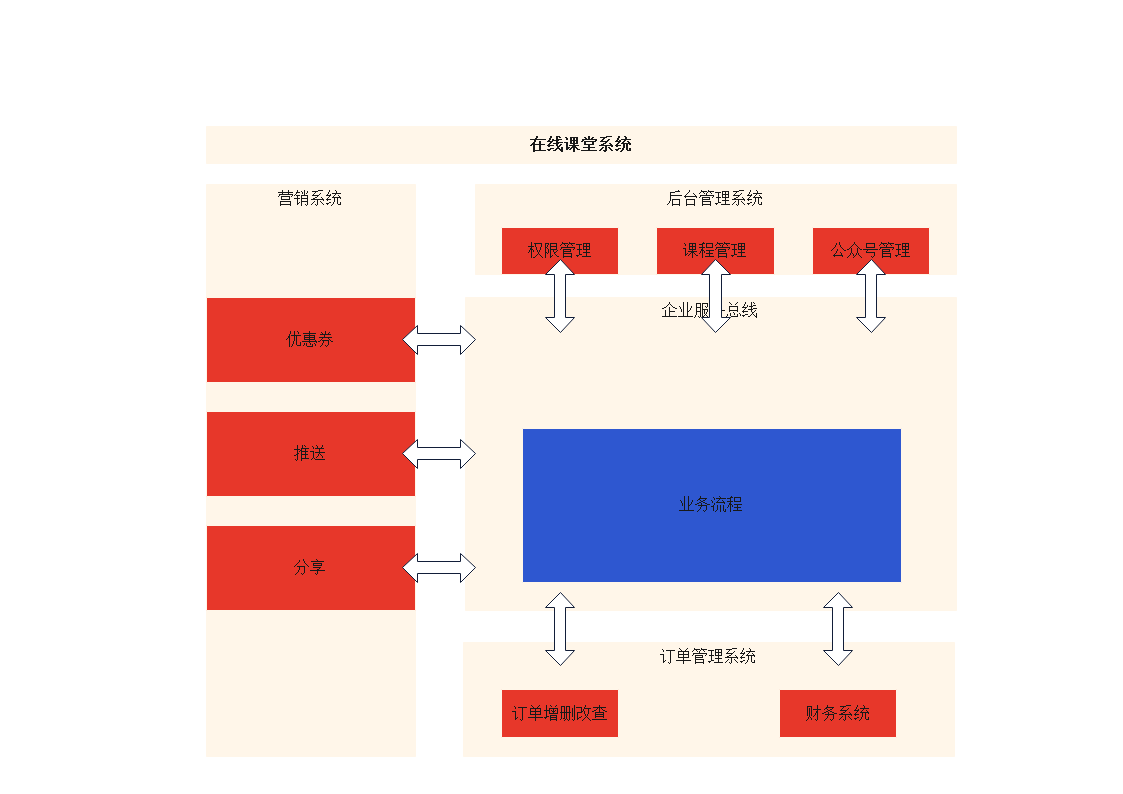
3.1业务流程建模

现有的业务结构（As-is）没有促销与推广以及更灵活的身份绑定，这样的结果会存在一些问题。例如，如果客户的微信号因为某种原因不能使用,将无法通过其他方式访问已经购买的课程,同时也没有分享等功能,和优惠券,免费试看等推广措施。因此我们实际的业务流程优化为（To Be）如下所示：

****

to-be

我们利用业务流程分析方法以及 SOMA 方法论提出的服务建模三个过程：服务发现、服务规约以及服务实现的实现步骤重新对系统服务进行分析并拆分为三个子系统，引入 ESB 企业服务总线来提高项目开发效率以及更好的对服务进行组合与重用：



3.2服务架构设计和服务组合:

根据对业务流程进行分析,以下是一个基于SOA设计理念的系统架构设计：

前端:

平台管理端

微信公众号端

高可用负载均衡:

使用Nginx和Keelived实现高可用负载均衡

统一接口网关:

使用Spring Cloud Gateway实现统一接口网关

服务治理:

使用Nacos进行服务注册和配置

中间件:

使用分布式文件系统作为存储

使用Redis集群作为缓存中间件

使用消息队列（如RabbitMQ）进行异步通信

API基础服务集群:

包括讲师管理、分类管理、课程管理等基础服务

腾讯云集成:

使用腾讯云提供的服务，如课程大纲、直播管理、订单管理、文件服务、短信服务等

服务监控:

使用Sentinel进行服务熔断和限流

使用SkyWalking进行分布式链路追踪

使用ELK（Elasticsearch、Logstash、Kibana）进行日志收集和分析

直播平台和点播平台:

使用Spring Boot构建直播平台和点播平台

实现直播配置、视频点播等功能

使用欢拓云直播进行直播功能支持

用户管理:

实现用户管理功能

数据平台:

使用MySQL数据库集群存储数据

使用Redis集群作为缓存

使用消息队列RabbitMQ进行异步处理

使用ELK进行日志收集和分析

系统部署和DevOps:

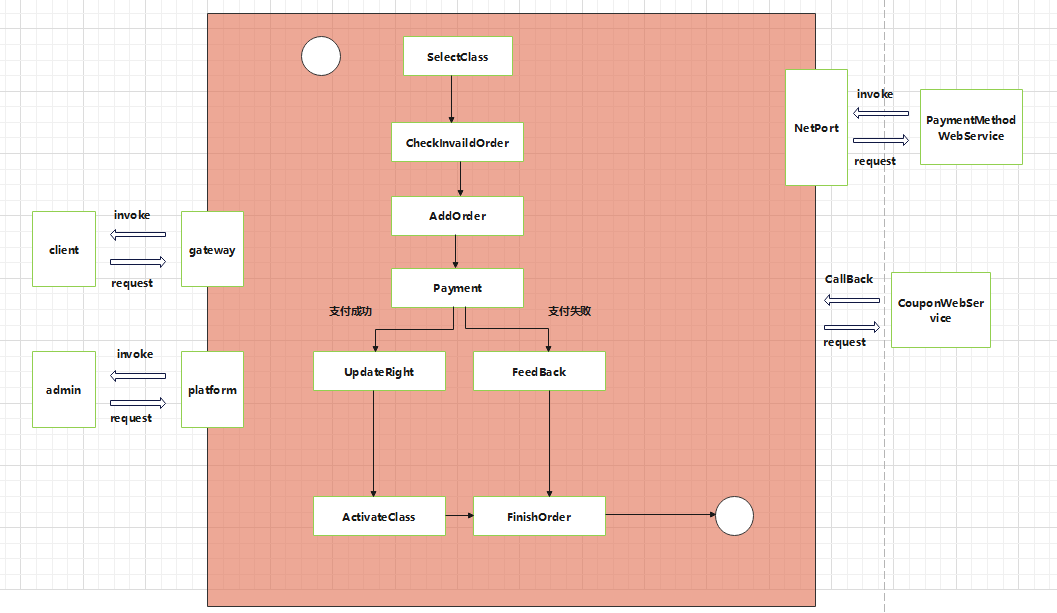
使用Docker进行容器化部署

使用Kubernetes（K8S）进行容器编排和管理

使用Jenkins进行持续集成和部署



技术架构图



服务组合模型

1. 总结

在疫情期间，为了满足“停课不停学”的需求并解决在线学习平台不够成熟的问题，采用业务流程管理的方法进行业务流程创新，构建了一个全方位的在线学习平台。该平台通过服务模式创新，解决了现有业务结构的问题，包括限制访问渠道、缺乏分享功能和优惠措施等。通过个性化推荐、社交化推荐、增加互动性、课程多样性、实时性和用户反馈等功能，提供了更好的用户体验和学习效果。基于已购课程的精准投放优惠券提高了优惠券的使用率和用户忠诚度。系统采用SOA设计理念，利用ESB企业服务总线实现服务组合与重用，通过统一接口网关、服务治理和中间件等技术支持实现高可用性和扩展性。整个系统架构涵盖了前端、负载均衡、统一接口网关、服务治理、中间件、API基础服务集群、腾讯云集成、服务监控、直播平台和点播平台、用户管理、数据平台、系统部署和DevOps等模块，实现了全面的在线学习功能。

1. 附录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **系统名称** | **业务流程名称** | **业务子流程名称** | **服务名称** | **初始服务构件名称** | **优化服务构件名称** |
| **后台管理系统** | **用户管理** | **购买课程** | **选择课程** | **SelectClass** |  |
|  |  |  | **使用优惠券** | **UseCoupon** |  |
|  |  |  | **新增订单** | **AddOrder** |  |
|  | **微信授权登录** |  | **授权登录** |  | **Authorize** |
|  | **课程管理** | **修改课程信息** | **新增课程** | **AddClass** |  |
|  |  |  | **更新课程** | **UpdateClass** |  |
|  |  |  | **删除课程** | **DeleteClass** |  |
|  |  | **视频管理** | **上传视频** | **UploadVideo** |  |
| **营销系统** | **优惠券** |  | **优惠券发放** |  | **CouponAllocate** |
|  |  |  | **优惠券信息** |  | **CouponInfo** |
|  |  |  | **优惠券使用** |  | **CouponUse** |
|  | **推送** | **同类推送** | **检索历史** |  | **SearchHistory** |
|  |  |  | **推送选择** |  | **SelectNew** |
|  |  | **联想推送** | **检索足迹** |  | **TrackFootprint** |
|  |  |  | **推送选择** |  | **SelectNew** |
| **订单管理系统** | **订单管理** | **支付金额** | **计算订单金额** |  |  |
|  |  |  | **选择支付方式** |  |  |
|  |  | **修改订单** | **修改订单** | **UpdateOrder** |  |
|  |  | **删除订单** | **删除订单** | **DeleteOrder** |  |
|  |  | **完成订单** | **完成订单** | **FinishOrder** |  |
|  |  | **订单结果** | **更新权限** | **UpdateRight** |  |
|  |  |  | **激活课程** | **ActivateClass** |  |
|  |  |  | **失败反馈** | **FeedBack** |  |
| **财务系统** | **订单汇总** |  | **订单信息** | **OrderInfo** |  |
|  | **支付接口** |  | **微信支付接口** | **WXPaymentAPI** | **PaymentAPI** |
|  |  |  | **支付宝支付接口** | **ZFBPaymentAPI** |  |

优惠券控制器代码

package cn.edu.hit.zxkt.activity.controller;

import cn.edu.hit.zxkt.activity.service.CouponInfoService;

import cn.edu.hit.zxkt.model.activity.CouponInfo;

import cn.edu.hit.zxkt.model.activity.CouponUse;

import cn.edu.hit.zxkt.result.Result;

import cn.edu.hit.zxkt.vo.activity.CouponUseQueryVo;

import com.baomidou.mybatisplus.core.metadata.IPage;

import com.baomidou.mybatisplus.extension.plugins.pagination.Page;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiParam;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

/\*\*

\* <p>

\* 优惠券信息 前端控制器

\* </p>

\*

\* @author Tsukinai

\* @since 2023-5-15

\*/

@RestController

@RequestMapping(value = "/admin/activity/couponInfo")

public class CouponInfoController {

@Autowired

private CouponInfoService couponInfoService;

@ApiOperation(value = "获取优惠券")

@GetMapping("get/{id}")

public Result get(@PathVariable String id) {

CouponInfo couponInfo = couponInfoService.getById(id);

return Result.ok(couponInfo);

}

@ApiOperation(value = "新增优惠券")

@PostMapping("save")

public Result save(@RequestBody CouponInfo couponInfo) {

couponInfoService.save(couponInfo);

return Result.ok(null);

}

@ApiOperation(value = "修改优惠券")

@PutMapping("update")

public Result updateById(@RequestBody CouponInfo couponInfo) {

couponInfoService.updateById(couponInfo);

return Result.ok(null);

}

@ApiOperation(value = "删除优惠券")

@DeleteMapping("remove/{id}")

public Result remove(@PathVariable String id) {

couponInfoService.removeById(id);

return Result.ok(null);

}

@ApiOperation(value = "根据id列表删除优惠券")

@DeleteMapping("batchRemove")

public Result batchRemove(@RequestBody List<String> idList) {

couponInfoService.removeByIds(idList);

return Result.ok(null);

}

//优惠券分页列表方法

@ApiOperation(value = "获取分页列表")

@GetMapping("{page}/{limit}")

public Result index(

@ApiParam(name = "page", value = "当前页码", required = true)

@PathVariable Long page,

@ApiParam(name = "limit", value = "每页记录数", required = true)

@PathVariable Long limit) {

Page<CouponInfo> pageParam = new Page<>(page, limit);

IPage<CouponInfo> pageModel = couponInfoService.page(pageParam);

return Result.ok(pageModel);

}

//获取已经使用优惠券列表

@ApiOperation(value = "获取分页列表")

@GetMapping("couponUse/{page}/{limit}")

public Result index(

@ApiParam(name = "page", value = "当前页码", required = true)

@PathVariable Long page,

@ApiParam(name = "limit", value = "每页记录数", required = true)

@PathVariable Long limit,

@ApiParam(name = "couponUseVo", value = "查询对象", required = false)

CouponUseQueryVo couponUseQueryVo) {

Page<CouponUse> pageParam = new Page<>(page, limit);

IPage<CouponUse> pageModel = couponInfoService.selectCouponUsePage(pageParam, couponUseQueryVo);

return Result.ok(pageModel);

}

}

ALS算法代码

import java.util.HashMap;

import java.util.Map;

public class ALSAlgorithm {

private Map<Integer, Map<Integer, Double>> ratings;

private int numUsers;

private int numItems;

private int numFactors;

private double[][] userFactors;

private double[][] itemFactors;

public ALSAlgorithm(Map<Integer, Map<Integer, Double>> ratings, int numUsers, int numItems, int numFactors) {

this.ratings = ratings;

this.numUsers = numUsers;

this.numItems = numItems;

this.numFactors = numFactors;

userFactors = new double[numUsers][numFactors];

itemFactors = new double[numItems][numFactors];

}

public void train(int numIterations, double lambda, double alpha) {

// Randomly initialize user and item factors

initializeFactors();

for (int iteration = 0; iteration < numIterations; iteration++) {

// Update user factors

for (int userId = 0; userId < numUsers; userId++) {

for (int factor = 0; factor < numFactors; factor++) {

double numer = 0.0;

double denom = lambda;

for (Map.Entry<Integer, Double> entry : ratings.get(userId).entrySet()) {

int itemId = entry.getKey();

double rating = entry.getValue();

double prediction = predict(userId, itemId);

double error = rating - prediction;

numer += itemFactors[itemId][factor] \* error;

denom += itemFactors[itemId][factor] \* itemFactors[itemId][factor];

}

userFactors[userId][factor] = numer / denom;

}

}

// Update item factors

for (int itemId = 0; itemId < numItems; itemId++) {

for (int factor = 0; factor < numFactors; factor++) {

double numer = 0.0;

double denom = lambda;

for (Map.Entry<Integer, Map<Integer, Double>> entry : ratings.entrySet()) {

int userId = entry.getKey();

Map<Integer, Double> userRatings = entry.getValue();

if (userRatings.containsKey(itemId)) {

double rating = userRatings.get(itemId);

double prediction = predict(userId, itemId);

double error = rating - prediction;

numer += userFactors[userId][factor] \* error;

denom += userFactors[userId][factor] \* userFactors[userId][factor];

}

}

itemFactors[itemId][factor] = numer / denom;

}

}

}

}

public double predict(int userId, int itemId) {

double prediction = 0.0;

for (int factor = 0; factor < numFactors; factor++) {

prediction += userFactors[userId][factor] \* itemFactors[itemId][factor];

}

return prediction;

}

private void initializeFactors() {

// Randomly initialize user and item factors

// You can use a random number generator to assign initial values

// to the userFactors and itemFactors matrices.

}

public static void main(String[] args) {

// Sample usage

Map<Integer, Map<Integer, Double>> ratings = new HashMap<>();

ratings.put(1, new HashMap<>());

ratings.get(1).put(1, 5.0);

ratings.get(1).put(2, 3.0);

ratings.put(2, new HashMap<>());

ratings.get(2).put(1, 4.0);

ratings.get(2).put(2, 2.0);

int numUsers = 2;

int numItems = 2;

int numFactors = 2;

int numIterations = 10;

double lambda = 0.01;

double alpha = 0.01;

ALSAlgorithm als = new ALSAlgorithm(ratings, numUsers, numItems, numFactors);

als.train(numIterations, lambda, alpha);

double prediction = als.predict(1, 2);

System.out.println("Prediction: " + prediction);

}

}