详细设计文档

编写人员：贺思超、陈杰、陈增耀、何毅、江桀

指导老师：张健

编写日期：2023.7.14

1 概述

1.1 简述

本项目开发了一个智能算法驱动的线上应用，以解决传统体测中存在的问题，提供一体化的线上训练、体测、数据收集与管理系统的 web 用户与管理端和安卓用户端。

1.2 软件设计目标

利用现代移动网络、信息技术和人工智能技术，构建一个集合体质测试、课外锻炼业务的校园体育综合信息系统。

1.2.1 功能要求

- 用户注册登录：用户可以登录注册账户、修改密码、完善个人信息。
- 身份验证：用户可以上传身份证照片，通过摄像头上传个人照片，完成身份验证。
- 体测成绩上传：用户可以上传个人体测成绩。
- 线上体测：用户可以使用摄像头完成线上体测并上传成绩，查询体测成绩。
- 线上训练：用户可以使用摄像头完成线上训练并上传成绩，查询训练记录。

1.2.2 性能要求

系统应具备高并发处理能力，保证在用户数量激增的情况下也能保持流畅的使用体验。

同时，对于线上体测和训练的数据上传，系统应保证高效的数据处理和存储能力。

1.2.3 输入要求

系统需要接收的输入主要包括用户的注册登录信息，用户的身份证照片，线上体测和训练的视频数据等。

1.2.4 输出要求

系统的输出主要为用户的体测和训练结果，体测和训练记录，以及个人信息的显示等。

1.2.5 安全与保密要求

系统应保障用户的个人信息和体测训练数据的安全性，遵守相关的数据保护法规，对用户数据进行加密处理，防止数据泄露。对于用户的密码等敏感信息，应采取加密存储，保证安全性。

3 对象描述

3.1 逻辑后端描述

3.1.1 实体类（Entity Classes）：

在逻辑后端中，有三个实体类，分别是 Test、Train 和 User。这些实体类对应着系统中的数据库表，用于存储和表示相关的数据信息。

- Test 类：表示测试数据的实体，包含了与测试相关的属性，例如测试 ID、测试名称、测试结果等。它映射到数据库中的测试表，通过与 Mapper 的配合，可以进行测试数据的增删改查等操作。
- Train 类：表示训练数据的实体，包含了与训练相关的属性，例如训练 ID、训练名称、训练时间等。它映射到数据库中的训练表，通过与 Mapper 的配合，可以进行训练数据的增删改查等操作。
- User 类：表示用户数据的实体，包含了与用户相关的属性，例如用户 ID、用户名、密码等。它映射到数据库中的用户表，通过与 Mapper 的配合，可以进行用户数据的

增删改查等操作。

这些实体类在逻辑后端起到了存储和传递数据的作用，通过与数据库的交互，可以实现对数据库表中数据的操作和管理。

3.1.2 Mapper：

每个实体类都有对应的 Mapper，用于定义与数据库表之间的映射关系以及进行数据库操作。Mapper 中定义了各种 SQL 语句和方法，用于实现数据的持久化操作，包括插入、查询、更新和删除等。

在 TestMapper 中，可以定义例如 insertTest、deleteTest、selectTest 等方法，用于对测试表中的数据进行增删查操作。

在 TrainMapper 中，可以定义例如 insertTrain、deleteTrain、selectTrain 等方法，用于对训练表中的数据进行增删查操作。

在 UserMapper 中，可以定义例如 insertUser、deleteUser、selectUser 等方法，用于对用户表中的数据进行增删查操作。

通过 Mapper 的使用，可以将实体类与数据库表进行映射，并且通过调用 Mapper 中定义的方法，可以对数据库表中的数据进行各种操作。

3.1.3 Service 和 ServiceImplment：

逻辑后端还包含了 Service 和 ServiceImplment 组件。Service 层定义了业务逻辑接口，通过 Service 组件可以对实体类进行各种业务操作。ServiceImplment 是 Service 接口的具体实现，其中包含了具体的业务逻辑代码。

在 TestService 和 TestServiceImplment 中，可以定义例如 addTest、deleteTest、getTest 等方法，用于对测试数据进行添加、删除、获取等操作。

在 TrainService 和 TrainServiceImplment 中，可以定义例如 addTrain、deleteTrain、getTrain 等方法，用于对训练数据进行添加、删除、获取等操作。

在 UserService 和 UserServiceImplment 中，可以定义例如 addUser、deleteUser、getUser 等方法，用于对用户数据进行添加、删除、获取等操作。

通过 Service 和 ServiceImplment 的配合，可以将业务逻辑与数据库操作进行解耦，实现业务功能的独立性和可扩展性。

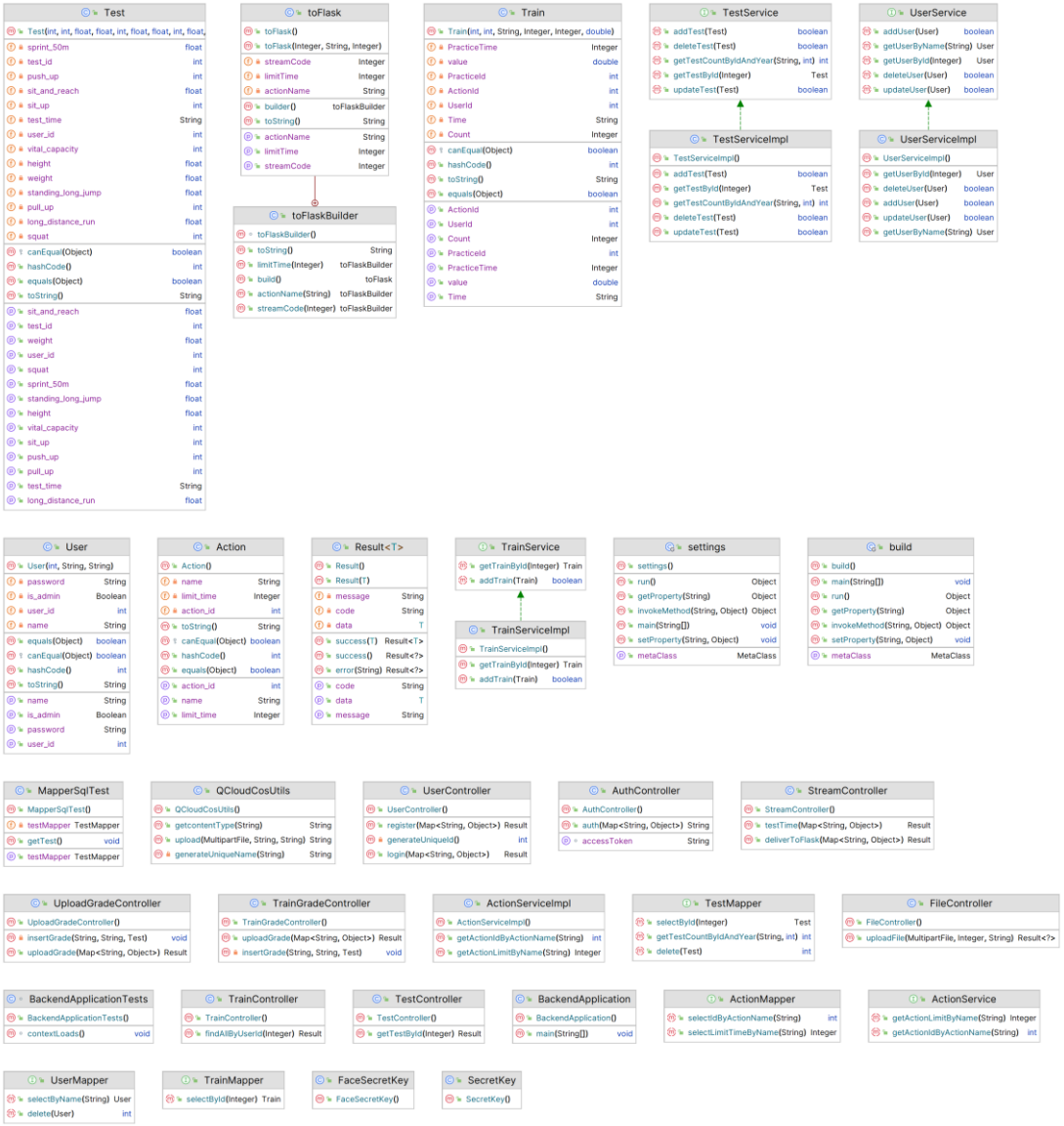
3.1.4 Utils

逻辑后端还包括一个 Utils 包，其中包含了一些方法类，用于处理数据库操作之外的内容。这些方法类可以提供一些通用的功能，例如腾讯云 COS 文件上传、百度智能云人脸比对等功能。通过 Utils 包中的方法类，可以方便地调用这些功能，并在逻辑后端中

完成相关的业务需求。

另外，在为了统一接口的返回值，逻辑后端定义了一个名为 Result 的类。Result 类用于封装操作成功和失败时的返回结果，包括状态码、消息和数据等信息。通过使用 Result 类，可以保持接口返回值的一致性，方便前端对接口返回结果的处理和解析。在每个接口的处理过程中，可以将操作结果封装为 Result 对象，并作为响应返回给前端。

这样的设计有助于规范接口的返回格式，减少重复代码的编写，提高代码的可维护性和可读性。通过统一的接口返回值类，前端可以更加方便地处理接口返回结果，进行错误处理、数据解析和展示等操作。



3.2 算法后端描述

3.2.1 姿势检测算法：

在算法后端中，使用 MediaPipe 库的姿势检测模块对每一帧进行姿势检测。通过姿势检测模块，可以检测到人体的关键点地标（landmarks）。这些关键点地标可以表示人体的姿势信息。

接下来，使用 poseembedding 将姿势的关键点地标转换为合适的格式，并利用 PoseClassifier 对姿势进行分类。PoseClassifier 是一个使用 KNN（K 最近邻）机器学习模型实现的人体姿态分类器。它基于一组姿势样本，通过嵌入向量和距离度量计算给定姿势与样本之间的相似度，并根据相似度进行分类和识别。

为了提高分类结果的稳定性，使用 EMADictSmoothing 对分类结果进行平滑处理。EMADictSmoothing 采用指数移动平均方法对分类结果进行平滑处理，以减少噪声和抖动。

最后，使用 RepetitionCounter 计算重复次数，得到最终的检测值。RepetitionCounter 用于跟踪和计算重复动作的次数。它基于一定的算法和规则，可以识别连续出现的相似动作，并计算其重复次数。

3.2.2 视频流分析算法：

视频流分析算法主要用于对实时视频流进行处理和分析。

首先，打开视频流，并检查是否成功打开。

然后，初始化 MediaPipe 的姿势检测器，以便对每一帧进行姿势检测。

接下来，初始化帧计数器和开始时间，用于计算帧率和分析持续时间。

进入主循环，读取视频流的每一帧。

对于每一帧，进行以下处理：

- 将帧从 BGR 颜色空间转换为 RGB 颜色空间，以满足姿势检测器的要求。
- 使用姿势检测器处理 RGB 帧，获取姿势关键点地标。
- 如果成功检测到姿势关键点地标，则进行以下处理：
    - 将姿势关键点地标的坐标转换为帧图像上的实际坐标，以便后续处理和展示。
    - 使用姿势分类器对姿势进行分类，得到分类结果。
    - 使用姿势平滑器对分类结果进行平滑处理，以减少噪声和抖动。
    - 使用姿势重复计数器对平滑后的分类结果进行计数，跟踪和记录重复动作的次数。
    - 打印关键信息，如当前帧数和重复计数。

- 释放帧图像和处理结果的内存，以避免内存泄漏。

在处理完所有帧之后，检查分析时间是否超过指定的持续时间，如果超过，则退出循环。

最后，释放视频流，结束视频流分析算法的执行。

3.2.3 Flask 接口设计

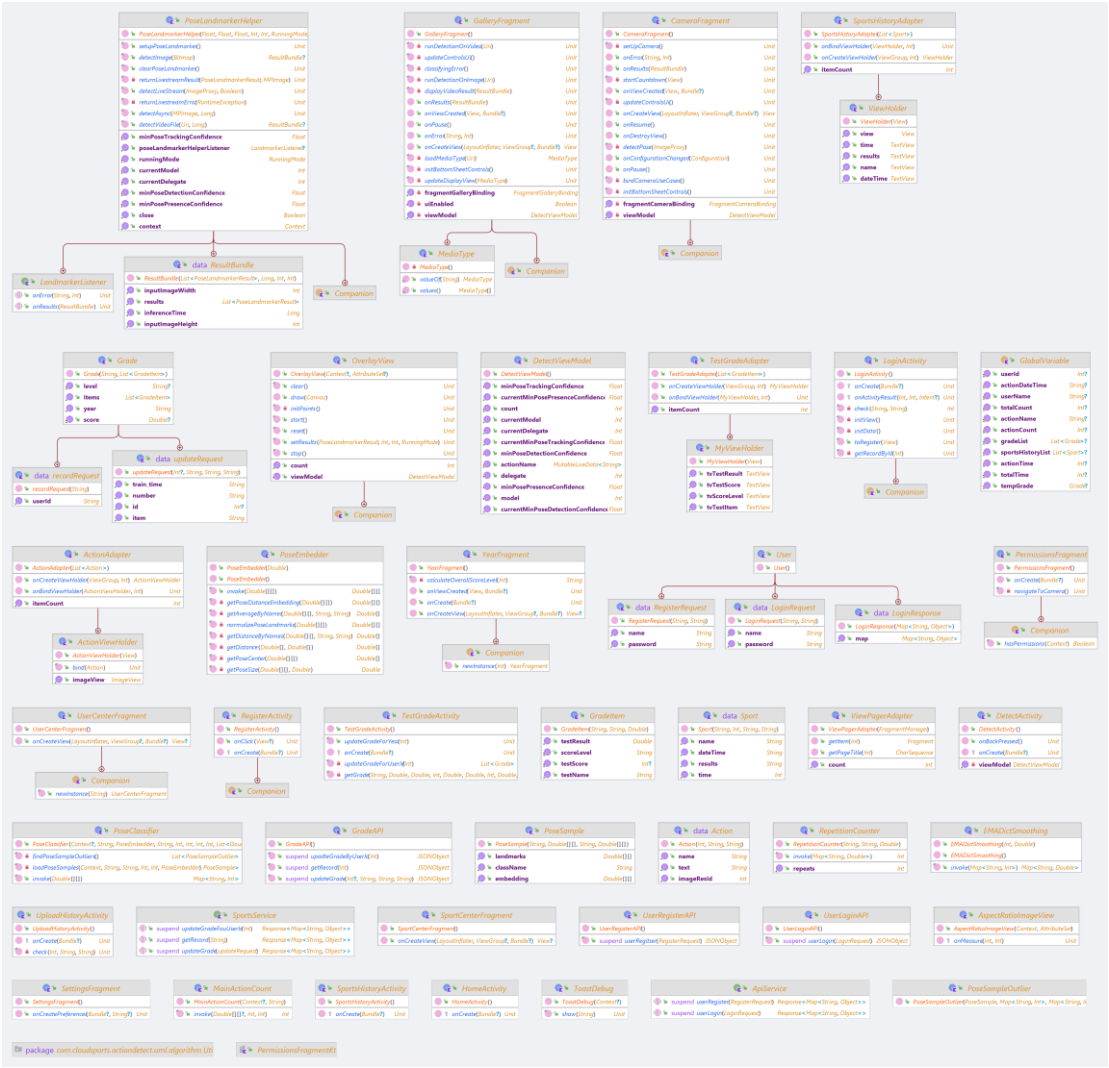
与数据后端的消息传递使用 Flask 框架进行实现。接收到数据后端传入的项目名和持续时间等信息后，后端通过 Flask 接口拉取视频流进行处理，并调用算法后端进行计算。最终，将计算结果返回给数据后端。

在 Flask 接口设计中，定义一个路由`/exercise`，使用 POST 方法接收来自数据后端的请求。在处理函数中，从请求中获取项目名、拉流地址和持续时间等信息。

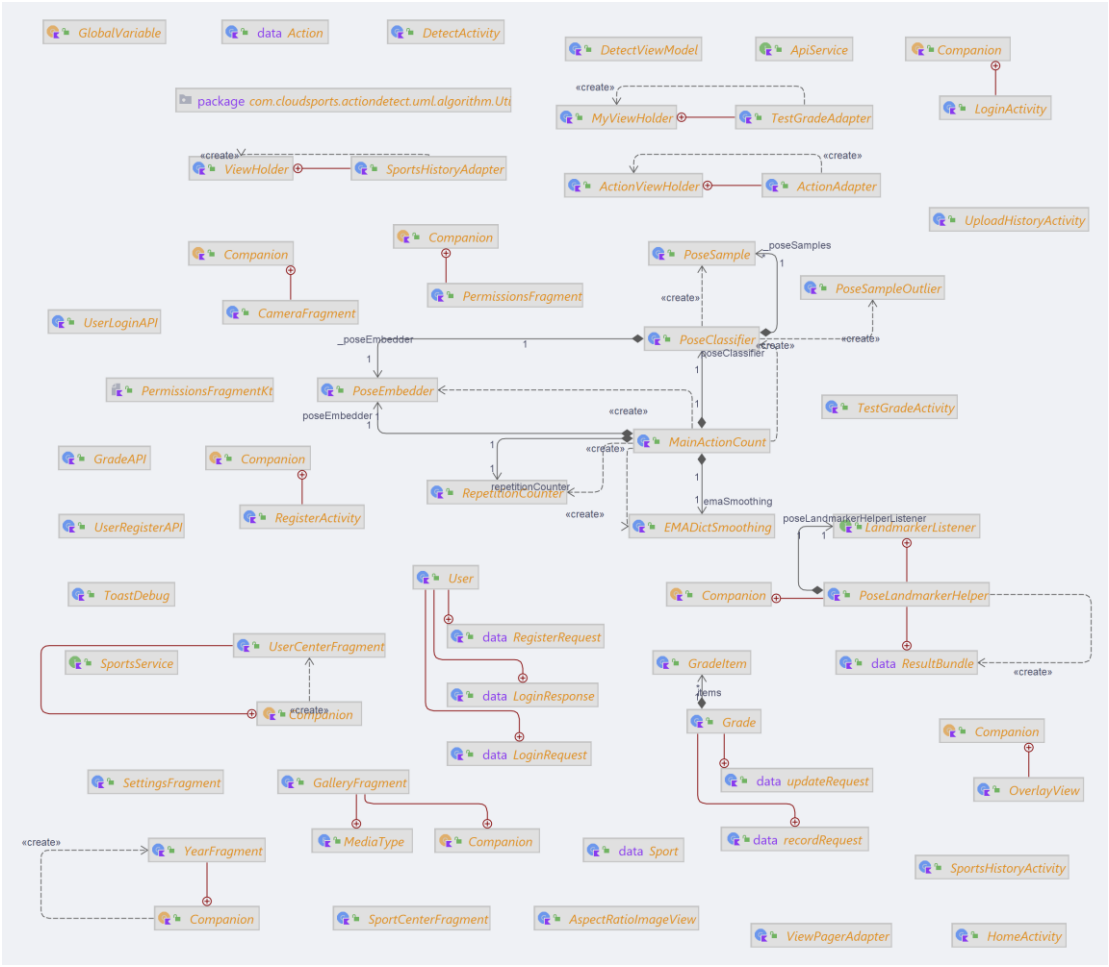
然后，通过调用算法后端的相应函数来处理视频流并进行计算。最后，将计算结果封装为 JSON 格式，并作为响应返回给数据后端。

3.3 安卓端描述

3.2.1 安卓 UML 类图

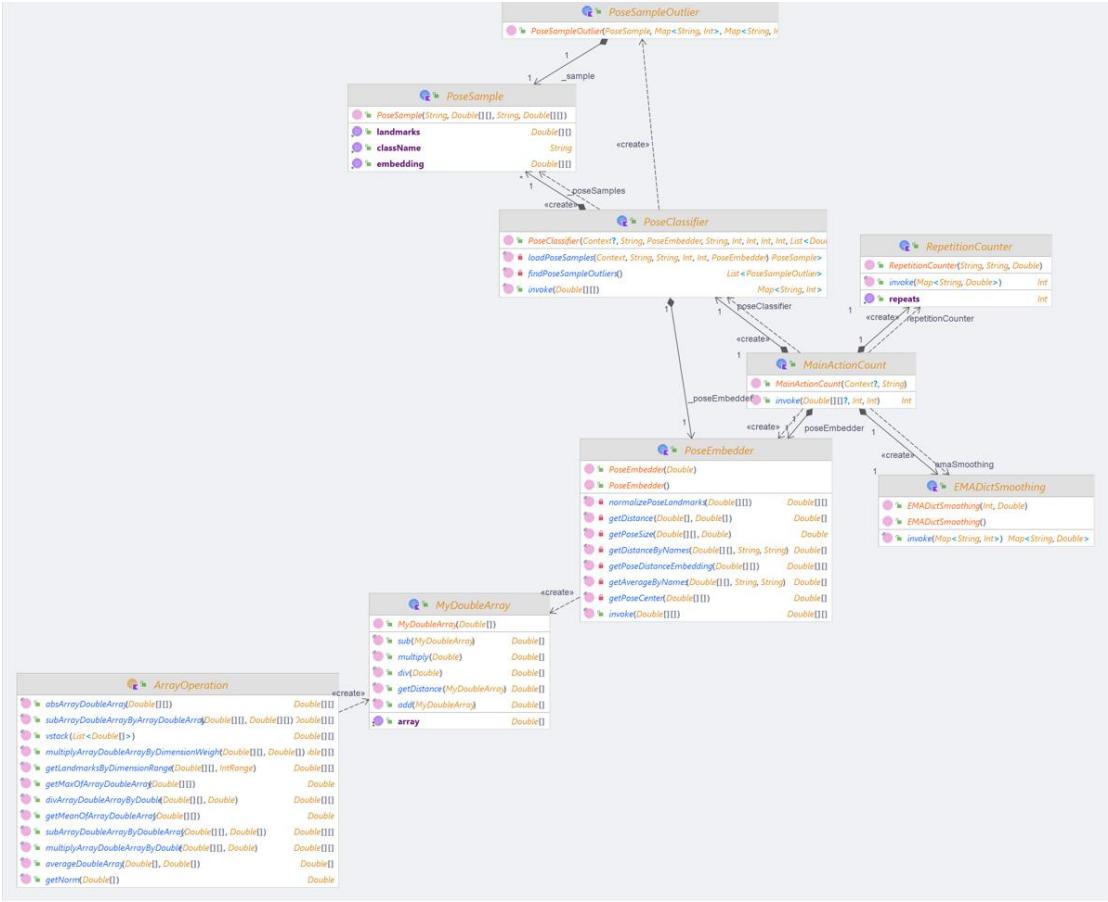


3.2.2 各模块依赖关系



3.2.3 Python 算法迁移至 Kotlin

创建自己的 MyDoubleArray 和 ArrayOperation 类，实现与 NumPy 类似的数组处理操作



3.2.4 主要界面设计

登录注册页

用户中心、体测中心、设置页


15:26 | 0.3K/s
63%
欢迎你：admin
运动时长: 10分钟
运动次数: 11次
体测成绩
运动记录
退出登录
用户中心
运动中心
设置


15:26 | 0.3K/s
63%
选择运动动作
引体向上
俯卧撑
仰卧起坐
深蹲
用户中心
运动中心
设置

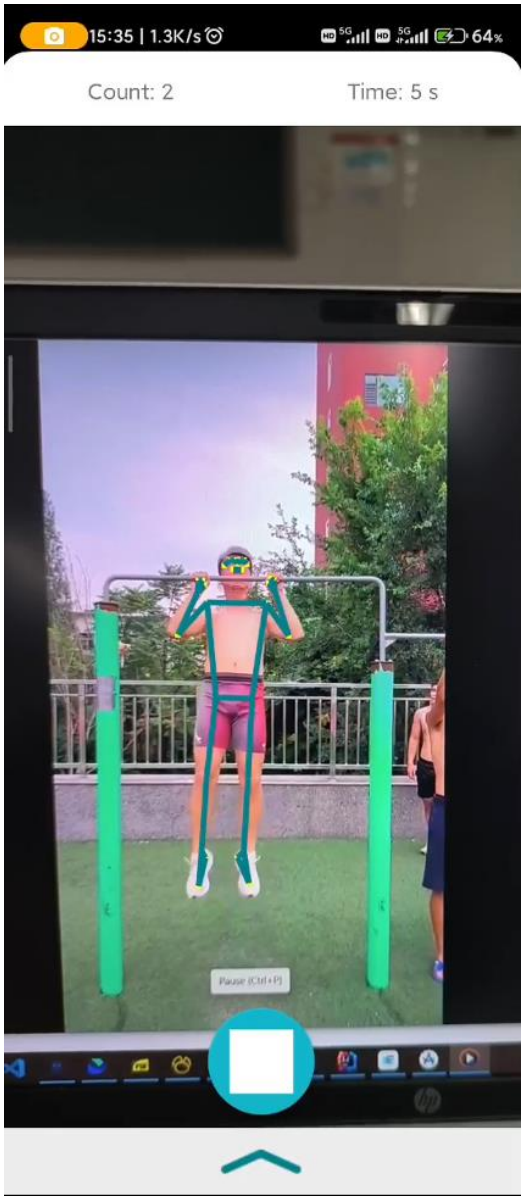

15:27 | 0.7K/s
63%
通用设置
接收通知
开启或关闭通知
语言
选择显示语言
高级设置
用户名
输入你的用户名
用户中心
运动中心
设置

体测成绩、运动记录页


15:34 | 0.9K/s
64%
2023
得分：73.75
及格
身高（厘米） 170.0 90 优秀
体重（千克） 65.0 100 优秀
肺活量（毫升） 3000.0 90 优秀
立定跳远（厘米） 230.0 90 优秀
坐位体前屈（厘米） 20.0 70 及格
引体/仰卧（次） 10.0 0 不及格
50米（秒） 7.0 90 优秀
800/1000米（分'秒） 300.0 60 及格


15:34 | 0.2K/s
64%
运动名：引体向上
运动时长：80秒
运动结果：10次
运动时间：2023-07-13 19:11:50
运动名：引体向上
运动时长：80秒
运动结果：10次
运动时间：2023-07-13 20:10:13
运动名：引体向上
运动时长：80秒
运动结果：10次
运动时间：2023-07-13 21:25:53
运动名：引体向上
运动时长：80秒
运动结果：10次
运动时间：2023-07-13 21:26:13
运动名：引体向上
运动时长：80秒

检测运动、上传成绩页



3.4 推拉流服务描述

3.4.1 推拉流业务流程概述

[该类型的内容暂不支持下载]

3.4.2 高性能异步网络架构 Boost.Asio

在场景中，推拉流服务器需要处理大量的推拉流信息，包括推流客户端、拉流客户端以及其他管理和控制连接。而 Boost.Asio 是一个基于 c++的网络编程库，可使用异步 I/O 操作和事件驱动模型，高效地处理大量的并发连接。能够利用操作系统提供的异步 I/O 机制（如 epoll、kqueue 等），在数据可用时立即进行处理，避免了阻塞和

资源浪费；同时，此框架具有相当良好的跨平台支持，能够保证服务器在不同操作系统上的性能不受较大的影响，可扩展性较强。此框架可以实现高性能、低延迟的推流处理能力，以应对大规模的推流并发连接和高负载的音视频数据处理需求

3.4.3 流媒体协议支持

推拉流均支持以下协议：

- RTMP（Real-Time Messaging Protocol）:实时信息传输协议
  - 音频解码支持：AAC,MP3
  - 视频解码支持：H.264、H.265
- HTTP-FLV（HTTP-based FLV）：HTTP-FLV 是一种基于 HTTP 的流媒体传输协议
  - 音频解码支持：AAC,MP3
  - 视频解码支持：H.264、H.265
- RTSP（Real-Time Streaming Protocol）：RTSP 是一种用于控制流媒体服务器的应用层协议，用于实时流传输和媒体会话控制。
  - 音频解码支持：AAC,MP3
  - 视频解码支持：H.264、H.265

3.4.4 储存与分发

- 储存：与分布式文件系统集成，例如 FastDFS、Ceph 等，将处理后的音视频数据存储在分布式存储节点上。这种存储方案提供高可靠性、可扩展性和容错性，适合存储大规模的音视频数据。尚在概念阶段
- 分发：可以与内容分发网络（CDN）集成，将音视频数据缓存到 CDN 边缘节点，以提供更高效的内容分发。CDN 可以将音视频数据就近分发给用户，减少传输延迟，提高用户体验。

3.4.5 客户端管理

除基本的客户端管理内容（如：建立链接，用户鉴权等），借鉴 ZlmediaKit 中的无人观看的检测机制，与心跳机制相类似，可以做到在无人观看时自动关闭流，节省流量及资源。

3.4.6 日志监控

服务器日志会记录系统运行过程中的各种事件和操作信息，并将其记录到日志文件中。这些日志包括系统启动信息、连接建立和断开信息、音视频数据处理日志、错误和异常信息等。

4 数据库设计

4.1 数据库说明

本项目使用的是 MySQL 数据库，包含 4 个主要的数据表：`user`、`action`、`test` 和 `train`。

4.2 表结构

4.2.1 用户表 `user`

包含四个字段：用户 ID `user_id`（主键），用户名 `name`，用户密码 `password`，用户是否为管理员 `is_admin`。

4.2.2 动作表 `action`

包含三个字段：动作 ID `action_id`（主键），动作名 `name`，动作体测时间限制 `limit_time`。

4.2.3 体侧表 `test`

包含 14 个字段：体测 ID `test_id`（主键，自增），用户 ID `user_id`（外键，引用用户表的用户 ID），身高 `height`，体重 `weight`，肺活量 `vital_capacity`，立定跳远 `standing_long_jump`，坐位体前屈 `sit_and_reach`，引体向上 `pull_up`，50 米跑 `sprint_50m`，800 或 1000 米 `long_distance_run`，俯卧撑 `push_up`，仰卧起坐 `sit_up`，深蹲 `squat`，体测年份 `test_time`。

4.2.4 训练运动记录表 `train`

包含 7 个字段：用户 ID `user_id`（外键，引用用户表的用户 ID），动作 ID `action_id`（外键，引用动作表的动作 ID），运动时间 `time`，运动持续时间 `practice_time`，运动计数 `count`，运动数值 `value`，训练 ID `practice_id`（主键，自增）。

5 非功能性设计

5.1 性能需求

响应时间：应用的所有主要功能在高峰期（例如体测期间）的平均响应时间应在 3 秒以内。

负载能力：系统应能在高并发环境下运行，如同时支持至少 2000 个在线用户。

资源使用：系统应优化内存和 CPU 使用，以最小化运行成本并提高效率。

5.2 安全性需求

数据加密：所有敏感信息（例如用户密码）都应进行适当的加密处理。

访问控制：只有经过身份验证和授权的用户才能访问特定功能，管理员账户具有高级权限，可以管理所有的用户信息。

防护措施：应实现预防 SQL 注入、跨站脚本（XSS）等网络攻击的安全机制。

5.3 可靠性需求

系统稳定性：系统需要有足够的稳定性，即使在高并发环境下也能保证运行不中断。

错误恢复：在出现故障时，系统需要能快速恢复，确保服务的连续可用性。

错误处理：需要有恰当的错误处理机制，例如，当部分功能出现问题时，应避免影响其他功能。

5.4 易用性需求

用户界面：应用的用户界面应简洁明了，便于用户理解和操作。

帮助与文档：系统需要提供充分的提示信息和帮助文档以支持用户使用，具备实时的错误提示和解决方案的建议。

用户指南：对于功能较复杂的部分，系统需要提供清晰的用户指南和教程。

5.5 可维护性与可拓展性需求

模块化：系统应设计成模块化，每个模块对应一项或一类功能，便于维护和升级。

扩展性：系统的设计应允许添加新功能或修改现有功能，以适应未来的需求变化。

代码质量：代码应易于理解和修改，有适当的注释，符合一定的编程规范，以便维护人员进行后续工作。