目录

1. 项目概述

1.1 项目背景与简介

《移动终端软件开发》课程大作业要求完成一个具备多界面、注册/登录、本地数据存储、相关服务（如背景音乐/白噪音等）以及应用间交互能力的移动应用，并强调使用线性布局与相对/叠层布局完成较完整的 UI 设计与功能闭环。基于该要求，本项目实现了 MOOC FLOW（智学流）：一款面向学习者的效率型教育应用，围绕“课程发现—学习计划—专注学习—数据回顾”形成完整学习流程。

应用课程内容来自开源慕课数据集 MOOCCube 的裁剪子集（以本地 JSON 形式存放）[7]，支持课程列表展示、分类筛选与搜索；用户可注册登录后收藏课程形成学习计划；在专注学习场景中提供番茄钟式倒计时[8]，并可选择白噪音/音乐伴学（即使音频加载失败仍可计时完成专注）；专注结束后将学习记录写入本地 SQLite 数据库，并在首页与“我的”页面生成统计数据（总专注时长、学习次数、收藏课程数量）；同时在课程详情页支持调用系统能力进行课程分享，实现应用间消息交互能力。

1.2 技术架构与开发环境

（1）技术架构

项目采用“页面展示 + 业务状态 + 数据持久化 + 系统能力”的结构组织。页面层使用 ArkUI 声明式 UI 构建登录页、首页、课程详情页、专注页与个人页，并通过路由完成参数传递与页面跳转。专注功能由会话管理模块维护计时状态，确保用户切换页面后计时仍可持续；统计数据通过全局状态与数据库查询联动，保证首页与个人页数据同步刷新。数据层使用本地 SQLite（HarmonyOS relationalStore/RdbStore）创建用户、收藏、学习记录三张表，实现注册登录校验、学习计划保存、专注记录落库与查询。音频服务基于 AVPlayer 播放 rawfile 内的白噪音/音乐资源，并提供音频发现与加载失败提示机制；课程分享通过 Want/Ability 调起系统消息能力，完成应用间交互。

（2）开发环境与关键技术

开发工具为 DevEco Studio，项目使用 ArkTS 语言与 ArkUI 框架开发，构建与打包由 hvigor 完成[1]。本地数据库使用 @ohos.data.relationalStore（RDB/SQLite），音频播放使用 @ohos.multimedia.media（AVPlayer）[5]，资源文件通过 resourceManager 读取（课程 JSON、音频清单与音频文件），页面导航使用 @ohos.router，交互提示使用 @ohos.promptAction。应用可通过

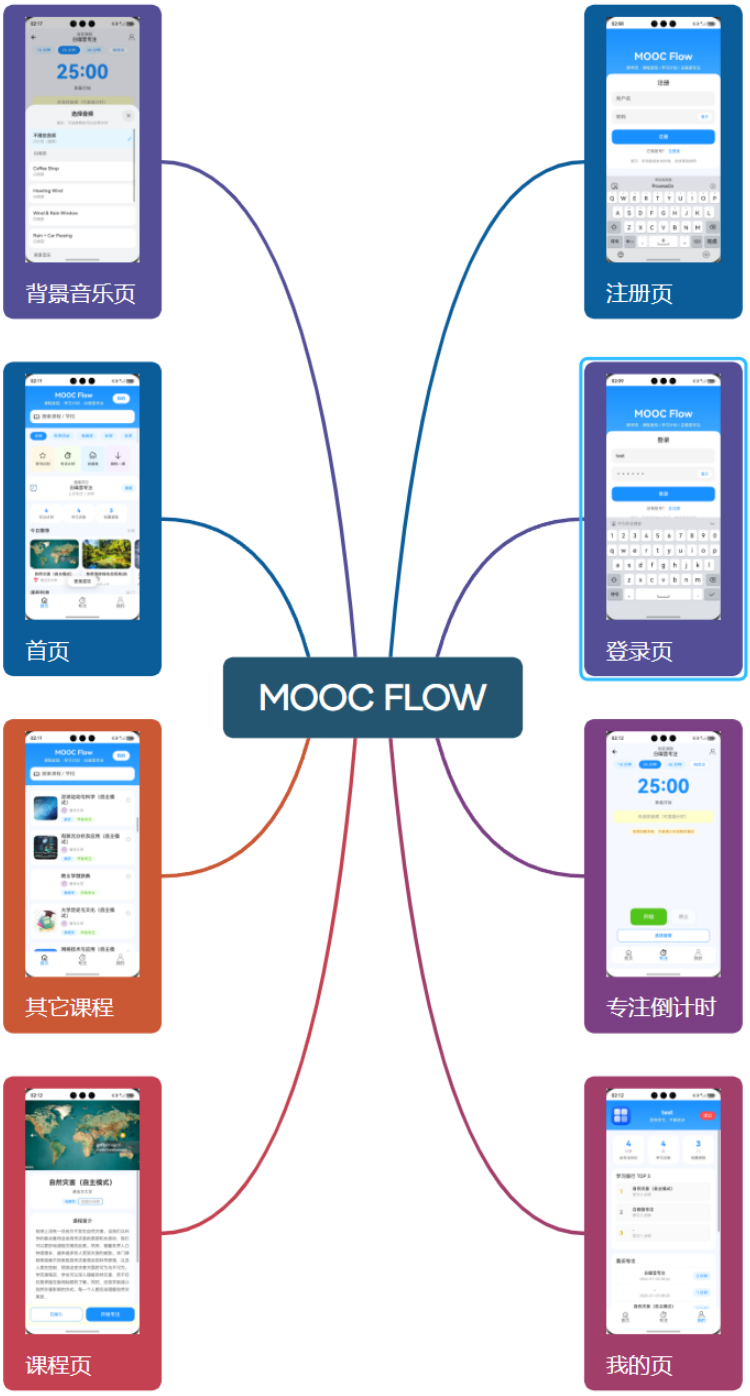
Preview 进行界面预览，也可在模拟器或真机运行并验证数据库与音频能力。

2. 系统设计

2.1 页面流程与交互设计

本项目以“认证进入—课程发现—课程详情—专注计时—数据回顾”的学习闭环为主线组织页面与交互。用户启动应用后首先进入注册/登录页完成身份认证，认证成功后进入首页；首页支持课程搜索与分类筛选，并可下滑查看更多课程列表。用户在首页选择课程进入课程详情页，详情页提供课程信息展示，并支持加入学习计划/收藏、开始专注以及课程分享等操作。点击开始专注后进入专注倒计时页，用户可选择固定时长或自定义时长，并进行开始、暂停/继续、停止保存等控制；当倒计时结束或用户主动停止时，系统会将本次专注写入本地数据库形成学习记录，并同步刷新首页与个人中心的统计数据。

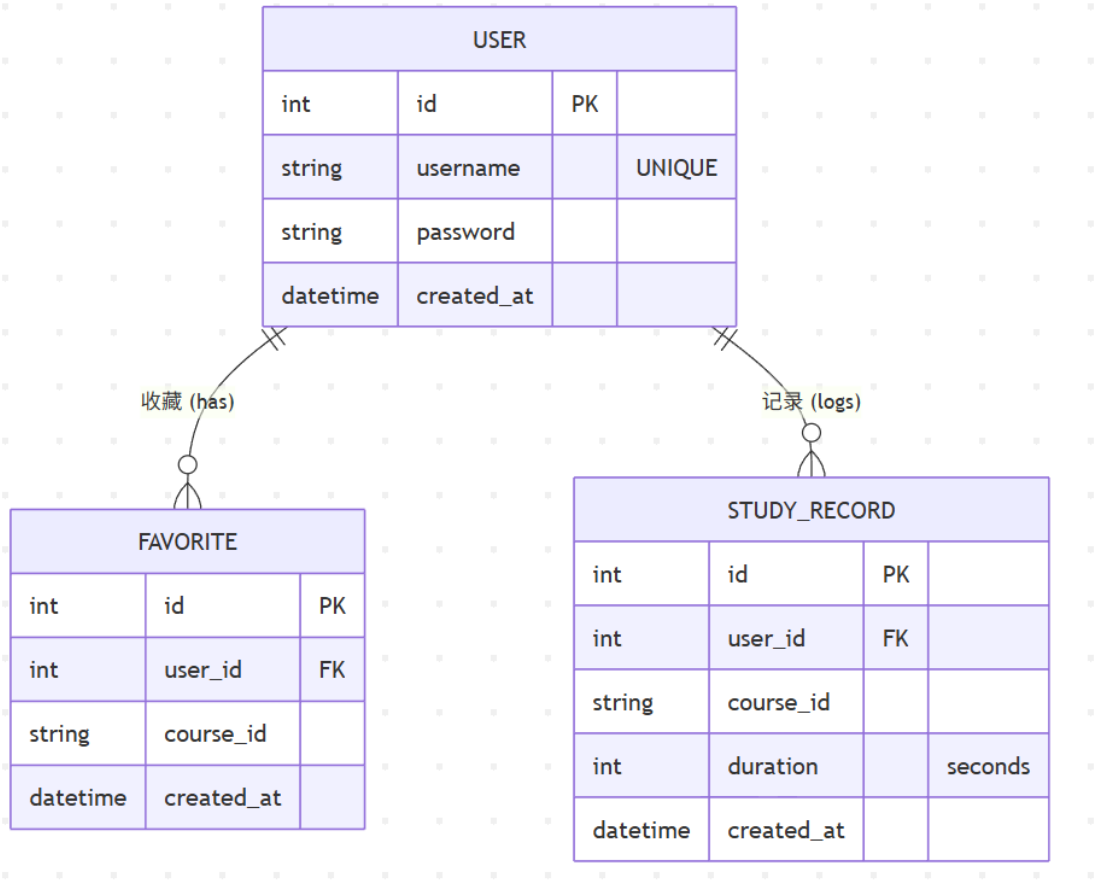
同时，项目还提供独立的“背景音乐/白噪音”入口，该入口进入专注计时功能时不绑定具体课程，以“白噪音专注”形式记录专注时长，使“课程专注”和“纯白噪音专注”都能统一沉淀为学习记录并参与统计。上述页面之间的跳转关系与交互闭环已在汇总页面关系图中展示，如下图所示。



2.2 数据库设计

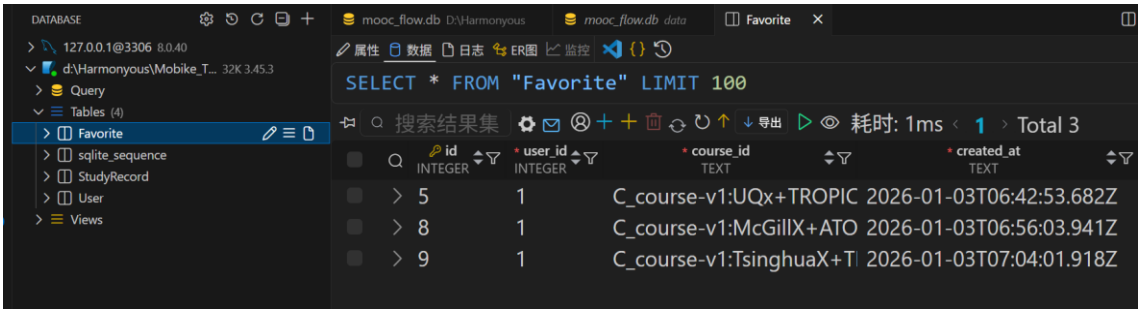
本项目使用 HarmonyOS relationalStore（RDB/SQLite）实现本地数据持久化，数据库文件名为 mooc_flow.db[4]。数据库的设计目标是满足课程要求中的“注册/登录”“本地数据存储”“学习计划/收藏”“学习记录与统计”等核心功能[6]，因此采用三张核心表形成数据闭环：User 用于账号注册与登录校验；

Favorite 用于保存用户收藏课程（学习计划），并通过联合唯一约束避免重复收藏；StudyRecord 用于记录每次专注的时长与时间戳，既支持课程专注记录，也支持白噪音专注记录，从而保证所有专注行为都能纳入统计分析。整体实体关系与表结构设计如下图所示。

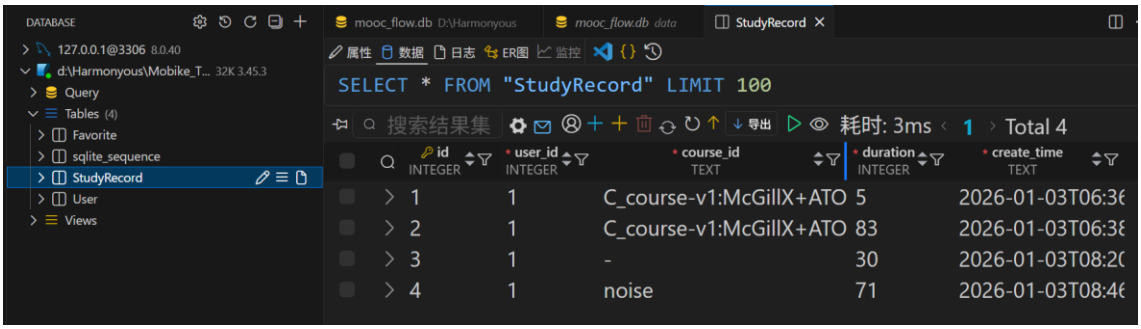


User 表如下图所示：

Favorite 表如下图所示：



StudyRecord 表如下图所示：



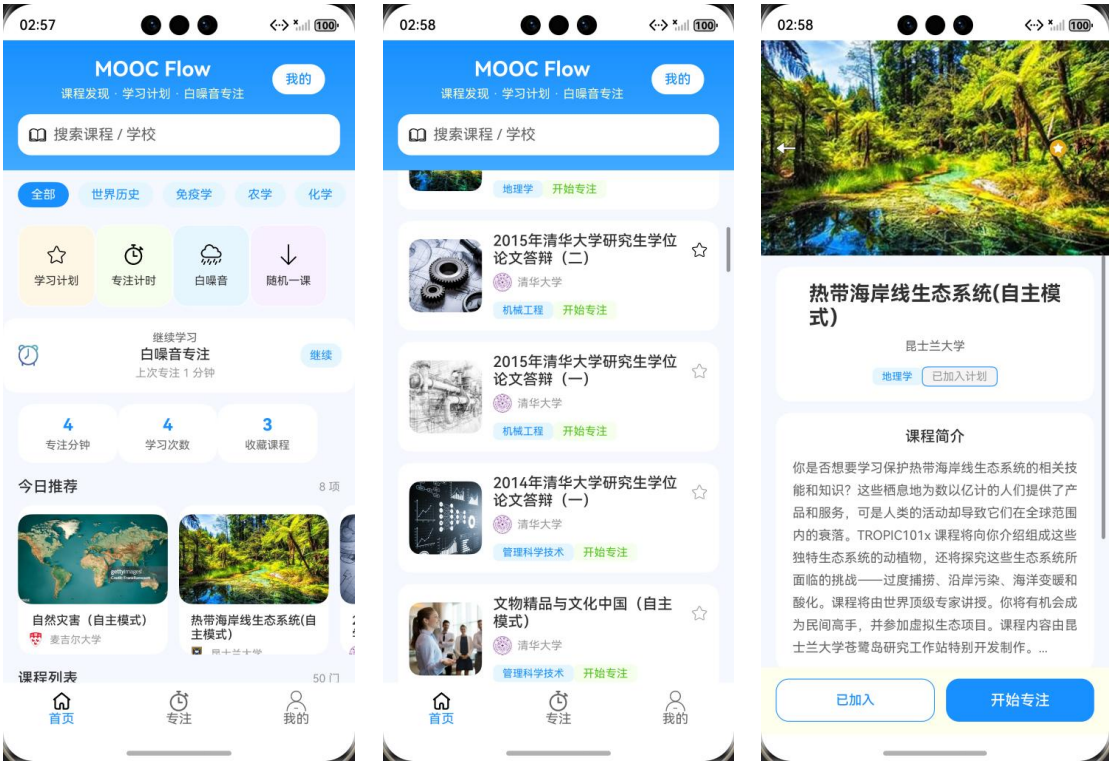
3. 功能模块与界面实现

3.1 认证模块

认证模块包含注册与登录两部分，负责用户身份建立与登录态维护。用户在注册/登录页输入用户名与密码后，系统会初始化本地数据库并进行校验：注册时写入 User 表（用户名唯一约束），登录时根据用户名与密码查询用户 id，成功后写入全局登录态并跳转首页，从而保证后续收藏、学习记录等数据均能与当前用户绑定。认证完成后会同步刷新一次统计缓存，确保首页/个人中心的统计项显示正确。认证页面效果如下图所示。

3.2 课程发现与展示模块

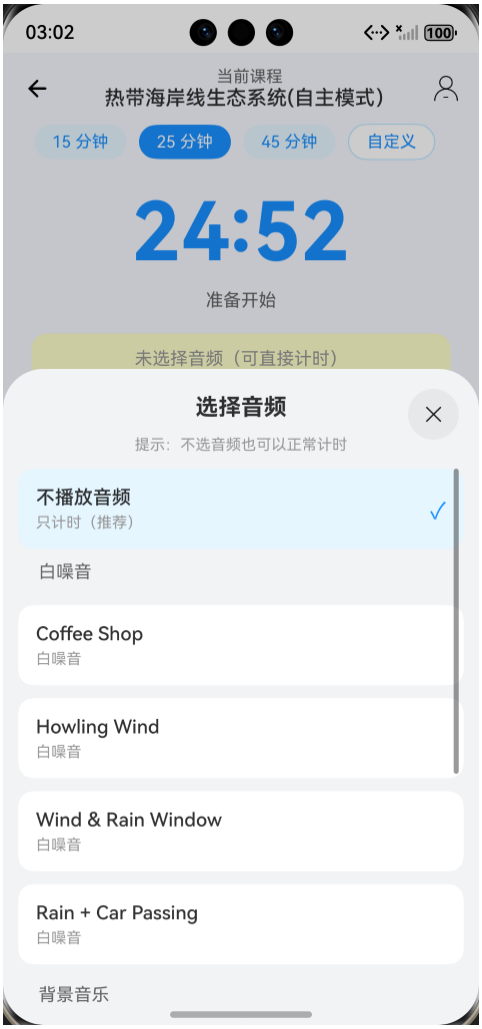
课程发现模块以首页为核心入口，课程数据来自本地裁剪后的 MOOCCube 子集文件 mooc_data.json。页面加载时读取 JSON 并缓存为课程列表，首页提供分类标签与搜索框，支持按“课程名/学校”等字段进行检索，同时提供推荐课程卡片与完整课程列表（支持下滑查看更多课程）。用户点击课程卡片进入课程详情页，详情页展示课程封面、课程信息（名称、学校、简介等），并提供“加入学习计划/收藏”“开始专注”“分享课程”等关键操作。课程详情页的分享功能通过系统能力调起消息应用并预填推荐文本，满足应用间交互要求。首页、下滑后的课程列表和课程详情页如下图所示。



3.3 专注与音频服务模块

专注模块以番茄钟式倒计时为核心，支持固定时长快捷选择与自定义时长选择，并提供开始、暂停/继续、停止保存等控制。专注可分为两种入口：从课程详情进入时会携带 courseId 并记录为“课程专注”；从首页“白噪音/背景音乐”入口进入时不绑定课程，以 courseId='noise' 记录为“白噪音专注”。专注过程结束（倒计时归零或用户主动停止）后，会将本次专注时长写入 StudyRecord 表，从而驱动后续统计与排行展示。

音频服务支持在专注时选择白噪音/音乐进行伴学。音频资源存放在 resources/rawfile/，系统会尝试扫描 rawfile 列表并读取 audio_manifest.json 作为兜底清单，再由 AudioManager 使用 AVPlayer 加载与播放；若音频加载失败会给出提示，但不影响倒计时功能（仍可正常专注计时）。专注页与音频选择效果如下图所示。



为提升应用在“专注学习”场景下的可用性与应用设计完整度，专注页除提供 15/25/45 分钟等快捷时长外，还额外实现了“自定义倒计时”功能。用户可在专注页点击“自定义”入口，通过时间选择器自由设定分钟与秒数（例如 00:30、10:00、35:00 等），确认后系统会将该时间转换为专注会话的倒计时总时长，并与开始/暂停/继续/停止保存等控制逻辑保持一致。该设计能够覆盖不同课程与不同学习习惯下的专注时长需求，使专注模块从固定模板式计时升级为可配置的通用专注工具，更符合移动应用的人机交互设计原则。自定义倒计时的交互与界面效果如下图所示。

3.4 个人中心与数据统计

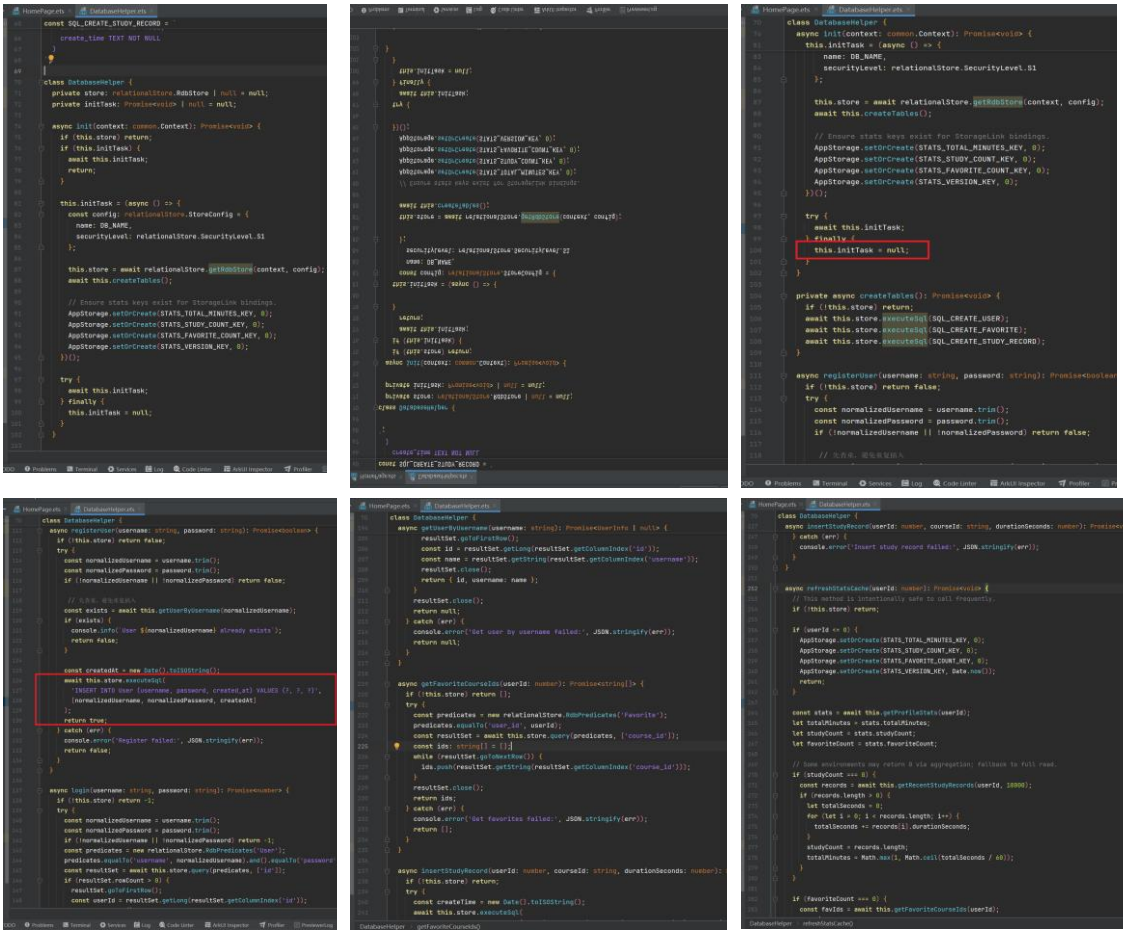
个人中心用于汇总用户学习数据与学习计划。页面展示三项核心指标：总专注时长、学习次数、收藏课程数量；同时提供学习排行 TOP3（按课程累计专注时长排序）、最近专注记录列表（展示课程/白噪音名称、时间与本次分钟数）、学习计划列表（收藏课程，支持跳转详情或直接进入专注）。统计数据由本地数据库查询得到，并通过全局状态绑定实现首页与个人中心同步刷新，因此当用户完成一次专注保存或收藏/取消收藏后，相关统计会即时更新。个人中心界面如下图所示。

4. 核心代码与模块分析

4.1 数据持久化层封装

本项目将本地数据存储统一封装在 DatabaseHelper 中，避免页面层直接编写 SQL，形成“页面调用接口—数据库层统一执行—返回结果”的结构，便于维护与扩展。数据库初始化在应用首次进入时完成，通过 init(context) 创建 RdbStore 并执行建表语句，确保 User / Favorite / StudyRecord 三张表存在。注册与登录分别对应 registerUser() 与 login()：注册前会先查重（用户名唯一），登录则通过用户名与密码查询用户 id。收藏相关逻辑集中在 toggleFavorite() 与 getFavoriteCourseIds()，保证收藏与取消收藏可以复用同一套接口并避免重复数据。学习记录写入由 insertStudyRecord() 完成，记录 user_id、course_id、duration、create_time，并支撑首页与个人中心的“总专注时长、学习次数、最近专注”等展示。

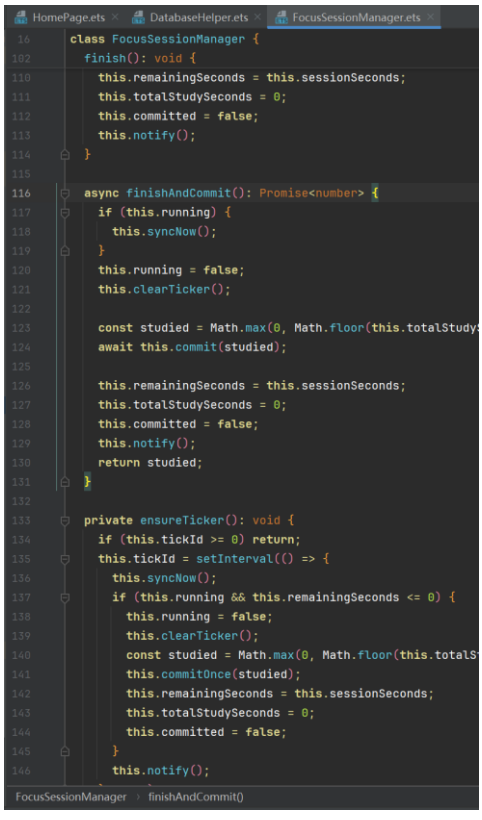
为了实现首页与“我的”页统计数据的实时同步，数据库层还提供 refreshStatsCache(userId)：在写入学习记录或切换收藏后，主动重算统计并写入全局 AppStorage，页面使用 @StorageLink 绑定后即可自动刷新。该设计将“统计更新”从页面逻辑中抽离出来，保证数据一致性与可复用性。数据持久化层的核心代码如下图所示。



4.2 专注会话管理器

专注模块的核心难点在于“倒计时状态管理与页面切换保持一致”。为此项目设计了 FocusSessionManager 作为专注会话管理器，采用单例方式维护当前会话的课程标识、会话总时长、剩余时间、已专注时长以及运行状态等数据，并通过定时 tick 机制每秒更新状态。页面层（专注页）只负责展示与触发控制，不直接维护复杂计时逻辑，从而降低 UI 与业务耦合。

在交互上，start() 负责启动计时并记录开始时间；pause() 会同步当前进度并停止 tick；当用户停止保存或倒计时结束时，通过 finishAndCommit()完成“停止计时 + 写入数据库学习记录 + 重置会话”的完整闭环。该写入动作会调用数据库层接口并触发统计缓存刷新，使首页与个人中心的数据能够即时更新。专注会话管理器的关键实现如下图所示。

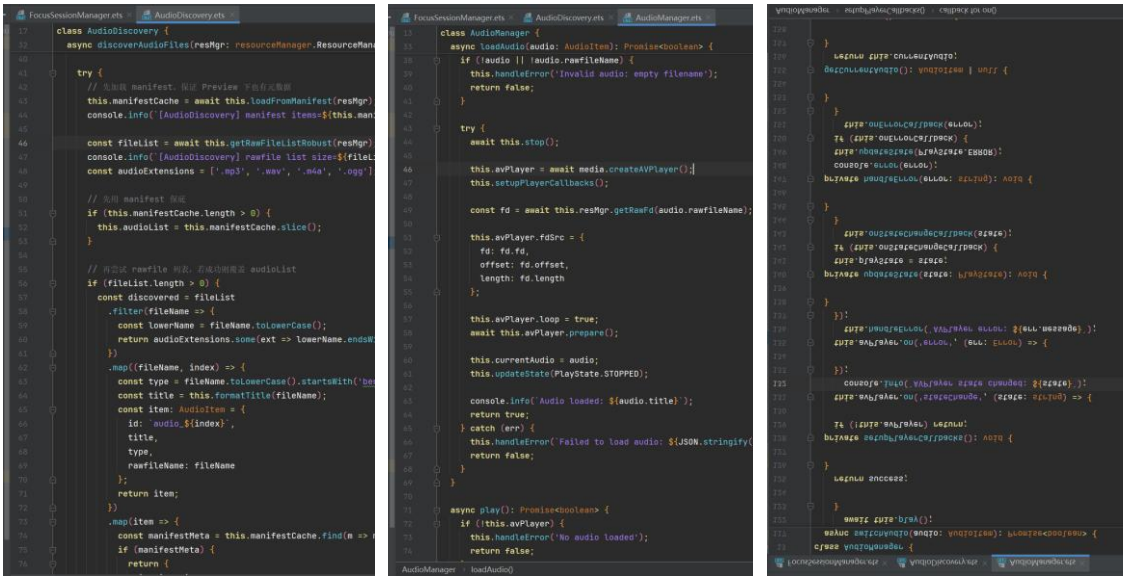


4.3 音频服务层

音频服务层采用“资源发现 + 播放控制”两段式封装，分别由

AudioDiscovery 与 AudioManager 负责。AudioDiscovery 用于从 rawfile 资源中发现可用音频文件，并可结合 audio_manifest.json 进行兜底加载，保证音频列表在不同运行环境中尽可能稳定展示；AudioManager 负责 AVPlayer 的创建、加载、播放、暂停与停止，并统一处理状态回调与错误提示。

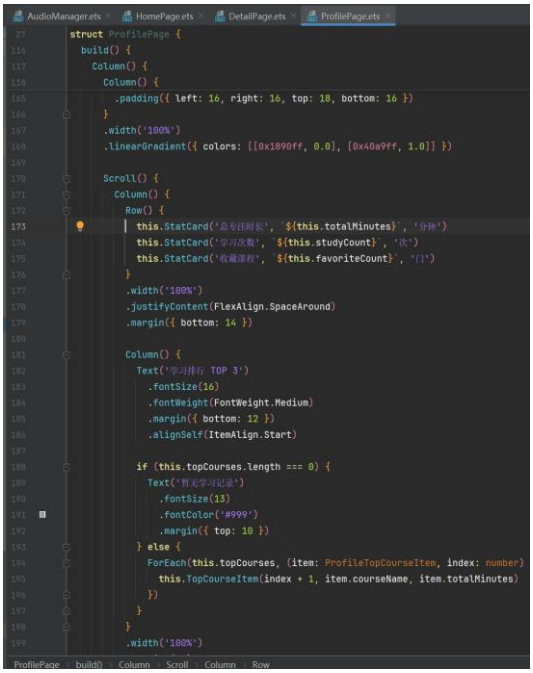
在专注场景中，音频属于增强体验的服务能力，因此项目实现了“音频可选、失败可提示、但不影响计时”的策略：即便音频加载失败或设备环境限制导致无法播放，专注倒计时仍可正常进行并写入学习记录，确保核心功能可用且符合课程对“相关服务”的要求。音频服务层的关键实现如下图所示。



4.4 UI 布局关键技术

项目界面采用 ArkUI 声明式开发，整体遵循“卡片化 + 分区布局”的思路[2]。页面主结构广泛使用 Column/Row 完成线性布局，以满足不同屏幕尺寸下的自适应排列；在需要叠层或相对位置表达时使用 Stack（例如课程详情页头图与顶部操作按钮叠加、专注页顶部区域与内容分层），从而覆盖课程要求中的线性布局与相对/叠层布局。对于内容较长的页面（如首页课程列表、个人中心信息汇总），使用 Scroll 提供纵向滚动，避免内容溢出并提升可用性。

专注页在交互组件上采用统一弹窗策略（音频选择与自定义时间选择均通过底部弹窗呈现），并将弹窗内容以条件渲染方式复用，避免多弹窗绑定冲突，提高交互稳定性。首页与个人中心的统计展示采用 @StorageLink 绑定全局状态，使统计卡片能随数据库写入自动刷新，从 UI 层体现“数据驱动界面”的设计思想。相关 UI 布局与关键写法如下图所示。



5. 项目总结与反思

5.1 遇到的问题与解决方案

本项目在开发过程中遇到的第一个典型问题是资源与构建规范带来的编译失败。例如部分图片或资源文件名包含中文字符、非法字符时，资源打包会直接报错，导致无法 Rebuild。针对该问题，统一对资源进行英文命名与规范化管理，并在资源目录结构上进行归类整理，使构建过程稳定可重复。与此同时，由于

Preview 与真机/模拟器在资源读取能力上存在差异，部分 rawfile 的枚举与读取在 Preview 下可能出现列表为空或读取失败的情况。为降低环境差异带来的影响，项目在音频与课程数据加载上增加了兜底机制（如 manifest 兜底与错误提示），保证核心功能在不同环境下仍可展示与运行。

第二个关键问题集中在 ArkTS 语法限制与类型规范上。开发初期从 TypeScript 迁移思路较强，出现了诸如对象字面量当作类型、索引签名、解构参数、any/unknown 等不被 ArkTS 编译规则允许的写法[3]，从而引发大量编译错误。解决方案是按 ArkTS 规范补齐显式类型声明，使用 interface/class 进行结构定义，避免不支持语法，并将公共逻辑抽离为工具类/管理器，使页面代码保持清晰可维护。

第三个核心问题是专注计时的状态一致性与数据统计更新。若倒计时逻辑仅存在于页面内部，在页面切换、返回或重新进入时容易被打断，造成用户体验不一致，也会导致学习记录写入时机混乱。项目最终通过专注会话管理器统一维护倒计时状态，确保专注过程能够跨页面保持一致，并在停止保存或倒计时结束时可靠写入学习记录。统计数据方面，初期存在“学习记录列表已出现但顶部统计仍为 0”的现象，根因通常是页面刷新时机与数据读取方式不一致。为保证首页与个人中心统计同步，项目采用“数据库写入后刷新统计缓存 + 页面通过全局状态绑定自动更新”的方式，形成稳定的数据驱动链路，从而解决统计不更新的问题。

最后，音频播放在模拟器环境可能出现加载失败或播放器能力受限的问题。针对该情况，项目将音频定义为增强体验的服务能力而非核心必需能力：当音频加载失败时提供明确提示，但不阻塞倒计时与学习记录写入，保证作业的核心功能闭环可完成，同时满足“相关服务”的实现要求。

5.2 改进方向

后续可从数据、体验与工程化三个方面继续完善。数据层面，可进一步扩大 MOOCCube 裁剪范围并增加更丰富的字段（如课程标签、难度、学习人数等），在首页加入更智能的推荐策略与“学习报告”生成逻辑，使应用更具“数据驱动交互”的特色。体验层面，可将专注功能做成更完整的番茄钟体系，例如增加多阶段循环（专注/休息）、专注提醒与本地通知、专注历史可视化统计（周/月趋势图），并对音频播放加入播放列表、音量渐变、混音等更细腻的交互设计。工程化层面，可引入更明确的模块边界与测试思路，对数据库、会话管理器、音频服务增加更系统的日志与异常处理策略，减少对运行环境差异的依赖，并进一步优化资源管理与构建配置，使项目在不同设备与版本上具备更强的可移植性与稳定性。

6. 参考文献

[1] 华为技术有限公司. HarmonyOS 应用开发文档[EB/OL]. (2024-06-01)[2024-06-01]. https://developer.huawei.com/consumer/cn/doc/harmonyos-guides.

[2] 华为技术有限公司. ArkUI 声明式 UI 开发指南[EB/OL]. (2024-06-01)[2024-06-01]. https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/arkui-overview.

[3] 华为技术有限公司. ArkTS 语言规范与开发说明[EB/OL]. (2024-06-01)[2024-06-01]. https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/arkts-overview.

[4] 华为技术有限公司. relationalStore (RDB/SQLite) 数据库开发指南[EB/OL]. (2024-06-01)[2024-06-01]. https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/database-relational-overview.

[5] 华为技术有限公司. AVPlayer 音频播放服务开发指南[EB/OL]. (2024-06-01)[2024-06-01]. https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/media-avplayer-overview.

[6] 王珊, 萨师煊. 数据库系统概论[M]. 5 版. 北京: 高等教育出版社, 2014.

[7] TANG J, ZHANG J, YAO L, et al. MOOCCube: A Large-scale Data Repository for NLP Applications in MOOCs[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 316-323.

[8] CIRILLO F. The Pomodoro Technique[M]. Berlin: FC Garage Press, 2018.