在线教育直播平台

概要设计说明书

2030416018 高歌

Version 1.0

2023.4.9

目录

1 设计目标

该系统是一个聚焦于直播教学的在线教育平台，除基本的课程管理、用户管理、基本的音视频直播、连麦、文字互动功能外，还为在线教育这一场景深度定制，提供屏幕分享、白板演示、在线答题、签到签退、课堂质量分析等功能，应能够适应绝大多数课堂教学需求，并实现相较传统课堂更好的教学效果。

本系统具有以下设计目标：

（1）高效的实时交互

通过使用 WebRTC 等实时音视频通信技术实现直播、连麦与实时文字聊天功能，确保教师与学生之间的实时交互顺畅，提升在线教育效果。

（2）灵活性和易用性

设计一个用户友好的界面，让教师和学生能够轻松上手并进行课堂互动。同时，为不同类型的课程提供灵活的配置选项，适应多种教学需求。

（3）定制化课程支持

针对不同的学习需求和目标，提供个性化的课程管理功能。允许师生上传或查阅课件、并支持在线直播答题、白板演示、签到签退功能。

（4）数据驱动优化

利用数据分析技术，收集和分析用户行为和教学效果数据，为师生提供更深入的课堂质量分析、学习情况分析等，帮助他们更好地调整教学方法或学习策略。

（6）可靠性与安全性

平台应在绝大多数时间内正常运行，并且确保较低的平均故障恢复时间。此外，平台应保护教师和学生的个人信息和数据安全，对所有隐私信息安全加密，确保教师和学生之间的互动信息不会被非法获取和使用。

（7）可扩展性与可维护性

平台应该能够方便地扩展以应对用户数量的变化和业务的发展。同时，平台也应保持易于维护，包括代码维护、数据备份和恢复等。

（8）跨平台支持

确保平台能够在多种设备和操作系统上顺畅运行，让用户无论在何种设备上都能轻松地访问和使用该平台。

2 系统架构

本系统采用典型的前后端分离设计。具体来说，分为前端、后端与数据层（数据库）三个主要部分。具体技术选型将在下一节介绍。

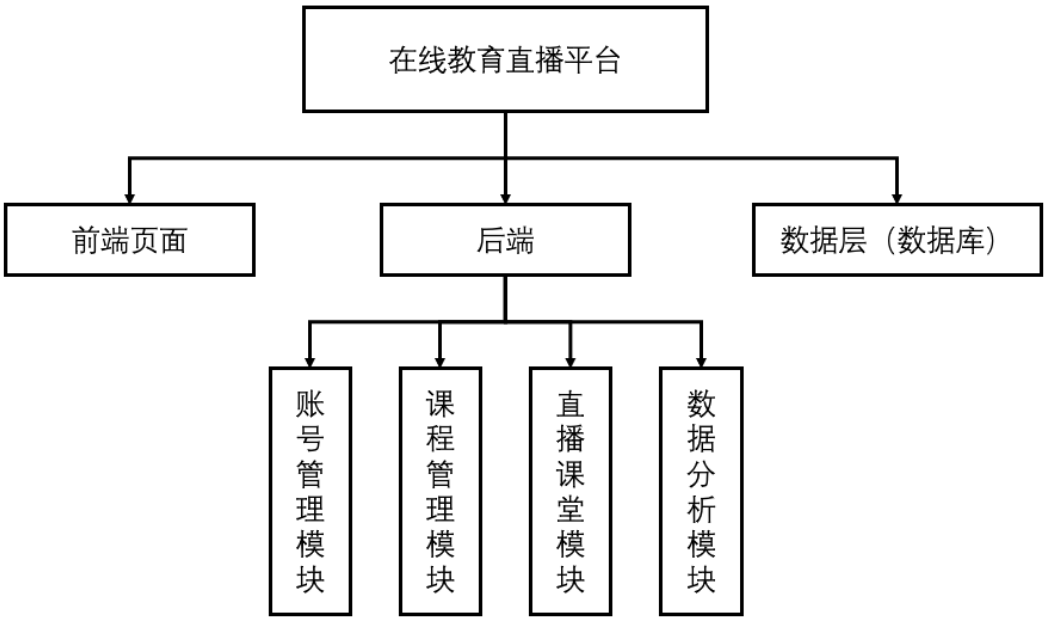

图 2-1 系统概观

在功能上，按领域将系统分为四个模块，它们分别是：

（1）账号管理模块：负责处理教师和学生的注册、登录、个人信息修改等功能。同时负责具体的角色管理（教师或学生）、权限分配等功能。

（2）课程管理模块：负责处理课程的创建、删除、编辑、成员管理、课件管理、题库管理等功能。

（3）直播课堂模块：负责直播课堂的创建、编辑、开启、关闭、音视频互动、文字互动、白板演示、在线答题、签到签退等功能。

（4）数据分析模块：该模块负责课堂数据的收集、处理、展示等功能，为师生生成精准直观的总结报告。

关于这几个功能模块的具体说明，将在第 4 节详细介绍。

3 核心技术设计

3.1 前端

前端使用 Astro 作为元框架（Meta-Framework），该框架提供了服务端渲染以加快客户端响应速度并增强对 SEO 的支持，同时提供了基于文件的路由以简化使用。

具体的 UI 框架使用 React，并使用 Ant Design 作为组件库。此外，为了简化自定义组件的编写，引入 SCSS 及 TailwindCSS。

在请求上，使用 Tanstack Query（前 React Query）作为请求缓存及统一数据处理层。由于后端采用 GraphQL 提供接口（下述），前端也需要采用一个 GraphQL 客户端对接，这里使用比较轻量的 GraphQL Request 作为此客户端，并使用 GraphQL Code Generator 生成类型定义文件以确保类型安全。

在直播方面，使用浏览器原生的 WebRTC API 以实现音视频直播。

在测试上，使用 Astro 推荐的 Vitest 编写测试。

3.2 后端

后端使用 Node.js 框架 NestJS。NestJS 使用依赖注入实现了类似 Java 中 Spring 框架的面向切面编程，同时，由于其基于 Node.js 生态，因此为本系统的技术栈（GraphQL、WebRTC 等）提供了更好的支持。

在鉴权方面，使用 NestJS 提供的 JWT 鉴权支持，进行权限管理，分为教师与学生两种角色。并且，使用该框架提供的 GraphQL 支持以提供常规的 GraphQL 接口，使用 graphql-ws 提供的基于 WebSocket 的 Subscription 实现以支持直播间的实时文字互动。接口文档使用 GraphQL Playground。

在数据库 ORM 框架上，使用 Node.js 生态中较成熟的 ORM 框架 TypeORM。具体的数据库实现则使用 MySQL。

在直播层面，使用 NestJS 提供的 Gateway 支持，使用 socket.io 作为其具体实现，用以向前端提供 WebRTC 支持。

在测试上，使用 NestJS 原生提供的 Jest 支持。

4 功能模块设计

4.1 账号管理模块设计

具体来说，该模块提供以下功能：

- 账号注册（需加密存储隐私信息）
- 登录（需确保接口的安全调用）
- 个人信息修改（需加密存储隐私信息）
- 角色管理（教师或学生）
- 接口鉴权（JWT）

下面使用顺序图展示账号注册及登录的基本流程。

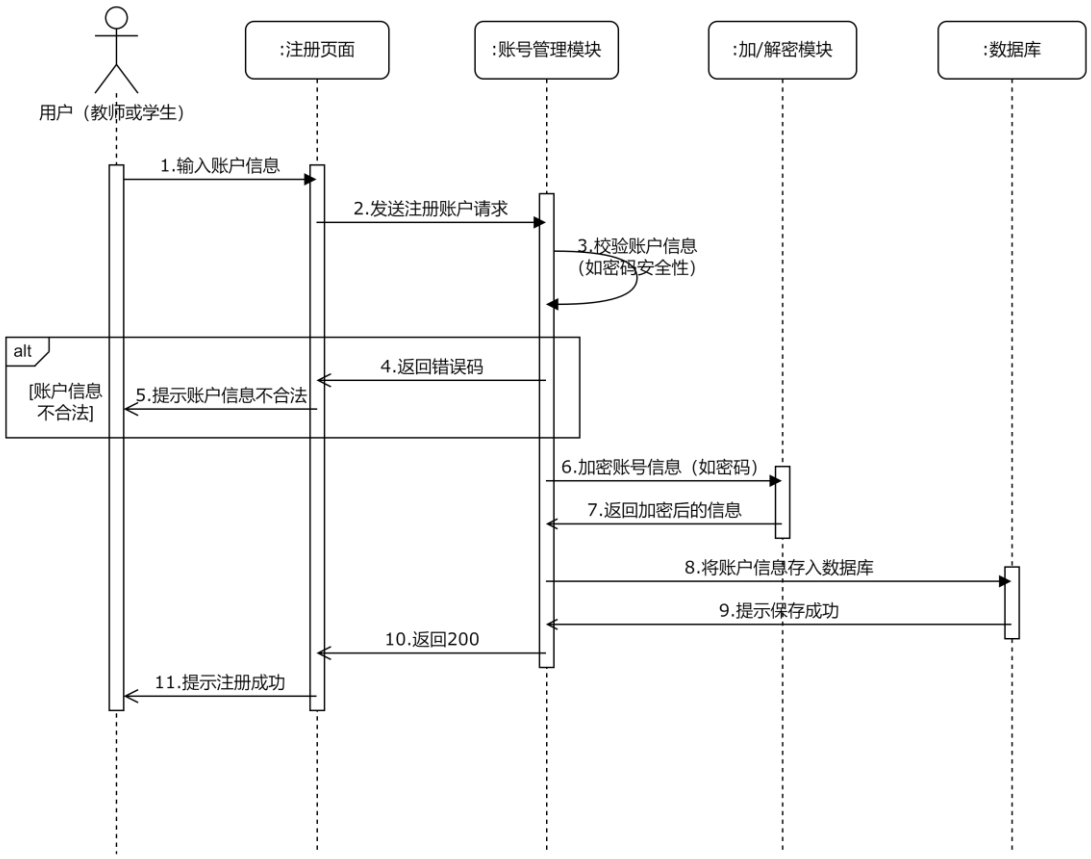

图 4-1 账号注册流程

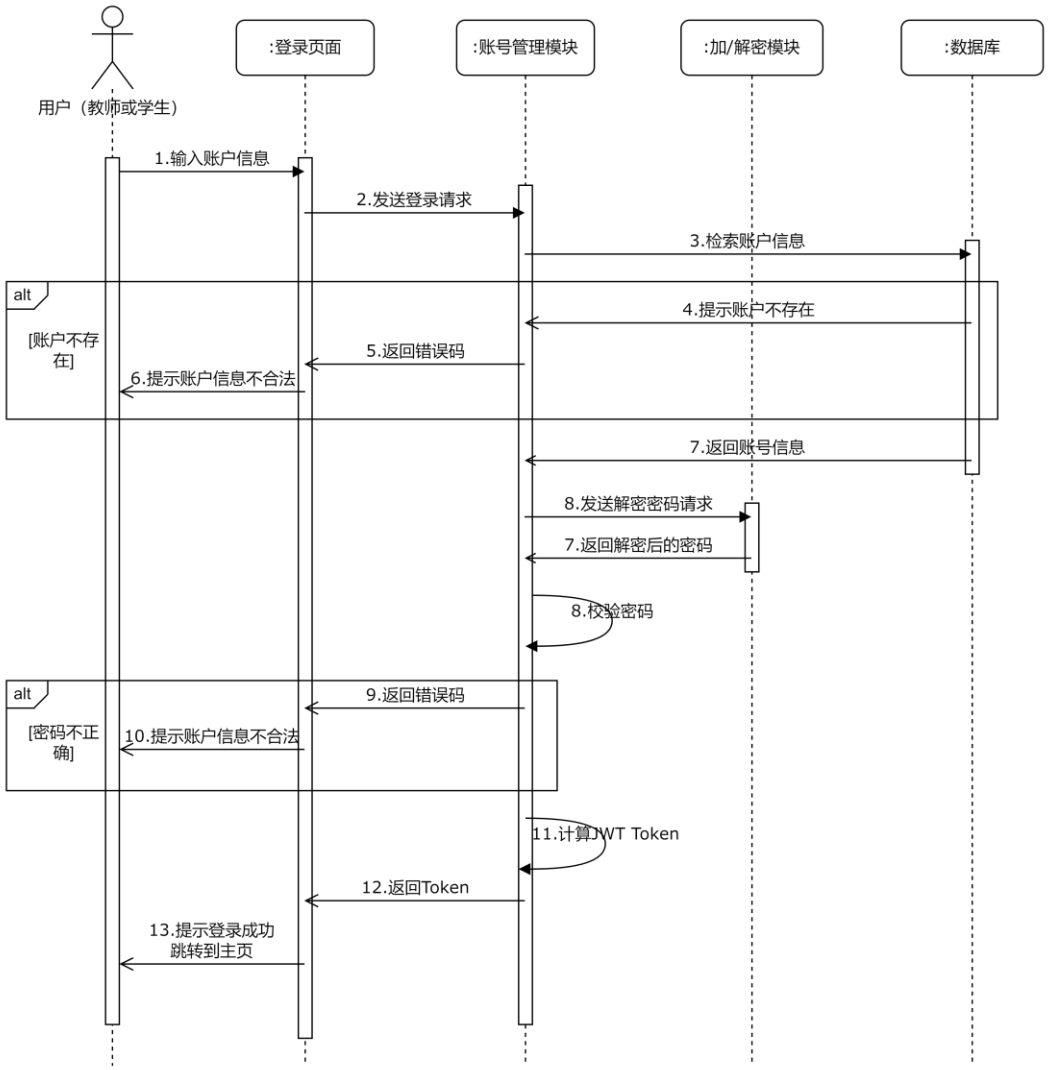

图 4-2 登录流程

登录成功后，前端应在全局状态中存储服务器返回的 JWT Token，并在每次请求时从响应体中取出更新后的 Token。在每次请求前，前端应校验 Token 是否已过期，如过期，则应提示用户重新登录。下面展示该鉴权流程。

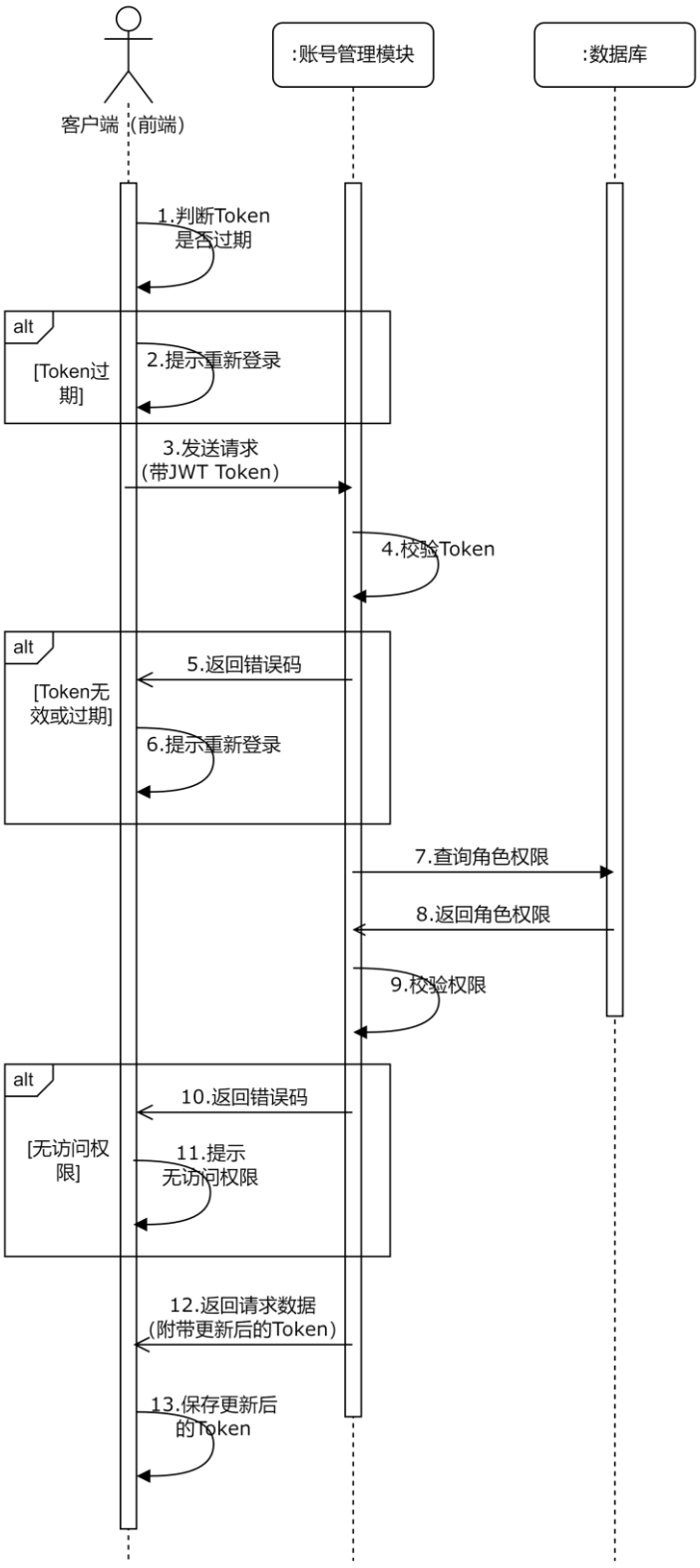

图 4-3 鉴权流程

如图所示，可以看到使用 JWT 鉴权较有效地减少了每次请求时的鉴权开销，并使用一套无状态的后端权限管理方案使得应用状态更易于管理。

4.2 课程管理模块设计

具体来说，该模块提供以下功能：

- 课程的创建（教师）
- 课程信息的修改（教师）
- 加入课程（学生）
- 课程成员管理（教师）
- 向课程中上传课件（教师）
- 向课程题库中上传题目（教师）
- 查看课件（教师及学生）
- 查看课程统计数据（教师及学生）

值得注意的是，本模块的“查看课程统计数据”功能仅将统计数据从数据库中获取并展示，不进行相关的统计计算。统计计算功能由数据分析模块完成。

这部分功能虽然琐碎，但都比较简单，这里不再一一单独描述。

4.3 直播模块设计

具体来说，该模块提供以下功能：

- 音视频推流
- 屏幕共享
- 连麦交流
- 白板演示
- 实时文字互动
- 在线题目作答
- 签到/签退

其中，“在线题目作答”功能比较复杂，在这里使用协作图简单描述该流程。

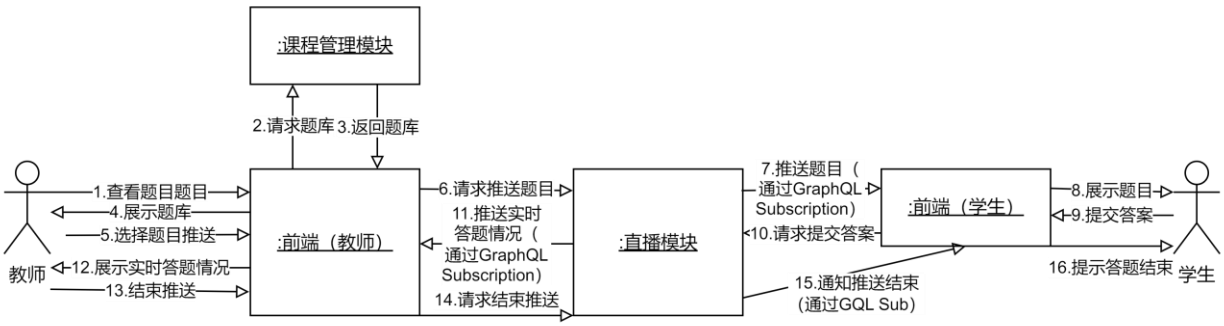

图 4-4 在线题目作答流程

在上图中，展示了教师选择题目、推送题目、学生作答，然后教师手动结束题目推送的流程。实际上，该推送过程还可以等待倒计时结束后自动结束，这里不再详述。

4.4 数据分析模块设计

具体来说，该模块应提供以下功能：

- 计算直播课堂的相关统计信息，包括答题情况、签到/签退情况等
- 生成教师端课堂总结报告
- 生成学生端学习报告

这部分内容总体上比较简单，在此不再详述。

5 接口设计

除视频推流外，所有接口均以 GraphQL 形式提供。总体上以 RESTful 风格安排资源的增、删、改、查操作：

- 所有创建资源的 Mutation 名称应以“create”开头，如“createUser”。传入参数必须命名为“input”，其类型应命名为“Create{资源名}Input”，如“CreateUserInput”。
- 所有获取资源的 Query 应直接以资源名作为名称，并以单复数形式区分返回全部还是返回单个资源，如“users”和“user”。
  其中返回全部资源的 Query 应提供一致的分页接口，其 GraphQL 类型定义如下：

```
type Connection {
  pageInfo: PageInfo!
  edges: [Edge!]!
}

type PageInfo {
  hasNextPage: Boolean!
  hasPreviousPage: Boolean!
  startCursor: String
  endCursor: String
}

type Edge {
  cursor: String!
  node: YourNodeTypeHere!
}
```

  所有返回全部资源的 Query 都应返回一个 Connection 类型的对象。
  而所有返回单个资源的 Query，其第一个接受的参数必须命名为“id”，表示所请求资源的 ID。此类接口可包含其他可选参数。
- 所有修改资源的 Mutation 名称都应以“update”开头，如“updateUser”。第一个传入参数必须命名为“id”，表示所请求修改资源的 ID，然后是一个命名为“input”的参数，其类型应命名为“Update{资源名}Input”，表示修改的字段及值，如“UpdateUserInput”，该参数中的所有字段都应是可选的。
- 所有删除资源的 Mutation 名称都应以“delete”开头，如“deleteUser”。第一个传入参数必须命名为“id”，表示所请求删除资源的 ID。该接口可含其他可选参数。
- 对于直播间中的实时文字互动、题目推送等需要实时双向通信的情况，全部使用 GraphQL 提供的 Subscription 实现。此类接口名称必须为“onXxxSubscription”，如“onMessageAddedSubscription”，并接受一些必要

的参数，如“onMessageAddedSubscription”应至少接受“lessonId”作为参数。对于此类接口的参数命名，不做强制要求。

除上述提到的几种基本接口类型外，还有数据分析、签到、签退等不便于使用这几种方式表示的接口，应根据实际情况选择一个尽量清晰的接口名，并且提供足够的文档信息。

6 数据结构设计

数据库大致设计如下图所示：

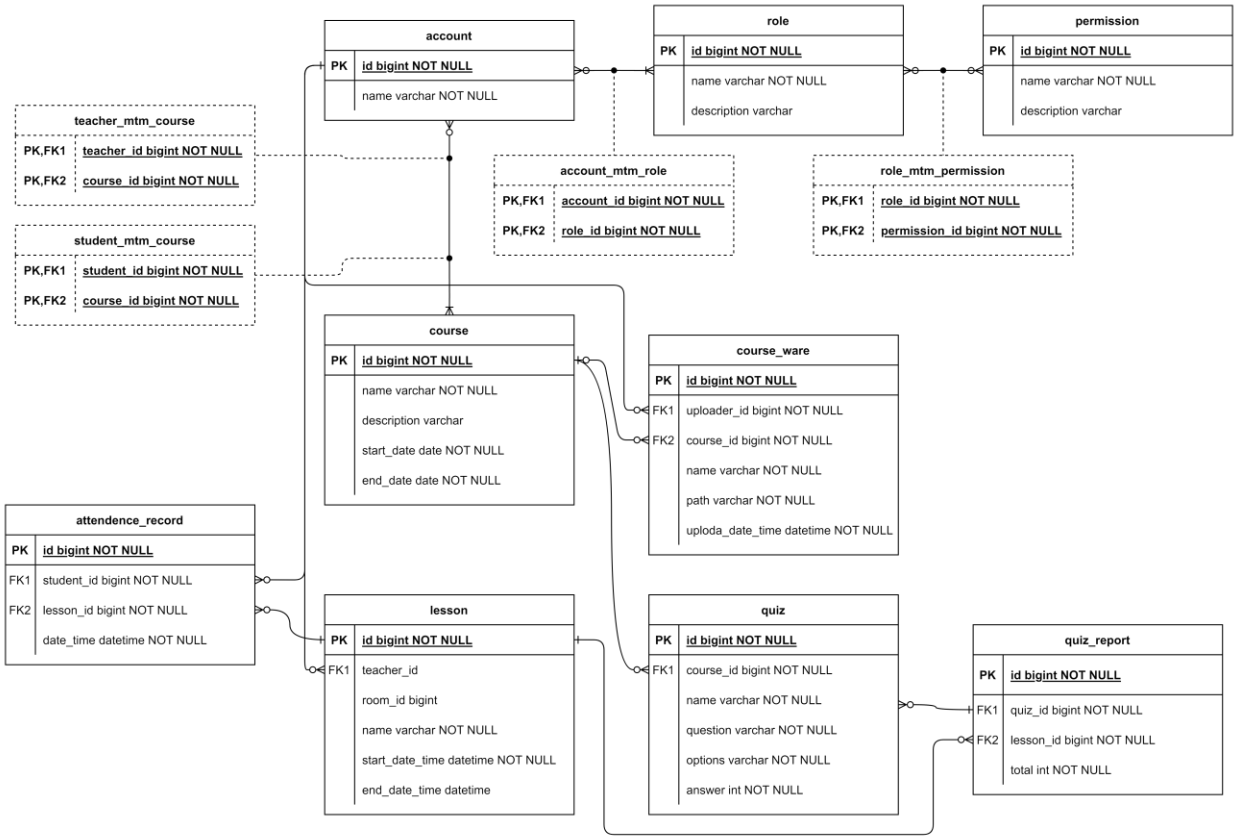

图 6-1 数据库 E-R 图

数据库整体上与需求说明书中的类图相对应。但是一些与直播相关的对象，如直播间，由于只在服务器内存中保存，因此不映射到数据库中。

值得注意的是这里 lesson 表中，room_id 是可 NULL 的，因为这里的直播间 ID 是保存在内存中的，当直播结束后就会销毁，因此需要在直播结束后将 room_id 设为 NULL；end_date_time 同理，由于直播结束前不能确定结束时间，因此其一开始为 NULL。

7 运行环境

7.1 开发环境

本系统的开发环境如表 7-1 所示。

表 7-1 开发环境

操作系统	Windows 11
开发语言	JavaScript
Node.js 版本	18 及以上
数据库	MySQL 5.0 及以上
开发工具	VS Code 1.76 / WebStorm 2023.1

前后端依赖库具体版本见实际代码库的 package.json 文件。

7.2 部署环境

系统实际部署环境应视平台实际用户量考虑。大致上，采用 Docker 进行容器化部署，视用户规模考虑是否使用 Kubernetes 进行自动化运维。

考虑到 Node.js 的跨平台特性，系统应能在 Windows Server 及 Linux 主流发行版上正常部署。这里给出建议的最低配置要求。

表 7-2 建议的部署环境最低配置要求

操作系统	Ubuntu 18.04 及以上版本（仅建议）
内核版本	Linux 4.15 及以上
Node.js 版本	18 及以上
数据库	MySQL 5.0 及以上
内存	4 GB 以上
CPU	Intel Xeon E3-1230 v6 及以上性能的 CPU 建议 4 核心及以上
硬盘容量	不低于 512 GB

此外，如后续需要大量持续性维护工作，应考虑使用 Jenkins 正确部署 CI/CD 服务。