**《课表管理系统》**

**一、系统架构设计**

课程管理系统采用多层架构设计，包括表现层、业务逻辑层、数据访问层和数据库层，各层之间相互协作，实现系统功能。

**（一）表现层**

**系统总体结构:图示

描述已自动生成**

**用例图:**

**图示

描述已自动生成**

1. **设计目标**  
   为用户（管理员、教师、学生）提供简洁、直观、易用的交互界面。通过图形化界面元素，使用户能够方便地执行各种操作，如信息查询、数据输入、选课、成绩管理等。
2. **技术选型**  
   采用 Web 界面技术，如 HTML、CSS、JavaScript 等构建前端页面。结合前端框架（如 Vue.js、React.js 等）实现动态交互效果和组件化开发，提高开发效率和页面性能。

**（二）业务逻辑层**

1. **设计目标**  
   处理系统的核心业务逻辑，包括课程信息管理逻辑、课程安排逻辑、选课逻辑、成绩管理逻辑、用户权限管理逻辑等。它是连接表现层和数据访问层的桥梁，负责协调各层之间的数据交互和业务处理流程。
2. **功能模块划分**
   * **课程管理模块**：处理课程的添加、修改、删除、查询等操作相关的业务逻辑。例如，在添加课程时，要验证课程编号的唯一性、检查课程信息的完整性等。
   * **课程安排模块**：根据教学资源和课程要求，实现课程安排的算法和逻辑。包括检查教师和教室的可用性、避免时间冲突等功能。
   * **选课管理模块**：处理学生选课和退选的业务逻辑，如检查选课冲突、更新选课信息、处理选课人数限制等。
   * **成绩管理模块**：负责教师录入成绩、成绩计算、成绩查询和修改的业务逻辑。根据设定的成绩计算规则（如平时成绩和考试成绩的权重）计算总成绩，并对成绩修改进行权限验证。
   * **用户权限管理模块**：根据用户类型（管理员、教师、学生）分配不同的操作权限，确保系统的安全性和数据的保密性。例如，管理员具有最高权限，可以对所有功能进行操作；教师只能操作与自己授课相关的功能；学生只能进行选课、查询成绩等操作。

**（三）数据访问层**

1. **设计目标**  
   负责与数据库进行交互，实现对数据库中数据的增、删、改、查操作。将业务逻辑层的请求转换为对数据库的具体操作，并将结果返回给业务逻辑层。通过抽象数据访问接口，提高系统的可维护性和可扩展性，方便在不同的数据库环境下进行切换。
2. **技术选型**  
   使用数据库访问框架，如 MyBatis（Java 环境下）或其他适合的框架。这些框架可以方便地编写 SQL 语句与 Java 对象（或其他编程语言的对象）之间的映射关系，减少代码的复杂性和重复性。

**（四）数据库层**

1. **设计目标**  
   存储系统的所有数据，包括课程信息、教师信息、学生信息、课程安排信息、选课信息、成绩信息等。设计合理的数据库结构，保证数据的完整性、一致性和高效存储。
2. **数据库选型**  
   选择关系型数据库，如 MySQL、Oracle 或 SQL Server 等。这些数据库具有成熟的技术支持、强大的数据存储和管理能力，适合处理课程管理系统这种结构化数据较多的应用场景。

**二、数据库设计**

**（一）数据库概念模型设计（E - R 模型）**

图示

描述已自动生成

**（二）数据库表结构设计**

**类图：**

图示

描述已自动生成

1. **课程表（Course）**
   * **CourseID（主键）**：课程编号，唯一标识一门课程，如 “C001”。
   * **CourseName**：课程名称，如 “高等数学”。
   * **Credit**：学分，如 “3”。
   * **ClassHour**：学时，如 “48”。
   * **Description**：课程简介，对课程内容的简要描述。
   * **CourseType**：课程类型，取值为 “必修”、“选修”、“公共课” 等。
2. **教师表（Teacher）**
   * **TeacherID（主键）**：教师编号，如 “T001”。
   * **TeacherName**：教师姓名，如 “张三”。
   * **Title**：职称，如 “教授”。
   * **AvailableTime**：可授课时间，以特定格式存储教师的空闲时间，如 “周一 1 - 2 节，周三 3 - 4 节”。
   * **Contact**：联系方式，如手机号码或邮箱。
3. **学生表（Student）**
   * **StudentID（主键）**：学生编号，如 “S001”。
   * **StudentName**：学生姓名，如 “李四”。
   * **Grade**：年级，如 “2023 级”。
   * **Major**：专业，如 “计算机科学与技术”。
   * **Contact**：联系方式。
4. **教室表（Classroom）**
   * **ClassroomID（主键）**：教室编号，如 “R001”。
   * **Capacity**：教室容量，可容纳的学生人数，如 “50”。
   * **Equipment**：设备信息，描述教室中的教学设备，如 “投影仪、多媒体设备”。
5. **授课关系表（Teach）**
   * **TeacherID（外键）**：关联教师表。
   * **CourseID（外键）**：关联课程表。
   * **TeachingTime**：授课时间，具体的上课时间安排，如 “周二 3 - 4 节”。
6. **选课关系表（Select）**
   * **StudentID（外键）**：关联学生表。
   * **CourseID（外键）**：关联课程表。
   * **SelectionTime**：选课时间，记录学生选课的时间戳。
   * **Grade**：成绩，存储学生在该课程的成绩（初始值可为空）。
7. **课程安排表（Arrange）**
   * **CourseID（外键）**：关联课程表。
   * **ClassroomID（外键）**：关联教室表。
   * **ClassTime**：上课时间，如 “周四 1 - 2 节”。

**三、系统接口设计**

**（一）用户接口**

1. **管理员接口**
   * **课程信息管理界面**：提供课程信息的添加、修改、删除、查询操作入口。通过表格形式展示课程信息，可进行单个或批量操作。
   * **教师管理界面**：用于创建教师账号、修改教师信息、查看教师授课安排等操作。
   * **学生管理界面**：实现学生账号的创建、信息修改、学生选课情况查看等功能。
   * **系统统计界面**：以图表或报表形式展示课程选修情况、教师工作量、学生成绩分布等统计信息。
2. **教师接口**
   * **授课信息界面**：教师可以查看自己所授课程的详细信息，包括课程安排、学生名单等。
   * **成绩管理界面**：提供成绩录入、修改、查询功能。支持以班级为单位或单个学生为单位进行成绩操作。
3. **学生接口**
   * **选课界面**：展示可选课程列表、课程详细信息，提供选课和退选功能按钮。
   * **成绩查询界面**：学生可以查看自己所选课程的成绩。

**（二）外部接口**

1. **与学校其他系统的接口（可选）**  
   如果学校已经存在其他相关系统，如学籍管理系统、财务管理系统等，可以设计接口实现数据共享和交互。例如，与学籍管理系统接口获取学生的基本信息，避免重复录入；与财务管理系统接口实现学费与课程选修的关联（如根据选修课程计算学费）。
2. **第三方登录接口（可选）**  
   考虑到用户使用的便捷性，可以设计与第三方平台（如学校统一认证平台、微信、QQ 等）的登录接口，用户可以使用已有的第三方账号登录课程管理系统，减少注册流程。

**四、原型图设计**

**图片包含 图形用户界面

描述已自动生成**

**1、登录 / 注册页面**

**图形用户界面, 应用程序, Word

描述已自动生成**

* **登录部分：**
  + 包含用户名 / 邮箱输入框，有相应的提示文字（如 “请输入用户名 / 邮箱”），输入框可设置为必填项且有格式校验（例如邮箱格式是否正确）。
  + 密码输入框，显示为加密形式（黑点或星号），同样设置为必填项，可添加 “忘记密码” 链接，点击可跳转到密码找回流程页面。
  + “登录” 按钮，点击后进行登录验证，若信息正确则跳转到系统主页面，错误则给出相应提示（如 “用户名或密码错误，请重新输入”）。
* **注册部分：**
  + 用户名、邮箱、密码、确认密码输入框，每个输入框都有明确的提示要求和格式校验。
  + “注册” 按钮，点击后将用户输入的注册信息发送到服务器进行注册验证，若信息合法且未被注册过，则注册成功并可直接登录系统，否则给出相应错误提示（如 “该用户名已被使用，请重新选择”）。

**2、系统主页面**

* **导航栏：**
  + 通常包含 “课程管理”“学生管理”“教师管理”“成绩管理”“系统设置” 等主要模块的导航链接，点击可切换到相应功能页面。
  + 可能还会有用户个人信息展示区域（如头像、用户名），点击可进入个人信息编辑页面。
* **信息展示区：**
  + 根据不同用户角色（管理员、教师、学生）展示不同的核心信息。例如 对于管理员，可能展示系统整体运行数据如课程总数、学生总数、教师 总数等概况信息；对于教师，可能展示其负责的课程列表及相关课程动态（如近期作业提交情况等）；对于学生，可能展示其已选课程列表及课程相关通知等。

**3、课程管理页面**

* **课程列表：**
  + 以表格形式呈现所有课程信息，包括课程名称、课程代码、授课教师、课程学分、开课学期等基本信息。
  + 每门课程可设置 “详情” 按钮，点击可查看该课程的详细信息（如课程大纲、教学目标、教材选用等）。
  + 对于管理员和教师角色，可能还会有 “编辑”“删除” 按钮，用于对课程信息进行修改和删除操作（需进行相应权限验证）。
* **课程添加 / 编辑页面：**
  + 包含课程名称、课程代码、授课教师（可通过下拉菜单选择已注册教师）、课程学分、开课学期、课程大纲、教学目标、教材选用等一系列课程相关信息的输入框或选择框。
  + “保存” 按钮，点击后将输入的课程信息保存到数据库，若信息完整且合法则保存成功并返回课程列表页面，否则给出相应提示（如 “课程名称不能为空，请重新输入”）。

**4、学生管理页面**

* **学生列表：**
  + 同样以表格形式列出所有学生的基本信息，如学号、姓名、性别、专业、年级等。
  + 每个学生记录可设置 “详情” 按钮，查看学生的详细个人资料（如联系方式、家庭住址等）。
  + 对于管理员角色，可能有 “编辑”“删除” 按钮用于对学生信息进行管理（需符合相关规定和权限验证）。
* **学生添加 / 编辑页面：**
  + 有学号、姓名、性别、专业、年级、联系方式、家庭住址等学生相关信息的输入框或选择框。
  + “保存” 按钮，点击后将新添加或修改后的学生信息保存到数据库，成**功则返回学生列表页面，失败则给出提示。**

**5、教师管理页面**

* **教师列表：**
  + 以表格形式展示教师的基本信息，如工号、姓名、性别、职称、所属专业等。
  + 每一位教师可设置 “详情” 按钮，查看教师的详细个人资料（如教学经历、研究方向等）。
  + 对于管理员角色，有 “编辑”“删除” 按钮用于教师信息的管理（需进行权限验证）。
* **教师添加 / 编辑页面：**
  + 包含工号、姓名、性别、职称、所属专业、教学经历、研究方向等教师相关信息的输入框或选择框。
  + “保存” 按钮，点击后将新添加或修改后的教师信息保存到数据库，成功则返回教师列表页面，失败则给出提示。

**6、成绩管理页面**

* **成绩录入页面：**
  + 首先通过下拉菜单选择要录入成绩的课程，然后会出现该课程下所有学生的学号、姓名以及成绩输入框。
  + “保存” 按钮，点击后将录入的成绩信息保存到数据库，若全部成绩都已录入且合法则保存成功并返回成绩管理主页面，否则给出相应提示（如 “请输入所有学生的成绩”）。
* **成绩查询页面：**
  + 可以通过选择课程、学号、姓名等条件进行成绩查询。查询结果以表格形式呈现，包括学号、姓名、课程名称、成绩等信息。

**7、系统设置页面**

* **用户权限设置：**
  + 以表格形式列出所有用户（管理员、教师、学生），并可设置每个用户的权限范围（如教师可对自己所授课程进行哪些操作等）。
  + “保存” 按钮，点击后将更新后的权限设置保存到数据库。
* **系统参数设置：**
  + 比如设置每学期的开学时间、放假时间、成绩评定标准等系统相关参数。
  + “保存” 按钮，点击后将新设置的参数保存到数据库。

**五、系统安全设计**

**（一）用户认证与授权**

1. **用户认证**  
   采用用户名和密码的方式进行用户认证。用户在登录界面输入账号和密码后，系统将对其进行验证。为了提高安全性，可以采用加密算法（如 MD5、SHA - 256 等）对密码进行加密存储和传输。同时，可以设置密码强度要求，如长度、包含字母和数字等。
2. **授权管理**  
   根据用户类型（管理员、教师、学生）在系统中设置不同的角色和权限。通过角色 - 权限映射关系，为每个用户分配相应的操作权限。例如，管理员角色具有所有功能的操作权限，教师角色只能操作与教学相关的功能，学生角色只能进行选课和成绩查询等操作。在系统实现过程中，可以使用权限管理框架（如 Spring Security 等）来实现细粒度的权限控制。

**（二）数据安全**

1. **数据加密**  
   对系统中的敏感数据，如学生成绩、教师联系方式、用户密码等进行加密存储。在数据传输过程中，采用安全的传输协议（如 HTTPS），确保数据在网络中的保密性和完整性。
2. **数据备份与恢复**  
   定期对数据库进行备份，可以采用全量备份和增量备份相结合的方式。同时，建立数据恢复机制，以便在出现数据丢失、损坏等情况时能够快速恢复数据。可以使用数据库自带的备份和恢复工具，也可以编写自定义的备份脚本。

**（三）系统防护**

1. **防止 SQL 注入攻击**  
   在数据访问层，对用户输入的数据进行严格的过滤和验证，防止恶意用户通过 SQL 注入攻击获取或篡改系统数据。可以使用参数化查询等技术来避免 SQL 语句的拼接，从而降低 SQL 注入的风险。
2. **防止 XSS 攻击**  
   在表现层，对用户输入的内容进行 HTML 编码处理，防止跨站脚本攻击（XSS）。同时，对从数据库中读取的数据在显示到页面之前也进行相应的编码处理，确保用户界面的安全性。