福州大学

“微签”软件设计规格说明书

选　　题：微签

组　　别：第四组

组　　名：从零开始

成　　员：杨泽隆、房蕴锦、许雅晶、方浩宇、刘洋、丘英杰、刘心怡、李臻、王宇航

指导教师：程永利

2021 年 10 月 28 日

目录

1. 导言

1.1目的

编写此文档的目的是进一步定制“微签”软件开发的细节问题，希望能使本软件开发工作更具体。为了使用户、软件开发者及分析和测试人员对该软件的初始规定有一个共同的理解，它说明了本软件的系统体系结构设计、构件设计及数据设计等，探讨每个模块该如何具体实现，每个实现中需要哪些算法、属性、参数、数据结构、接口，确定了系统的详细功能模块和数据结构，为下阶段开发工作提供依据。

1.2文档约定

本文档按以下要求和约定进行书写：

1、标题最多分三级，分别为黑体小二、黑体三号、黑体小三，标题均加粗。

2、正文字体为宋体小四，行距 20 磅。

3、需求优先级说明：每个需求陈述都有其自己的优先级。

1.3项目范围

“微签”适用于帮助老师打造一个智能精准的课堂签到环境，利用各种反作弊功能精准确定学生的出勤情况。

1.4编写依据

《软件工程实践选题报告》

《“微签”需求分析报告》

2. 总体描述

2.1产品愿景

微签是一款轻量化的签到软件，体积小、效率高、易上手。 微签将成为教师课堂的好帮手，培养学生学习习惯的好伙伴。

2.2产品特性

微签是一款体积小、效率高、易上手的软件。同时具有人脸识别、定位、签到码实时更新等多种反作弊手段。

对教师而言，能够利用微签打造一个智能精准的课堂签到环境，营

造良好课堂出勤氛围。对学生而言，能够防止学生作息紊乱、避免迟到旷课现象，帮助学生养成良好的作息，正常出勤。

2.3用户类型和特征

用户包括教师和学生两种类型。

教师具有以下特征：由于传统的课堂签到模式占用课堂教学时间，影响教学任务，同时不可避免的出现代签代课现象，使得课堂出勤存在虚假性。教师一直希望能有一种工具来解决上述问题。

学生具有以下特征：部分学生可能出于某些原因，企图旷课，或者代签代课，“微签”可以很大程度上避免代签代课，同时可以精准的找到未出勤学生，让学生按时出勤，认真上课。

2.4操作环境

前端环境	微信小程序
后端环境	Ubuntu 18
	Python 3

2.5设计和实施约束

时间约束	开发时间约为6周
人员约束	前端4人，后端4人
技术约束	适用微信开发者工具和python语言进行开发

2.6假设和依赖

本项目是否能够成功实施，主要依赖于以下的假设：

（1）团队成员的积极合作配合，为了项目的开发和实施，对个人时间进行合理规划同时为团队做出合理牺牲，配合队友完成任务。

（2）团队成员积极学习，努力掌握先进的能够适用于该项目的技术，这是系统的性能是否优化和项目能否成功的保证。

（3）团队项目的设计是否合理，这是实现难易程度的根本。

（4）假设项目需求在确定后不会有较大的改动。

（5）假设成品能极大程度地还原原型设计。

3. 体系结构设计

3.1体系结构风格

“微签”小程序采用层次体系结构和面向对象体系结构。

总体上采用三层体系结构：用户界面层、逻辑架构层、数据层。

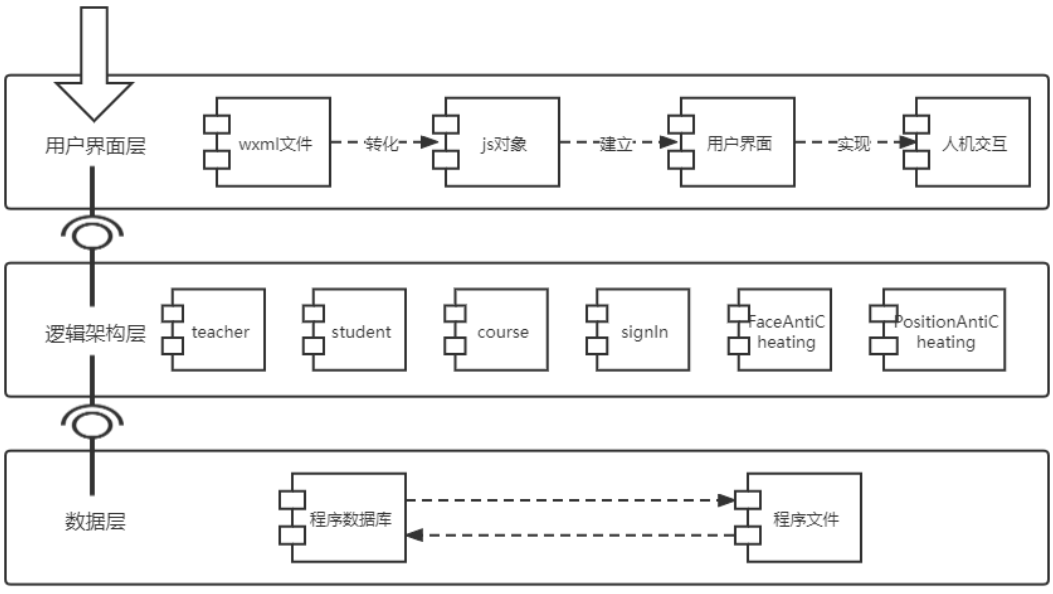
用户界面层：将多个 wxml 文件转化为相应的 js 对象，建立用户界面，完成人机交互。

逻辑架构层：通过接口建立与其余两层的联系并且完成核心功能开发，包括教师端基本应用，学生端基本应用，以及对应课堂数据创建参与逻辑，以及签到基础逻辑其中包含提供反作弊功能的人脸识别与定位应用。

数据层：存放该程序的数据库、文件。

系统的构件封装了数据和必须用于控制该数据的操作，构件间通过信息传递进行通信与合作。

3.2系统整体视图



4. 构件设计

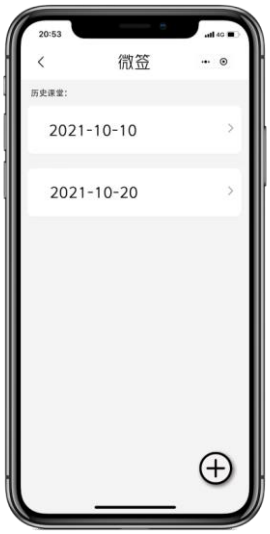
4.1用户界面层


20:53
微签
我是老师
我是学生


20:53
微签
课堂信息：
课程名
教师姓名


20:53
微签
已签到（2）
未签到（0）
杨泽隆
刘洋
结束签到

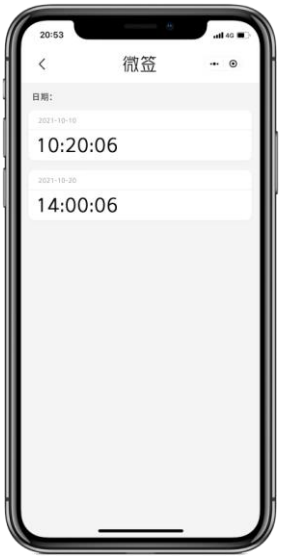

20:53
微签
我创建的课堂：
软工实践A班
软工实践B班


20:53
微签
历史课堂：
2021-10-10
2021-10-20


20:53
微签
个人信息：
学号
姓名


20:53
微签
我加入的课堂：
软工实践
数据库系统原理


20:53
微签
输入课堂码
PIN


20:53
微签
日期：
10:20:06
14:00:06

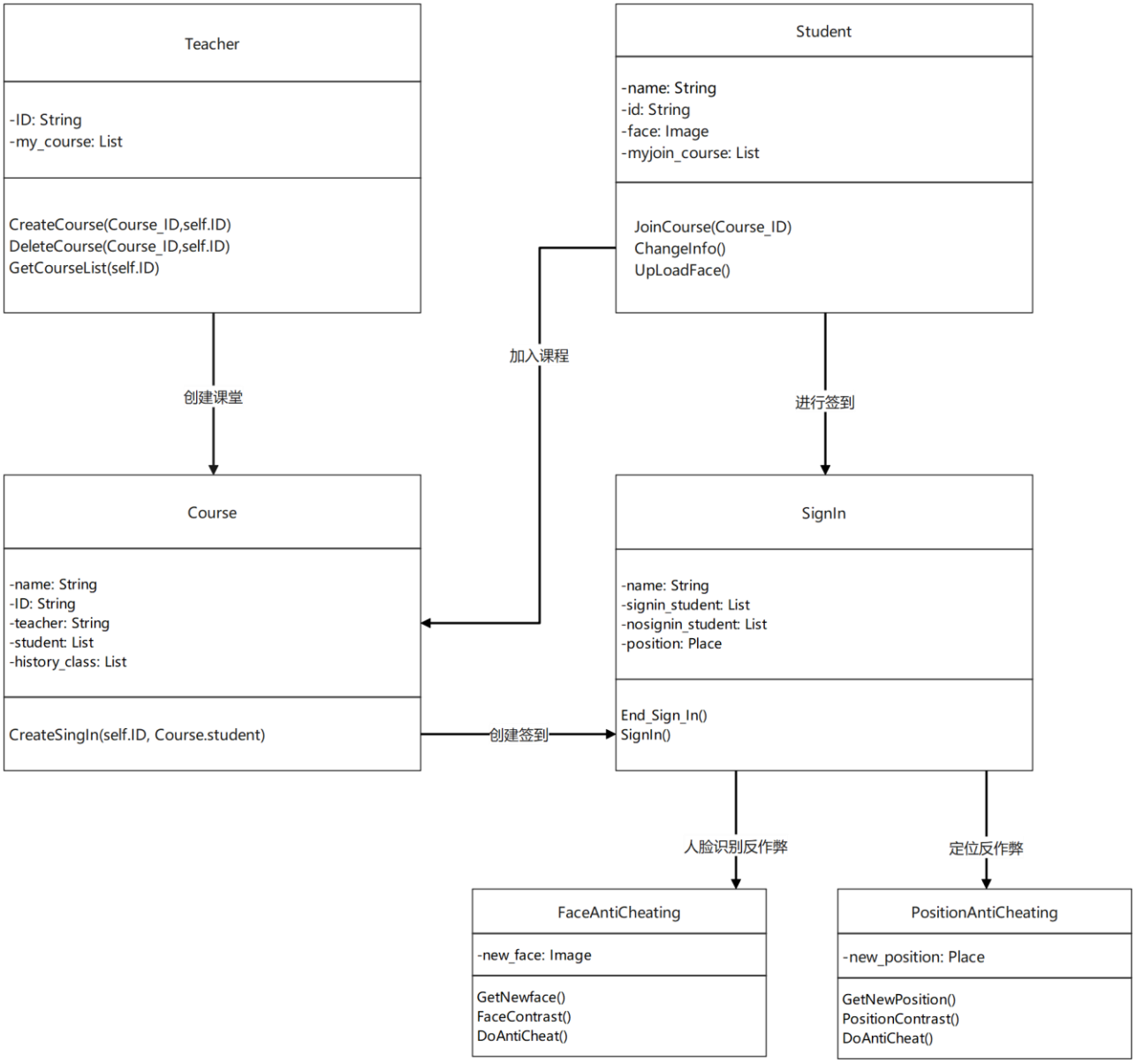

20:53
微签
签 到


20:53
微签
签到成功
2021-10-10 10:20:06

4.2逻辑架构层

4.2.1总类图

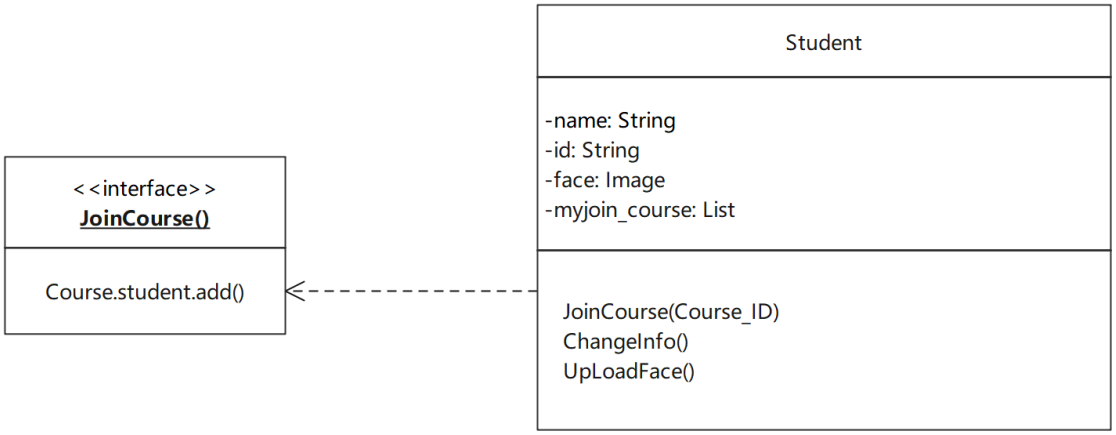
下图描述了逻辑架构层的各个类及之间的关系



4.2.2 各个构建的详细设计

1. 学生（Student）

构件图：

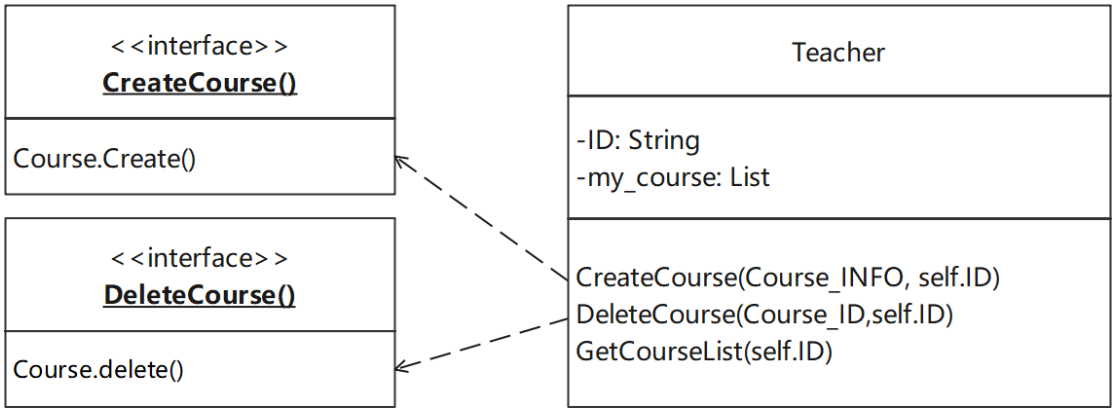


关键算法描述：

方法：	JoinCourse
描述：	输入 Course_ID，数据库中匹配 Course_ID，并将 Student.ID 加入 Course.student。
输入：	Course_ID
返回值：	加入成功；加入失败（Course.ID 不存在）；加入失败（Student.name 已存在 Course.student 中）

2. 教师（Teacher）

构件图：



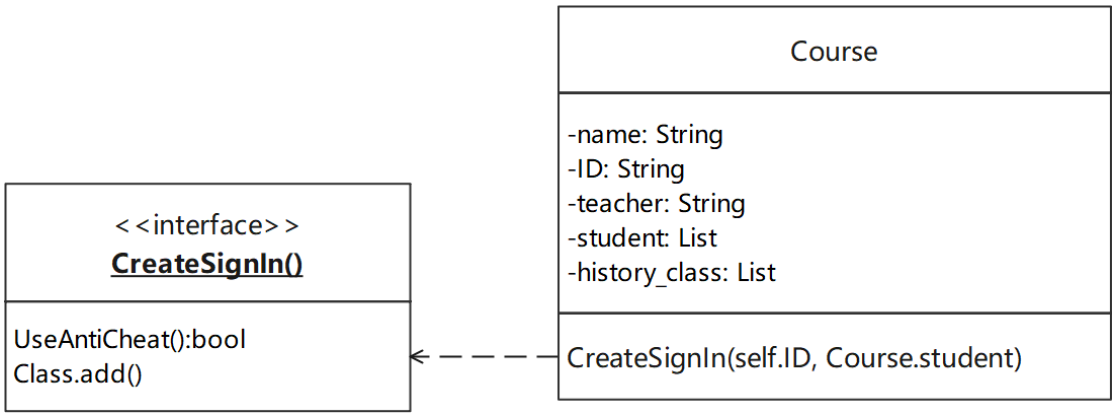
关键算法描述：

方法：	CreateCourse
描述：	输入课程的信息,Teacher.ID，并在数据库中插入一个 Course 并初始化。
输入：	Course_INFO,Teacher.ID
返回值：	新建成功（Course.ID）

方法：	DeleteCourse
描述：	根据 Course.ID,Teacher.ID，数据库中匹配 Course，删除。
输入：	Course_ID,Teacher.ID
返回值：	删除成功

3. 课程（Course）

构件图：

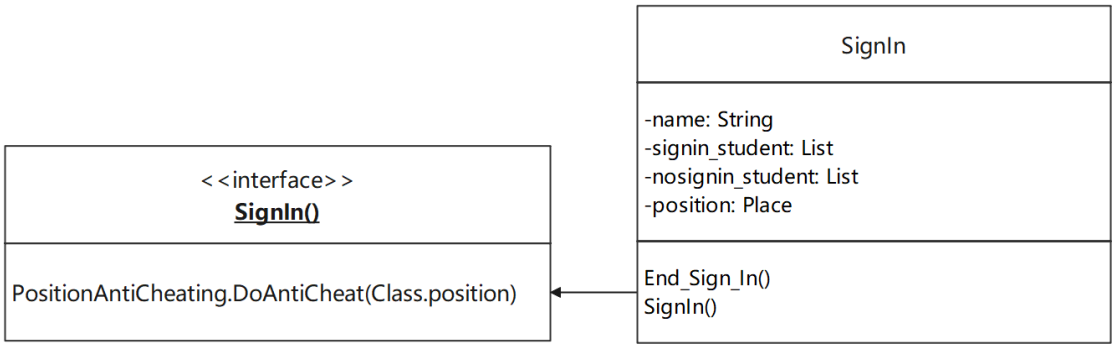


关键算法描述：

方法：	CreateSignIn
描述：	在数据库中新建一场签到并初始化。
输入：	可选参数：UseAntiCheat(Face,Position)
返回值：	新建成功；新建失败

4. 签到（SignIn）

构件图：

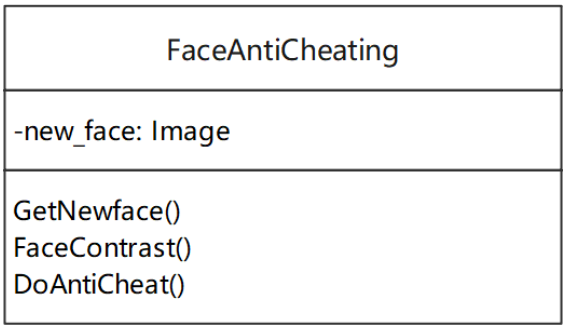


关键算法描述：

方法：	SignIn
描述：	将学生的 ID 传给 SignIn 类，若有反作弊验证，则同时上传相关数据（如人脸数据、位置信息等），进行签到，将 Student.ID 加入 SignIn.signin_Student。
输入：	Student.ID [可选参数：Student.Face, Student.position]
返回值：	签到成功；签到失败

5. 人脸识别（FaceAntiCheating）

构件图：

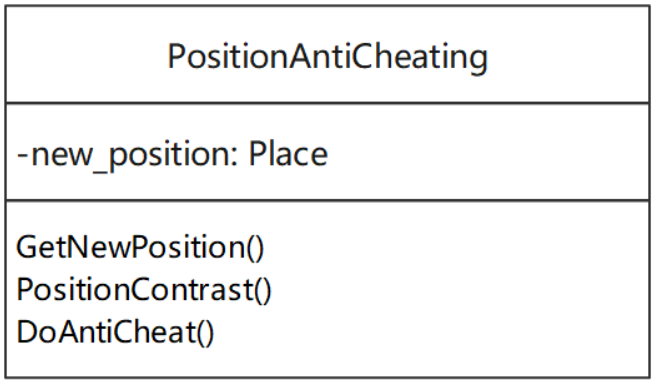


关键算法描述：

方法：	FaceAntiCheating.DoAntiCheat
描述：	获取 NewFace 与 Student.face 对比，误差在合理范围内则通过检验。
输入：	self.face
返回值：	检验成功；检验失败（请重新上传照片）

6. 定位（PositionAntiCheating）

构件图：



关键算法描述：

方法：	PositionAntiCheating.DoAntiCheat
描述：	获取当前定位，与签到（SignIn）定位对比，误差在合理范围内则通过检验。
输入：	self.position
返回值：	检验成功；检验失败（请重新定位）

4.3用户界面层与逻辑架构层之间的接口定义

接口	描述	输入参数	返回值
joinCourse	学生加入课堂	CourseID	课程信息或空 Json 格式字符串
studentChangInfo	学生修改个人信息	StudentID StudentINFO	修改成功或失败
studentSignIn	学生签到	CourseID StudentID	成功或失败
getCourseList	学生课程列表	StudentID	课程信息列表 Json 格式字符串
getHistorySignIn	学生查看历史签到记录	CourseID StudentID	历史签到记录列表 Json 格式字符串
Create_Course	教师创建课堂	TeacherID	Course.ID 课程信息 Json 格式字符串
getTCourseList	教师查看课程列表	TeacherID	课程信息列表 Json 格式字符串
getTHistorySignIn	教师查看历史签到	CourseID TeacherID	历史签到记录列表 Json 格式字符串
getStatistics	教师查看签到统计	CourseID TeacherID Date	已签到学生 未签到学生 Json 格式字符串

5. 数据设计

5.1概念结构设计

文字描述：

(1). 用户有学生和教师。

(2). 学生包括学生 ID、姓名、学号和人脸信息属性。

(3). 教师有教师 ID。

(4). 一个学生可以选修多门课程，每门课程可以供多个学生选修。

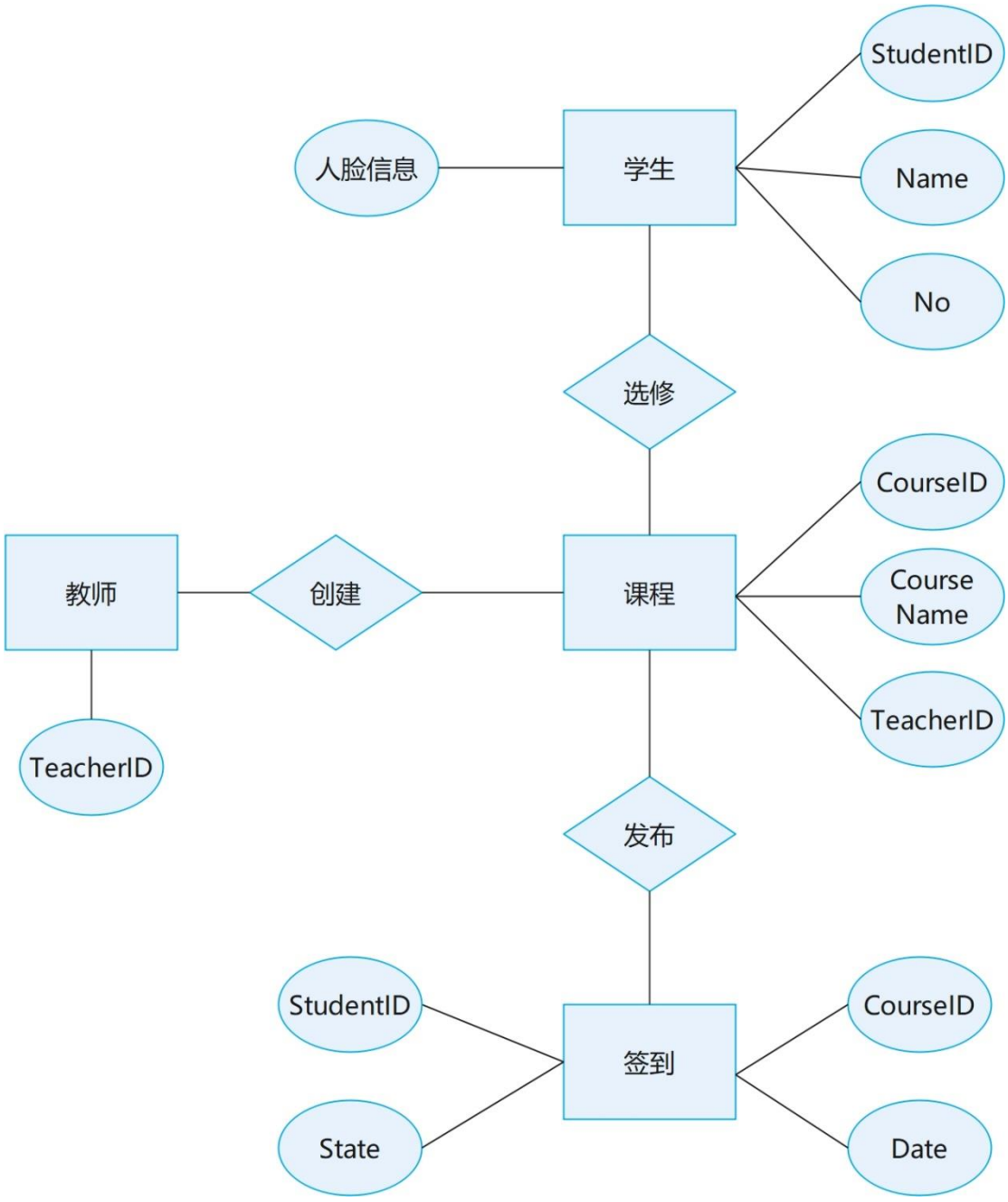
(5). 课程包括课程 ID、课程名、教师 ID 属性。

(6). 一门课程由一个教师负责。

(7). 每门课程可以发布多次签到。

(8). 签到包括签到日期、课程 ID、学生 ID、学生签到状态。

微签数据系统 E-R 图



5.2逻辑结构设计

进行 E-R 模型向关系模型的转换，抽象出以下关系模式：

学生(StudentID, Name, No, Face)

教师(TeacherID)

选修(StudentID, CourseID)

签到(CourseID, Date, StudentID, State)

课程(CourseID, CourseName, TeacherID)

5.3 物理结构设计

采用 MySQL 做为数据库管理系统。数据库底层采用 B+树和散列表（HashTable）的方法实现索引，能够明显的提高查询效率。

6. 安全性设计

6.1账户管理

保证用户权限的正确设置，防止出现越权行为，保证平台的正常良好运行。账户的产生、修改、变更、删除以及身份认证应采用统一的微信身份认证平台来实现。

6.2 加密

不使用自创加密方法，使用正确的算法和密钥大小，确保加密密钥的安全。要有足够的防御能力，防止用户信息被不良分子窃取。

6.3 敏感数据保护

应用软件应包含数据安全设计：包括数据库的安全、数据采集、数据传输、数据处理、数据存储、数据备份和恢复的安全。对重要的、敏感数据应进行加密和完整性保护。处理诸如人脸数据等用户私人信息应该采取专门的步骤，来确保这些数据的保密性，并确保其不被修改。