

自然写互动课堂手机端应用软件 V1.0

软件鉴别材料 — 用户操作手册与设计说明书

软件全称：自然写互动课堂手机端应用软件

软件版本：V1.0

权利人：深圳自然写科技有限公司

文档类型：移动应用用户操作手册 + 设计说明书

文档编号：WRITECH-APP-MOBILE-DS-001

编制日期：2026年2月

适用平台：Android 8.0+ / iOS 14.0+

目录

- 第一章 软件整体概述
 - 1.1 软件简介与功能综述
 - 1.2 软件用途与适用场景
 - 1.3 运行环境与系统要求
 - 1.4 开发语言与技术规范
 - 1.5 版本说明
- 第二章 系统架构与设计思路
 - 2.1 总体架构设计
 - 2.2 MVVM架构说明
 - 2.3 各层次详细说明
 - 2.4 数据设计
 - 2.5 接口设计
 - 2.6 安全设计
 - 2.7 权限说明
- 第三章 核心模块功能详细说明
 - 3.1 登录与身份认证模块
 - 3.2 教师端 — 课堂互动控制模块
 - 3.3 教师端 — 作业布置与批改模块
 - 3.4 教师端 — 实时收笔与展示模块
 - 3.5 家长端 — 学情报告查看模块
 - 3.6 家长端 — 书写回放模块
 - 3.7 消息通知模块

- 3.8 蓝牙连接点阵笔模块
- 3.9 学习数据统计图表模块
- 3.10 拍照搜题模块
- 3.11 离线缓存与数据同步模块
- 3.12 个人中心与设置模块
- 第四章 操作流程与使用步骤
- 4.1 安装与首次启动
- 4.2 账号登录与角色选择
- 4.3 教师端完整使用流程
- 4.4 家长端完整使用流程
- 4.5 消息与通知使用流程
- 4.6 异常处理与故障排查
- 第五章 与源代码的对应关系
- 5.1 模块与源代码文件对应表
- 5.2 核心类与方法说明
- 5.3 状态管理架构说明
- 附录A 界面原型说明
- 附录B 第三方SDK集成说明
- 附录C 术语表
- 附录D 版本历史

第一章 软件整体概述

1.1 软件简介与功能综述

自然写互动课堂手机端应用软件（以下简称"手机APP"）是自然写互动课堂系统面向教师和家长的手机端应用程序，同时支持Android和iOS双平台。手机APP采用Flutter跨平台框架开发，基于MVVM架构，提供课堂管理、作业布置、学情查看、家校沟通等核心功能，是教师日常移动教学和家长了解孩子学习情况的重要工具。

手机APP设计遵循"简洁高效、一端多角色"原则，同一安装包支持教师和家长两种角色，登录后自动呈现对应的功能界面。教师侧重课堂互动控制和批改管理，家长侧重孩子学情报告和作业完成情况跟踪。

主要功能模块综述：

角色	功能模块	说明
教师端	课堂互动控制	开课、发题、收卷、随机点名、暂停课堂
教师端	实时收笔展示	实时接收学生笔迹，选取作品展示
教师端	作业布置与批改	发布作业/试卷，查看AI批改结果，人工复核

角色	功能模块	说明
教师端	班级学情数据	班级整体得分分布、知识点掌握情况
教师端	蓝牙移动教学	蓝牙连接点阵笔，移动板书模式
家长端	学情报告	查看孩子知识掌握度、书写能力成长轨迹
家长端	作业完成通知	接收孩子作业完成提醒，查看完成质量
家长端	书写回放	回放孩子书写过程，查看笔顺是否规范
家长端	学习打卡	记录孩子每日练字打卡情况
通用	消息通知	家校沟通、系统通知、互动消息
通用	拍照搜题	拍照识别题目，与孩子作答对比
通用	个人中心	账号管理、设备管理、通知设置

1.2 软件用途与适用场景

教师使用场景：

- **移动巡课：**教师在教室内走动巡视时，通过手机实时查看每位学生的书写状态和完成进度，无需回到讲台PC操作
- **课后批改：**课后在手机上快速浏览AI批改结果，对需要人工复核的题目进行点评标注
- **家校沟通：**通过消息功能向特定学生家长发送学习提醒或布置个性化练习任务
- **移动板书：**教师手持点阵笔直接在教室任意位置书写，笔迹实时投影至智慧黑板

家长使用场景：

- **学情追踪：**每日/每周查看孩子作业完成情况和AI评分，了解薄弱知识点
- **书写监督：**通过书写回放功能查看孩子练字的笔顺是否正确，提供有针对性的辅导
- **作业提醒：**收到孩子未完成作业的系统提醒，督促孩子及时完成
- **成长记录：**查看孩子书写能力的月度/学期成长对比图表

1.3 运行环境与系统要求

Android平台：

配置项	最低要求	推荐配置
Android版本	Android 8.0 (API Level 26)	Android 12.0+
内存	2GB RAM	4GB RAM
存储	200MB可用空间	1GB可用空间（含缓存）
网络	WiFi 或 4G/5G	5G / WiFi 6

配置项	最低要求	推荐配置
蓝牙	BLE 4.0（教师端）	BLE 5.0
摄像头	800万像素（拍照搜题）	1200万像素以上

iOS平台：

配置项	最低要求	推荐配置
iOS版本	iOS 14.0	iOS 16.0+
设备型号	iPhone 8 及以上	iPhone 13 系列及以上
存储	200MB可用空间	1GB可用空间
网络	WiFi 或 4G/5G	5G / WiFi 6
蓝牙	Core Bluetooth 支持BLE	BLE 5.0

网络环境： – 正常使用需要网络连接（云端API调用） – 已缓存的作业列表和学情报告支持离线查看 – 弱网络（2G环境）下基础功能可用，书写回放等大数据功能受限

1.4 开发语言与技术规范

主要技术栈：

技术	版本	用途
Flutter	3.16.0	跨平台UI框架（Android + iOS）
Dart	3.2.0	主要开发语言
flutter_bloc	8.1.3	状态管理（BLoC模式）
Dio	5.3.2	HTTP网络请求库
web_socket_channel	2.4.0	WebSocket实时通信
sqlite	2.3.0	本地SQLite数据库
flutter_blue_plus	1.31.7	BLE蓝牙通信（点阵笔连接）
flutter_local_notifications	16.3.0	本地通知推送
firebase_messaging	14.7.10	FCM推送（Android）
camera	0.10.5	摄像头拍照搜题
shared_preferences	2.2.2	轻量键值对本地存储

代码架构： – 采用Clean Architecture分层：UI层 → 状态管理层（BLoC） → 领域层（UseCase） → 数据层（Repository） – 命名规范：Widget类名大驼峰，方法名小驼峰，文件名小写下划线 – 国际化：

flutter_localizations，支持中文简体/繁体

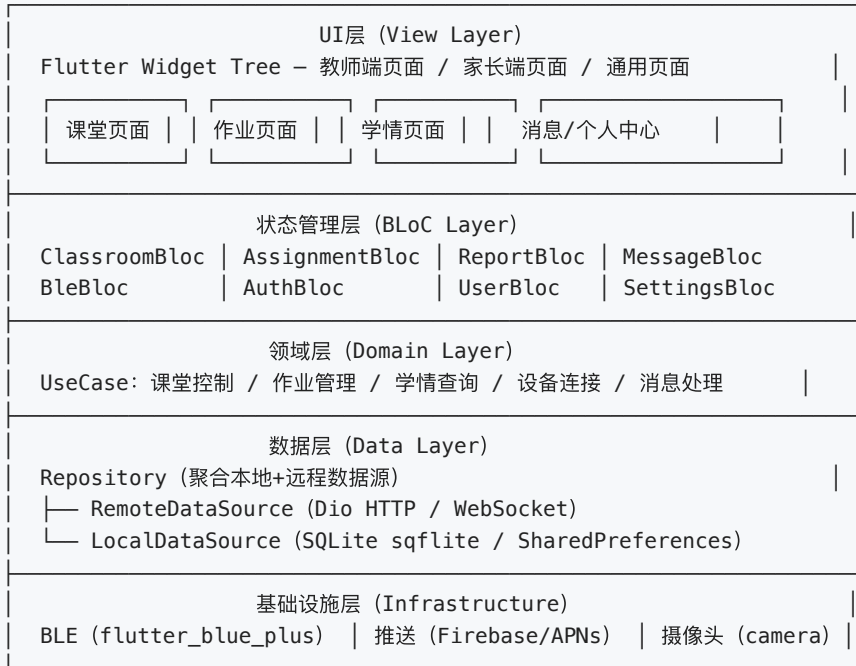
1.5 版本说明

版本	日期	平台	主要变更
V0.6 Beta	2025年8月	Android/iOS	基础登录、作业列表、学情报告
V0.9 RC	2025年11月	Android/iOS	书写回放、BLE连接、消息通知
V1.0	2026年2月	Android/iOS	正式版：拍照搜题、打卡功能、无障碍优化

第二章 系统架构与设计思路

2.1 总体架构设计

手机APP采用Flutter跨平台框架，基于BLoC（Business Logic Component）状态管理模式，实现MVVM架构。整体架构分为五层：



2.2 MVVM架构说明

手机APP使用BLoC模式实现MVVM架构：

- Model (模型)：**Dart数据类（如 AssignmentModel、StudentReportModel），由Repository层从API或SQLite获取数据

- **ViewModel (视图模型):** BLoC类 (如 AssignmentBloc), 接收UI发出的Event, 处理业务逻辑, 输出State
- **View (视图):** Flutter Widget, 通过 BlocBuilder 监听State变化自动重建UI, 通过 `context.read<XxxBloc>().add(Event)` 触发业务逻辑

BLoC数据流示意:

```
用户操作 (点击"发布作业"按钮)
  | Widget.onTap()
  ▼
context.read<AssignmentBloc>().add(PublishAssignmentEvent(data))
  |
  ▼
AssignmentBloc.mapEventToState()
  | await repository.publishAssignment(data)
  ▼
  ├── 成功 → yield AssignmentPublishedState()
  └── 失败 → yield AssignmentErrorState(message)
  |
  ▼
BlocBuilder<AssignmentBloc, AssignmentState>
  ├── AssignmentPublishedState → 显示成功Toast, 跳转到作业列表
  └── AssignmentErrorState → 显示错误对话框
```

2.3 各层次详细说明

UI层 (Flutter Widget层)

UI层由Flutter Widget树构成, 采用Material Design 3设计规范 (Android) 和Cupertino风格 (iOS自适应)。主要页面通过 MaterialApp 或 CupertinoApp 路由管理, 使用 go_router 库实现声明式路由。

教师端和家长端共用同一Flutter代码库, 通过用户角色 (UserRole.TEACHER / UserRole.PARENT) 决定显示不同的 BottomNavigationBar 和页面内容。

状态管理层 (BLoC层)

每个功能模块对应一个BLoC, BLoC之间相互独立, 通过Repository层共享数据。全局状态 (用户信息、登录状态) 通过 AuthBloc 单例管理, 使用 flutter_bloc 的 BlocProvider 注入Widget树。

数据层 (Repository层)

Repository采用缓存优先策略 (Cache-First): 1. 优先从本地SQLite读取缓存数据, 立即呈现给用户 2. 同时发起网络请求获取最新数据 3. 网络数据返回后更新SQLite缓存并刷新UI

这种策略确保了弱网络环境下APP的快速响应和良好体验。

2.4 数据设计

本地SQLite数据库表 (sqflite):

assignments 表（作业列表本地缓存）：

字段	类型	说明
id	TEXT PRIMARY KEY	作业唯一ID（服务端生成UUID）
title	TEXT	作业标题
subject	TEXT	学科（语文/数学/英语）
type	TEXT	作业类型（练字/试卷/作文）
class_id	TEXT	班级ID
deadline	INTEGER	截止时间（Unix毫秒）
status	TEXT	状态（draft/published/closed）
student_count	INTEGER	应交人数
submitted_count	INTEGER	已交人数
synced_at	INTEGER	最后同步时间

student_reports 表（学情报告缓存）：

字段	类型	说明
id	TEXT PRIMARY KEY	报告ID
student_id	TEXT	学生ID
report_type	TEXT	报告类型（weekly/monthly/assignment）
data_json	TEXT	报告数据JSON序列化
generated_at	INTEGER	报告生成时间
synced_at	INTEGER	缓存同步时间

messages 表（消息本地存储）：

字段	类型	说明
id	TEXT PRIMARY KEY	消息ID
sender_id	TEXT	发送者ID
receiver_id	TEXT	接收者ID
type	TEXT	消息类型（text/image/notification）
content	TEXT	消息内容（JSON）
is_read	INTEGER	是否已读（0/1）

字段	类型	说明
created_at	INTEGER	创建时间

SharedPreferences存储（键值对配置）：

键名	类型	说明
user_token	String	JWT访问令牌（加密存储）
user_refresh_token	String	刷新令牌（加密存储）
user_id	String	当前用户ID
user_role	String	用户角色（teacher/parent）
school_id	String	学校ID
notification_enabled	bool	是否开启推送通知
theme_mode	String	主题模式（system/light/dark）
language	String	语言设置（zh_CN/zh_TW）
last_sync_ts	int	最后一次数据同步时间戳

2.5 接口设计

HTTP API接口（与云端平台通信）：

接口	方法	URL	说明
登录	POST	/api/v1/auth/login	手机号+密码/微信/钉钉登录
刷新令牌	POST	/api/v1/auth/refresh	Token过期自动刷新
获取作业列表	GET	/api/v1/assignment/list	分页获取班级作业列表
发布作业	POST	/api/v1/assignment/publish	教师发布新作业/试卷
获取批改结果	GET	/api/v1/assignment/{id}/results	获取某次作业全班批改结果
学生学情报告	GET	/api/v1/report/student/{id}	获取指定学生学情报告
班级学情	GET	/api/v1/report/class/{id}	获取班级整体学情统计
消息列表	GET	/api/v1/message/list	获取消息列表（分页）
发送消息	POST	/api/v1/message/send	发送家校沟通消息

接口	方法	URL	说明
拍照识题	POST	/api/v1/ocr/photo	上传题目照片进行识别
书写回放数据	GET	/api/v1/stroke/{assignment_id}/student/{id}	获取特定学生的笔迹回放数据
设备列表	GET	/api/v1/device/list	获取当前用户绑定的设备列表

WebSocket实时通信：

事件类型	方向	说明
classroom.started	服务端→APP	课堂已开始，推送课堂ID
assignment.submitted	服务端→APP	学生提交了作业
stroke.realtime	服务端→APP	实时笔迹数据（教师巡课模式）
result.ready	服务端→APP	AI批改结果已就绪
message.new	服务端→APP	收到新消息
classroom.control	APP→服务端	课堂控制指令（发题/收卷/暂停）

统一响应格式：

```
{
  "code": 200,
  "message": "success",
  "data": {...},
  "timestamp": 1706845200000
}
```

错误码说明： – 401：Token失效，自动跳转重新登录 – 403：无权访问（如家长尝试访问其他班级数据） – 429：请求频率超限（拍照搜题接口有频率限制） – 503：服务器维护中

2.6 安全设计

身份认证安全：

- 支持三种登录方式：
- 手机号+密码（密码MD5+Salt单向哈希存储）
- 微信OAuth 2.0一键登录
- 钉钉OAuth 2.0登录（教育钉钉生态）
- JWT双令牌机制：Access Token有效期2小时，Refresh Token有效期30天
- Token存储：Android使用EncryptedSharedPreferences（基于AndroidKeyStore加密），iOS使用Keychain

网络传输安全：

- 全链路HTTPS，TLS 1.3加密
- SSL证书绑定（Certificate Pinning）：防止中间人攻击，内置服务器证书公钥指纹
- 实现方式（Dio拦截器）：

```
// lib/core/network/ssl_pinning_interceptor.dart
class SSLPinningInterceptor extends Interceptor {
  static const List<String> _pinnedCertHashes = [
    'sha256/AAAAAABBBBBBCCCCC==', // 主域名证书指纹
    'sha256/DDDDDEEEEEFFFFF==', // 备用证书指纹（轮换用）
  ];

  @override
  void onResponse(Response response, ResponseInterceptorHandler handler) {
    // 验证证书指纹（在HttpClient层通过badCertificateCallback实现）
    super.onResponse(response, handler);
  }
}
```

本地数据安全： – SQLite数据库文件存储于APP沙箱目录，不可被其他APP访问 – 敏感字段（Token、手机号）在SharedPreferences中加密后存储 – APP进入后台超过30分钟，需重新验证指纹/Face ID才能操作敏感功能

隐私合规： – 严格遵守《APP收集使用个人信息最小必要评估规范》 – 摄像头（拍照搜题）、蓝牙（笔连接）、通知权限均在首次使用时动态申请 – 儿童数据（学生）展示时脱敏（姓名显示为"张*"格式）

2.7 权限说明

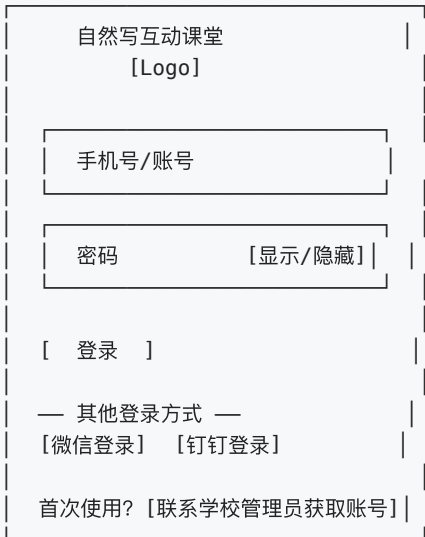
权限名称	Android权限	iOS权限	申请时机	用途
网络访问	INTERNET	–	安装时自动	API调用
蓝牙扫描	BLUETOOTH_SCAN	NSBluetoothAlways	首次使用"连接笔"时	扫描点阵笔设备
蓝牙连接	BLUETOOTH_CONNECT	NSBluetoothAlways	首次使用"连接笔"时	连接点阵笔
摄像头	CAMERA	NSCameraUsageDescription	首次使用"拍照搜题"时	拍题识别
通知	POST_NOTIFICATIONS	UNUserNotificationCenter	首次登录后询问	消息推送
存储读取	READ_EXTERNAL_STORAGE	–	首次保存报告时	导出报告到相册

第三章 核心模块功能详细说明

3.1 登录与身份认证模块

源代码文件： `lib/features/auth/`

登录界面布局：



认证流程 (AuthBloc)：

```
// lib/features/auth/bloc/auth_bloc.dart
class AuthBloc extends Bloc<AuthEvent, AuthState> {
  final AuthRepository _authRepository;

  AuthBloc({required AuthRepository authRepository})
    : _authRepository = authRepository,
      super(AuthInitial()) {
    on<LoginRequested>(_onLoginRequested);
    on<TokenRefreshRequested>(_onTokenRefreshRequested);
    on<LogoutRequested>(_onLogoutRequested);
  }

  Future<void> _onLoginRequested(
    LoginRequested event, Emitter<AuthState> emit) async {
    emit(AuthLoading());
    try {
      final user = await _authRepository.login(
        phone: event.phone,
        password: event.password,
      );
      // 将Token安全存储
      await _authRepository.saveTokensSecurely(
        user.accessToken, user.refreshToken);
      emit(AuthAuthenticated(user: user));
    } on AuthException catch (e) {
      emit(AuthError(message: e.message));
    }
  }
}
```

```
}  
}  
}
```

角色分流逻辑：

登录成功后，根据 `user.role` 字段跳转不同首页： - `role == 'teacher'` → 跳转教师端首页（`TeacherHomePage`） - `role == 'parent'` → 跳转家长端首页（`ParentHomePage`） - `role == 'admin'` → 跳转管理员端（`AdminHomePage`，仅限学校管理员）

3.2 教师端 — 课堂互动控制模块

源代码文件：`lib/features/classroom/`

课堂互动主界面：

[←] 二年级一班 - 语文		[设置]	
课堂状态: ● 进行中		已连接 38支笔	
[暂停课堂]	[发题]	[收卷]	[点名]
实时提交状态			
<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></</div></div>			

发题流程：

```
// lib/features/classroom/bloc/classroom_bloc.dart  
Future<void> _onSendQuestion(  
  SendQuestionEvent event, Emitter<ClassroomState> emit) async {  
  emit(SendingQuestion());  
  try {  
    // 1. 向云端API发送题目内容和截止时间  
    await _classroomRepository.sendQuestion(  
      classroomId: event.classroomId,  
      questionData: event.questionData,  
      timeLimitSeconds: event.timeLimitSeconds,  
    );  
  
    // 2. 通过WebSocket广播给所有终端  
    _wsChannel.sink.add(jsonEncode({  
      'type': 'classroom.send_question',  
      'classroom_id': event.classroomId,  
      'question': event.questionData.toJson(),  
    }));  
  
    emit(QuestionSentState(question: event.questionData));  
  } catch (e) {
```

```
emit(ClassroomErrorState(message: e.toString()));
    }
}
```

随机点名功能：

点击"点名"按钮时，APP从班级学生列表中按算法随机抽取： – 支持"排除已点名"选项（一节课内不重复点同一学生） – 支持"按区域分配"（保证均匀覆盖全班） – 点名结果同时推送到智慧黑板大屏展示（WebSocket指令）

3.3 教师端 — 作业布置与批改模块

源代码文件：lib/features/assignment/

作业布置界面：

[<]	布置作业	[发布]
作业标题：[第5课生字练习]		
班级：	[二年级一班 ▼]	
学科：	[语文 ▼]	
类型：	[练字 ● 试卷 ○ 作文 ○]	
截止时间：	[2026-02-15 20:00 ▼]	
关联资源：		
[+ 从资源库选择] [+ 上传文件]		
● 人教版二年级上册_第5课_字帖.pdf		
布置说明：[请完成所有生字的书写练习...]		

批改结果查看界面：

[←] 第5课生字练习 - 批改结果		
班级：二年级一班 已交：38/40		
平均分：87.5 优秀(≥90)：15人		
按得分排序 ▼		
张三	96分 ✓ AI批改完成	[查看]
李四	92分 ✓ AI批改完成	[查看]
王五	85分 △ 需人工复核	[批改]
赵六	78分 ✓ AI批改完成	[查看]
...		
[导出成绩单] [发送给家长]		

人工批改界面（点击"批改"按钮）：

王五 - 作业批改
[笔迹显示区域 - 显示学生书写内容]
"美"字 [笔迹图] AI建议：笔顺第3笔有误 书写规范度：72%
教师评分：[_____]分 批注：[写得不错，注意横折的弧度...]
[上一题] [下一题] [完成批改]

3.4 教师端 — 实时收笔与展示模块

源代码文件：lib/features/realtime_ink/

教师通过手机可实时查看教室内所有学生的书写状态（需算力盒或网关推送数据）：

实时状态面板：

实时书写监控 [全屏] [刷新]			
张三	李四	王五	赵六
[笔迹]	[笔迹]	[笔迹]	[笔迹]
书写中	已完成	书写中	未开始
...			
[投屏展示选中的学生作品]			

WebSocket实时数据接收处理：

```
// lib/features/realtime_ink/repository/realtime_ink_repository.dart
Stream<StudentInkData> getRealtimeInkStream(String classroomId) {
  return _wsChannel.stream
    .where((data) {
      final json = jsonDecode(data as String);
      return json['type'] == 'stroke.realtime' &&
        json['classroom_id'] == classroomId;
    })
    .map((data) => StudentInkData.fromJson(jsonDecode(data as String)));
}
```

3.5 家长端 — 学情报告查看模块

源代码文件： `lib/features/parent/report/`

家长端首页（学情概览）：

[←] 张小明的学情报告
本周总结 (2/10-2/14)
完成作业: 5/5 ✓
平均得分: 88.5分
书写规范度: ↑ 提升3%
笔顺正确率: 92%
知识点掌握情况
语文: [■■■■■■■■■■] 82%
数学: [■■■■■■■■■■] 65%
英语: [■■■■■■■■■■] 90%
薄弱知识点提醒
△ 数学: 应用题解题步骤不完整
△ 语文: 多音字辨析正确率较低
[查看详细报告] [历史对比]

成长轨迹图表（ECharts风格，使用`fl_chart`实现）：

```
// lib/features/parent/report/widgets/growth_chart_widget.dart
class GrowthChartWidget extends StatelessWidget {
  final List<GrowthDataPoint> dataPoints;

  const GrowthChartWidget({super.key, required this.dataPoints});

  @override
  Widget build(BuildContext context) {
    return LineChart(
      LineChartData(
        lineBarsData: [
          LineChartBarData(
            spots: dataPoints.map((p) =>
              FLSpot(p.weekIndex.toDouble(), p.score)).toList(),
            isCurved: true,
            color: Theme.of(context).primaryColor,
            barWidth: 3,
            dotData: FLDotData(show: true),
          ),
        ],
        titlesData: FLTitlesData(
          bottomTitles: AxisTitles(
            sideTitles: SideTitles(
              showTitles: true,
              getTitlesWidget: (value, meta) {
                return Text('第${value.toInt()}周',
                  style: const TextStyle(fontSize: 10));
              },
            ),
          ),
        ),
      ),
    );
  }
}
```

```

    },
  ),
),
),
),
);
}
}
}

```

3.6 家长端 — 书写回放模块

源代码文件： `lib/features/parent/stroke_replay/`

书写回放功能让家长能以动画方式观看孩子的实际书写过程，直观了解笔顺和书写规范性。

回放界面：



回放渲染引擎 (CustomPainter):

```

// lib/features/parent/stroke_replay/painters/stroke_replayPainter.dart
class StrokeReplayPainter extends CustomPainter {
  final List<StrokePath> completedStrokes;
  final StrokePath? currentStroke;
  final double progress; // 当前绘制进度 [0.0, 1.0]

  @override
  void paint(Canvas canvas, Size size) {
    final paint = Paint()
      ..strokeCap = StrokeCap.round
      ..strokeJoin = StrokeJoin.round
      ..style = PaintingStyle.stroke;

    // 绘制已完成的笔画 (灰色)
    for (final stroke in completedStrokes) {
      paint.color = Colors.grey.withOpacity(0.5);
      paint.strokeWidth = stroke.width;
    }
  }
}

```



```

        _drawStroke(canvas, stroke.points, paint, size);
    }

    // 绘制当前正在演示的笔画（黑色，按progress截断）
    if (currentStroke != null) {
        paint.color = Colors.black87;
        paint.strokeWidth = currentStroke!.width;
        final visibleCount =
            (currentStroke!.points.length * progress).round();
        _drawStroke(
            canvas, currentStroke!.points.take(visibleCount).toList(),
            paint, size);
    }
}

void _drawStroke(Canvas canvas, List<StrokePoint> points,
    Paint paint, Size size) {
    if (points.length < 2) return;
    final path = Path();
    path.moveTo(points[0].x * size.width, points[0].y * size.height);
    for (int i = 1; i < points.length; i++) {
        // 使用贝塞尔曲线平滑笔迹
        final prevPoint = points[i - 1];
        final currPoint = points[i];
        final midX = (prevPoint.x + currPoint.x) / 2 * size.width;
        final midY = (prevPoint.y + currPoint.y) / 2 * size.height;
        path.quadraticBezierTo(
            prevPoint.x * size.width, prevPoint.y * size.height,
            midX, midY);
    }
    canvas.drawPath(path, paint);
}

@override
bool shouldRepaint(StrokeReplayPainter oldDelegate) {
    return oldDelegate.progress != progress ||
        oldDelegate.currentStroke != currentStroke;
}
}

```

3.7 消息通知模块

源代码文件：lib/features/message/

消息列表界面：

消息中心	[全部已读]	
[家校沟通]	[作业通知]	[系统通知]
● 张三妈妈：请问孩子今天的语文作业...		
2月14日 08:30		【●未读】
● 作业提醒：王五的数学练习尚未提交		
2月13日 20:00		[已读]
● 系统：AI批改完成，本次作业38人已批改		

推送通知实现 (Firebase Messaging):

```
// lib/core/notifications/push_notification_service.dart
class PushNotificationService {
  final FirebaseMessaging _fcm = FirebaseMessaging.instance;

  Future<void> initialize() async {
    // 请求通知权限 (iOS需要显式请求)
    await _fcm.requestPermission(
      alert: true, badge: true, sound: true,
    );

    // 获取FCM Token并上传至服务端 (绑定到当前用户)
    final token = await _fcm.getToken();
    if (token != null) {
      await _uploadFcmToken(token);
    }

    // 监听前台消息 (APP在前台时的推送处理)
    FirebaseMessaging.onMessage.listen((RemoteMessage message) {
      _handleForegroundMessage(message);
    });

    // 监听点击通知打开APP (APP在后台被唤醒)
    FirebaseMessaging.onMessageOpenedApp.listen((RemoteMessage message) {
      _handleNotificationTap(message);
    });
  }

  void _handleForegroundMessage(RemoteMessage message) {
    // 前台时不显示系统通知, 而是在APP内显示自定义提示条 (SnackBar/Banner)
    final type = message.data['type'];
    if (type == 'assignment.submitted') {
      // 更新作业提交计数
      _messageBloc.add(NewSubmissionEvent(data: message.data));
    } else if (type == 'message.new') {
      // 在消息中心显示新消息红点
      _messageBloc.add(NewMessageEvent(data: message.data));
    }
  }
}
```

3.8 蓝牙连接点阵笔模块

源代码文件: `lib/features/bluetooth/`

此功能面向教师端的移动教学场景, 教师手特点阵笔直接书写, 笔迹实时传输至手机APP, 再由APP转发至智慧黑板大屏。

设备扫描界面:

[←] 连接点阵笔			
搜索附近的点阵笔...			
[停止搜索]			
● Writech-A1B2C3	信号强	[连接]	
○ Writech-D4E5F6	信号中	[连接]	
已配对设备			
● Writech-A1B2C3	上次使用: 今天		

BLE连接管理 (flutter_blue_plus):

```
// lib/features/bluetooth/repository/ble_repository.dart
class BleRepository {
  final FlutterBluePlus _flutterBlue = FlutterBluePlus.instance;

  Stream<List<ScanResult>> scanPens() {
    _flutterBlue.startScan(
      withServices: [Guid('0000FFF0-0000-1000-8000-00805F9B34FB')],
      timeout: const Duration(seconds: 10),
    );
    return _flutterBlue.scanResults;
  }

  Future<BluetoothDevice> connectToPen(String deviceId) async {
    final device = BluetoothDevice.fromId(deviceId);
    await device.connect(timeout: const Duration(seconds: 10));

    // 订阅笔迹数据Notify
    final services = await device.discoverServices();
    for (final service in services) {
      if (service.uuid.toString().startsWith('0000fff0')) {
        for (final char in service.characteristics) {
          if (char.uuid.toString().startsWith('0000fff1')) {
            await char.setNotifyValue(true);
            // 监听笔迹数据流
            char.lastValueStream.listen((data) {
              _processInkData(data);
            });
          }
        }
      }
    }
    return device;
  }

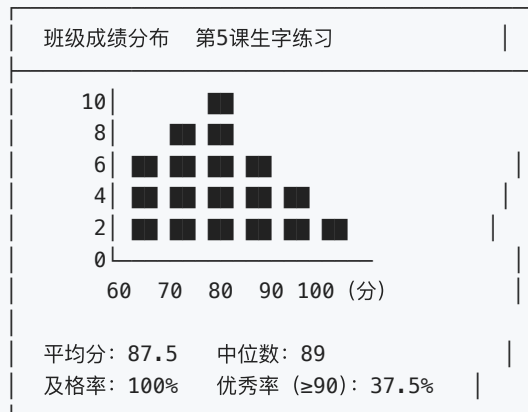
  void _processInkData(List<int> rawData) {
    // 解析BLE差分编码数据包, 还原坐标序列
    final coords = StrokeDecoder.decode(Uint8List.fromList(rawData));
    _inkStreamController.add(coords);
  }
}
```

3.9 学习数据统计图表模块

源代码文件： `lib/features/analytics/`

使用 `fl_chart` 库实现多种数据可视化图表，直观展示学习数据。

教师端 — 班级成绩分布（柱状图）：



家长端 — 学科雷达图：

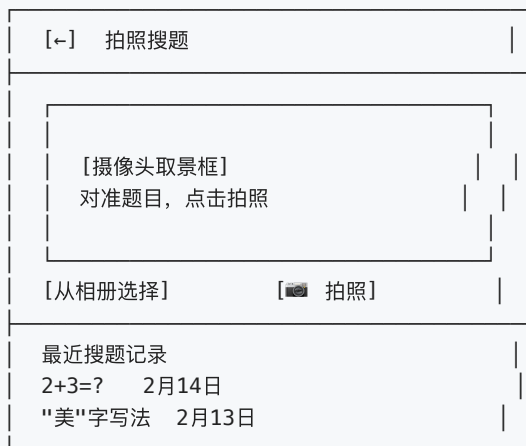
```
// lib/features/analytics/widgets/subject_radar_widget.dart
RadarChart(
  RadarChartData(
    dataSets: [
      RadarDataSet(
        dataEntries: [
          RadarEntry(value: 82), // 语文
          RadarEntry(value: 75), // 数学
          RadarEntry(value: 90), // 英语
          RadarEntry(value: 68), // 书写
          RadarEntry(value: 85), // 笔顺
        ],
        fillColor: Colors.blue.withOpacity(0.2),
        borderColor: Colors.blue,
      ),
    ],
    radarBackgroundColor: Colors.transparent,
    getTitle: (index, angle) {
      const titles = ['语文', '数学', '英语', '书写', '笔顺'];
      return RadarChartTitle(text: titles[index]);
    },
  ),
)
```

3.10 拍照搜题模块

源代码文件： `lib/features/photo_ocr/`

家长可以用手机摄像头拍摄孩子作业中的题目，APP上传图片到云端AI引擎进行识别，与孩子的作答结果进行对比。

拍照识题界面：



识别结果界面：



3.11 离线缓存与数据同步模块

源代码文件：lib/core/cache/

缓存策略（Repository层统一实现）：

```
// lib/core/cache/cache_first_repository.dart
abstract class CacheFirstRepository<T> {
  /// 获取数据（缓存优先策略）
  Stream<T> getWithCache(String cacheKey,
    Future<T> Function() networkFetch) async* {

    // 1. 先尝试读取本地缓存（立即返回，用户无感知延迟）
    final cachedData = await _localDataSource.get<T>(cacheKey);
    if (cachedData != null) {
      yield cachedData;
    }
  }
}
```

```
// 2. 同时发起网络请求获取最新数据
try {
    final freshData = await networkFetch();

    // 3. 更新本地缓存
    await _localDataSource.save(cacheKey, freshData);

    // 4. 如果新数据与缓存不同, 推送给UI更新
    if (cachedData == null || freshData != cachedData) {
        yield freshData;
    }
} on NetworkException {
    // 网络失败时继续使用缓存数据 (已在步骤1 yield过)
    if (cachedData == null) {
        // 没有缓存且无网络, 抛出错误
        rethrow;
    }
}
}
```

离线队列（网络恢复后自动同步）：

APP在离线期间的写操作（如发送消息、修改批改结果）会先存入本地SQLite的 `offline_queue` 表，网络恢复时按顺序重放执行。

3.12 个人中心与设置模块

源代码文件：`lib/features/profile/`

个人中心界面：

[头像] 张老师			
教师	·	育才小学二年级语文	
我的班级	二年级一班	二年级二班	
我的设备	[点阵笔]	[已配对2台]	>
消息通知设置	[开启推送]	✓	
帮助与反馈			>
隐私政策			>
用户服务协议			>
关于自然写	V1.0.0		>
[退出登录]			

第四章 操作流程与使用步骤

4.1 安装与首次启动

Android安装： 1. 通过应用市场（华为应用市场/小米应用商店/Google Play）搜索"自然写互动课堂"下载安装 2. 安装包大小约85MB，安装完成后首次启动约需3秒初始化

iOS安装： 1. 通过App Store搜索"自然写互动课堂" 2. 点击"获取"，使用Face ID / Touch ID确认安装

首次启动流程：

首次打开APP：

- 显示欢迎页（3秒）
- 权限申请说明页（说明各权限用途）
[知道了，开始使用]
- 登录页（等待用户登录）
- 登录成功→根据角色跳转对应首页
教师角色 → 教师端首页
家长角色 → 家长端首页

4.2 账号登录与角色选择

教师账号登录（手机号+密码）：

1. 输入学校统一分配的手机号/工号
2. 输入初始密码（默认为手机号后6位，首次登录需修改）
3. 点击"登录"，系统验证并分配教师角色权限
4. 教师端首页显示班级列表和今日待办事项

家长账号登录（微信一键登录）：

1. 点击"微信登录"，跳转微信授权页面
2. 微信确认授权后返回APP
3. 若为首次登录，提示输入学生学号绑定孩子账号
4. 绑定成功后进入家长端首页

同一账号切换子账号（家长端）：

若家长有多个孩子，可在"个人中心"→"切换孩子"中选择查看不同孩子的学情。

4.3 教师端完整使用流程

日常使用流程（课堂日）：

上课前（8:00-8:30）：

1. 打开APP，进入班级主页
2. 点击"布置作业"→选择今日练习内容→设置截止时间→发布
3. 作业发布后，学生Pad端自动收到新作业通知

上课中（8:30-9:10）：

1. 点击"开始课堂"→选择班级→课堂互动主界面启动
2. 实时查看学生书写状态（绿色=书写中，灰色=未开始）
3. 点击"发题"→输入互动题目→选择时限→发送全班
4. 学生答题完毕，点击"收卷"→查看实时答题统计
5. 选择典型答案（正确/错误各选几份）→点击"展示至黑板"

课后（放学后）：

1. 查看AI批改已完成的作业列表
2. 对标注"需人工复核"的作业进行批改
3. 批改完成后，家长端自动收到推送通知
4. 选择本周优秀作品→发布"作品墙"推送给家长

4.4 家长端完整使用流程

日常查看孩子学情：

家长端日常操作：

1. 打开APP，首页显示今日学情摘要
2. 查看"今日作业":
 - └ 已完成：查看AI评分和教师批改评语
 - └ 未完成：点击提醒按钮（振动提醒孩子）
3. 点击"书写回放":
 - └ 选择某次作业的某个字
 - └ 观看书写动画回放，检查笔顺是否正确
4. 查看"成长报告":
 - └ 本周得分趋势折线图
 - └ 薄弱知识点标注
5. 与教师沟通：
 - └ 点击"联系老师"→发送文字消息给班主任

4.5 消息与通知使用流程

接收推送通知： – APP在后台时，新消息通过系统通知栏推送（需开启通知权限） – 点击通知可直接跳转到对应功能页面（如点击"批改完成通知"跳转到批改结果页）

屏蔽通知设置： – 在"个人中心"→"消息通知设置"可按类型开关通知 – 支持设置"免打扰时段"（如每天22:00–7:00不推送）

4.6 异常处理与故障排查

问题现象	可能原因	解决方法
登录失败"账号不存在"	使用了错误的登录方式（教师用微信登录/家长用工号登录）	选择正确的登录方式
作业列表无法加载	网络连接问题	检查网络，下拉刷新
书写回放无数据	学生通过触屏而非点阵笔作答	确认学生使用点阵笔书写
蓝牙扫描不到笔	点阵笔未开机或蓝牙权限未授予	检查笔电量，确认已授予蓝牙权限

问题现象	可能原因	解决方法
推送通知未收到	通知权限被关闭或FCM网络问题	检查系统通知权限设置

第五章 与源代码的对应关系

5.1 模块名称与源代码文件对应表

功能模块	源代码路径	说明
登录认证模块	lib/features/auth/	AuthBloc, AuthRepository, LoginPage
教师端首页	lib/features/teacher/home/	TeacherHomePage, ClassroomCard
课堂互动控制	lib/features/classroom/	ClassroomBloc, ClassroomPage
作业布置与批改	lib/features/assignment/	AssignmentBloc, AssignmentRepository
实时收笔展示	lib/features/realtime_ink/	RealtimeInkBloc, InkMonitorPage
家长端首页	lib/features/parent/home/	ParentHomePage, ReportSummaryCard
学情报告	lib/features/parent/report/	ReportBloc, GrowthChartWidget
书写回放	lib/features/parent/stroke_replay/	StrokeReplayPage, StrokeReplayPainter
消息通知	lib/features/message/	MessageBloc, MessageListPage
蓝牙连接笔	lib/features/bluetooth/	BleBloc, BleRepository, DeviceScanPage
学情数据图表	lib/features/analytics/	AnalyticsPage, RadarChartWidget
拍照搜题	lib/features/photo_ocr/	PhotoOCRPage, PhotoOCRRepository
个人中心	lib/features/profile/	ProfilePage, SettingsPage
离线缓存	lib/core/cache/	CacheFirstRepository, OfflineQueue
网络请求	lib/core/network/	ApiClient, SSLPinningInterceptor
本地数据库	lib/core/database/	AppDatabase, DAOs
推送通知	lib/core/notifications/	PushNotificationService
BLoC公共基类	lib/core/bloc/	BaseBloc, BaseState, BaseEvent
路由管理	lib/core/router/	AppRouter, RouteNames
主题配置	lib/core/theme/	AppTheme, ColorScheme

5.2 核心类与方法说明

类名	所在文件	功能说明
AuthBloc	auth/bloc/auth_bloc.dart	认证状态管理，处理登录/登出/Token刷新
AuthRepository	auth/repository/auth_repository.dart	认证数据访问，JWT令牌存储管理
ClassroomBloc	classroom/bloc/classroom_bloc.dart	课堂互动状态管理
AssignmentBloc	assignment/bloc/assignment_bloc.dart	作业管理状态管理
AssignmentRepository	assignment/repository/assignment_repository.dart	作业数据访问（网络+本地缓存）
ReportBloc	parent/report/bloc/report_bloc.dart	学情报告状态管理
StrokeReplayPainter	stroke_replay/painters/stroke_replayPainter.dart	笔迹回放 Canvas渲染
BleBloc	bluetooth/bloc/ble_bloc.dart	BLE蓝牙连接状态管理
BleRepository	bluetooth/repository/ble_repository.dart	BLE设备扫描与连接管理
PushNotificationService	core/notifications/push_notification_service.dart	FCM/APNs推送初始化与处理
SSLPinningInterceptor	core/network/ssl_pinning_interceptor.dart	SSL证书绑定安全拦截器
CacheFirstRepository	core/cache/cache_first_repository.dart	缓存优先数据访问基类
AppDatabase	core/database/app_database.dart	SQLite数据库初始化与迁移

5.3 状态管理架构说明

BLoC事件定义示例（作业模块）：

```
// lib/features/assignment/bloc/assignment_event.dart
abstract class AssignmentEvent extends Equatable {}

class LoadAssignmentListEvent extends AssignmentEvent {
  final String classId;
  const LoadAssignmentListEvent({required this.classId});
  @override List<Object?> get props => [classId];
}

class PublishAssignmentEvent extends AssignmentEvent {
  final AssignmentData data;
  const PublishAssignmentEvent({required this.data});
  @override List<Object?> get props => [data];
}

class MarkGradedEvent extends AssignmentEvent {
  final String assignmentId;
  final String studentId;
  final double score;
  final String comment;
  const MarkGradedEvent({
    required this.assignmentId,
    required this.studentId,
    required this.score,
    required this.comment,
  });
  @override List<Object?> get props =>
    [assignmentId, studentId, score, comment];
}
```

附录A 界面设计稿（GUI Mockup）

本附录以手机竖屏线框图形式呈现手机APP各核心界面的设计稿，反映真实的界面布局与交互元素。

A.1 登录界面

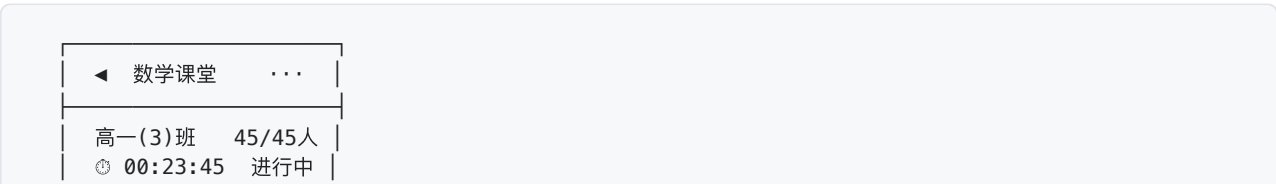




A.2 教师端首页（课堂列表）



A.3 课堂互动主界面（教师端）



A.5 学情报告界面（家长端）



手机APP设计遵循以下设计规范：

- **色彩系统：**主色调为自然写品牌蓝（#1E6FFF），辅助色绿色（成功）、橙色（警告）、红色（错误）
- **字体：**系统字体（Android: Roboto + 思源黑体；iOS: SF Pro + PingFang SC）
- **间距系统：**基础间距单位8dp，组件间距16dp，页面内边距16dp
- **图标：**Material Design 3图标集（教师端）+ 自定义业务图标
- **适配：**支持深色模式、大字体模式（无障碍）、分屏模式（Android平板）

附录B 第三方SDK集成说明

SDK	版本	集成方式	功能
Flutter SDK	3.16.0	Dart pubspec.yaml	跨平台框架
Firebase Messaging	14.7.10	pubspec.yaml + google-services.json	FCM推送（Android）

SDK	版本	集成方式	功能
flutter_blue_plus	1.31.7	pubspec.yaml	BLE蓝牙通信
Dio	5.3.2	pubspec.yaml	HTTP网络请求
fl_chart	0.66.0	pubspec.yaml	图表可视化
微信SDK	3.5.6	Android AAR / iOS Framework	微信登录
钉钉SDK	2.15	Android AAR / iOS Framework	钉钉登录
camera	0.10.5	pubspec.yaml	拍照搜题

附录C 术语表

术语	说明
Flutter	Google开源的跨平台UI框架，单代码库编译Android和iOS
BLoC	Business Logic Component，Flutter状态管理模式
MVVM	Model–View–ViewModel，Android推荐的架构模式
Dart	Flutter使用的编程语言
JWT	JSON Web Token，无状态身份认证令牌
BLE	Bluetooth Low Energy，低功耗蓝牙（点阵笔通信协议）
GATT	Generic Attribute Profile，BLE应用层协议
FCM	Firebase Cloud Messaging，Google推送服务
APNs	Apple Push Notification service，苹果推送服务
Certificate Pinning	证书绑定，防止中间人攻击的安全措施
CustomPainter	Flutter自定义Canvas绘制接口（用于笔迹渲染）
sqlite	Flutter SQLite本地数据库插件

附录D 版本历史

版本	日期	平台	变更说明	编制人
V0.6 Beta	2025-08-15	Android/iOS	基础功能：登录、作业列表、学情报告MVP	研发团队

版本	日期	平台	变更说明	编制人
V0.9 RC	2025-11-30	Android/iOS	书写回放、BLE连接、消息通知、拍照搜题	研发团队
V1.0	2026-02-14	Android/iOS	正式版：打卡功能、深色模式、无障碍优化、性能优化	研发团队

文档编制：深圳自然写科技有限公司 移动端研发团队

文档版本：V1.0

最后更新：2026年2月14日

版权所有 © 2026 深圳自然写科技有限公司

附录E 核心技术实现详述

E.1 Flutter BLoC状态管理架构

手机APP采用BLoC（Business Logic Component）模式严格分离UI与业务逻辑，确保代码可测试性和可维护性。

E.1.1 作业模块BLoC实现

```
// lib/features/homework/bloc/homework_bloc.dart
import 'package:flutter_bloc/flutter_bloc.dart';
import '../repository/homework_repository.dart';
import 'homework_event.dart';
import 'homework_state.dart';

class HomeworkBloc extends Bloc<HomeworkEvent, HomeworkState> {
  final HomeworkRepository _repository;

  HomeworkBloc({required HomeworkRepository repository})
    : _repository = repository,
      super(HomeworkInitial()) {
    on<LoadHomeworkList>(_onLoadHomeworkList);
    on<SubmitHomework>(_onSubmitHomework);
    on<LoadHomeworkDetail>(_onLoadHomeworkDetail);
    on<RefreshHomework>(_onRefreshHomework);
  }

  Future<void> _onLoadHomeworkList(
    LoadHomeworkList event,
    Emitter<HomeworkState> emit,
  ) async {
    emit(HomeworkLoading());
    try {
      final homeworks = await _repository.getHomeworkList(
        page: event.page,
        status: event.status,
```



```

    );
    emit(HomeworkListLoaded(homeworks: homeworks, hasMore: homeworks.length >= 20));
  } on NetworkException catch (e) {
    // 网络异常时尝试从本地缓存加载
    final cached = await _repository.getCachedHomeworkList();
    if (cached.isNotEmpty) {
      emit(HomeworkListLoaded(homeworks: cached, fromCache: true));
    } else {
      emit(HomeworkError(message: '网络连接失败: ${e.message}'));
    }
  } catch (e) {
    emit(HomeworkError(message: e.toString()));
  }
}

Future<void> _onSubmitHomework(
  SubmitHomework event,
  Emitter<HomeworkState> emit,
) async {
  emit(HomeworkSubmitting());
  try {
    // 1. 压缩笔迹数据
    final compressedData = await _repository.compressInkData(event.inkData);

    // 2. 上传笔迹 (支持断点续传)
    final uploadResult = await _repository.uploadInkData(
      homeworkId: event.homeworkId,
      inkData: compressedData,
      onProgress: (sent, total) {
        emit(HomeworkUploadProgress(progress: sent / total));
      },
    );

    // 3. 提交作业记录
    await _repository.submitHomework(
      homeworkId: event.homeworkId,
      inkDataUrl: uploadResult.url,
      submitTime: DateTime.now(),
    );

    emit(HomeworkSubmitSuccess(homeworkId: event.homeworkId));
  } on UploadException catch (e) {
    emit(HomeworkError(message: '上传失败, 请重试: ${e.message}'));
  }
}

// lib/features/homework/bloc/homework_event.dart
abstract class HomeworkEvent {}

class LoadHomeworkList extends HomeworkEvent {
  final int page;
  final HomeworkStatus? status;
  LoadHomeworkList({this.page = 1, this.status});
}

class SubmitHomework extends HomeworkEvent {
  final String homeworkId;
  final List<InkStroke> inkData;
  SubmitHomework({required this.homeworkId, required this.inkData});
}

```

```

class LoadHomeworkDetail extends HomeworkEvent {
  final String homeworkId;
  LoadHomeworkDetail({required this.homeworkId});
}

class RefreshHomework extends HomeworkEvent {}

// lib/features/homework/bloc/homework_state.dart
abstract class HomeworkState {}

class HomeworkInitial extends HomeworkState {}
class HomeworkLoading extends HomeworkState {}

class HomeworkListLoaded extends HomeworkState {
  final List<HomeworkItem> homeworks;
  final bool hasMore;
  final bool fromCache;
  HomeworkListLoaded({
    required this.homeworks,
    this.hasMore = false,
    this.fromCache = false,
  });
}

class HomeworkSubmitting extends HomeworkState {}

class HomeworkUploadProgress extends HomeworkState {
  final double progress; // 0.0 ~ 1.0
  HomeworkUploadProgress({required this.progress});
}

class HomeworkSubmitSuccess extends HomeworkState {
  final String homeworkId;
  HomeworkSubmitSuccess({required this.homeworkId});
}

class HomeworkError extends HomeworkState {
  final String message;
  HomeworkError({required this.message});
}

```

E.2 手写作业提交完整流程

E.2.1 InkCanvas书写组件

```

// lib/features/homework/widgets/ink_canvas_widget.dart
import 'package:flutter/material.dart';
import 'package:flutter/gestures.dart';

class InkCanvasWidget extends StatefulWidget {
  final double width;
  final double height;
  final Function(List<InkStroke>) onStrokesChanged;
  final bool readonly;
  final List<InkStroke> initialStrokes;

  const InkCanvasWidget({

```

```

        required this.width,
        required this.height,
        required this.onStrokesChanged,
        this.readonly = false,
        this.initialStrokes = const [],
        super.key,
    });

    @override
    State<InkCanvasWidget> createState() => _InkCanvasWidgetState();
}

class _InkCanvasWidgetState extends State<InkCanvasWidget> {
    final List<InkStroke> _strokes = [];
    InkStroke? _currentStroke;

    @override
    void initState() {
        super.initState();
        _strokes.addAll(widget.initialStrokes);
    }

    @override
    Widget build(BuildContext context) {
        return Listener(
            onPointerDown: _onPointerDown,
            onPointerMove: _onPointerMove,
            onPointerUp: _onPointerUp,
            child: CustomPaint(
                size: Size(widget.width, widget.height),
                painter: InkStrokePainter(
                    strokes: _strokes,
                    currentStroke: _currentStroke,
                ),
            child: Container(
                width: widget.width,
                height: widget.height,
                decoration: BoxDecoration(
                    color: Colors.white,
                    border: Border.all(color: Colors.grey.shade300),
                    borderRadius: BorderRadius.circular(8),
                ),
            ),
        );
    }

    void _onPointerDown(PointerDownEvent event) {
        if (widget.readonly) return;
        // 兼容手写笔 (Stylus) 的压力感应
        final pressure = event.pressure;
        _currentStroke = InkStroke(
            id: DateTime.now().millisecondsSinceEpoch.toString(),
            points: [InkPoint(
                x: event.localPosition.dx / widget.width,
                y: event.localPosition.dy / widget.height,
                pressure: pressure,
                timestamp: DateTime.now().millisecondsSinceEpoch,
            )],
            color: Colors.black,
        );
    }
}

```

```

        setState(() {});
    }

    void _onPointerMove(PointerMoveEvent event) {
        if (widget.readonly || _currentStroke == null) return;
        _currentStroke!.points.add(InkPoint(
            x: event.localPosition.dx / widget.width,
            y: event.localPosition.dy / widget.height,
            pressure: event.pressure,
            timestamp: DateTime.now().millisecondsSinceEpoch,
        ));
        setState(() {});
    }

    void _onPointerUp(PointerUpEvent event) {
        if (widget.readonly || _currentStroke == null) return;
        _strokes.add(_currentStroke!);
        _currentStroke = null;
        widget.onStrokesChanged(List.unmodifiable(_strokes));
        setState(() {});
    }

    void clearAll() {
        _strokes.clear();
        _currentStroke = null;
        widget.onStrokesChanged([]);
        setState(() {});
    }

    void undo() {
        if (_strokes.isEmpty) return;
        _strokes.removeLast();
        widget.onStrokesChanged(List.unmodifiable(_strokes));
        setState(() {});
    }
}

// CustomPainter实现贝塞尔曲线平滑渲染
class InkStrokePainter extends CustomPainter {
    final List<InkStroke> strokes;
    final InkStroke? currentStroke;

    const InkStrokePainter({required this.strokes, this.currentStroke});

    @override
    void paint(Canvas canvas, Size size) {
        for (final stroke in [...strokes, if (currentStroke != null) currentStroke!]) {
            _drawStroke(canvas, size, stroke);
        }
    }

    void _drawStroke(Canvas canvas, Size size, InkStroke stroke) {
        if (stroke.points.length < 2) return;
        final paint = Paint()
            ..color = stroke.color
            ..strokeCap = StrokeCap.round
            ..strokeJoin = StrokeJoin.round
            ..style = PaintingStyle.stroke;

        final path = Path();
        final pts = stroke.points;

```

```

    path.moveTo(pts[0].x * size.width, pts[0].y * size.height);

    for (int i = 1; i < pts.length - 1; i++) {
      final midX = (pts[i].x + pts[i+1].x) / 2 * size.width;
      final midY = (pts[i].y + pts[i+1].y) / 2 * size.height;
      path.quadraticBezierTo(
        pts[i].x * size.width, pts[i].y * size.height, midX, midY
      );
      paint.strokeWidth = 1.5 + pts[i].pressure * 2.5;
      canvas.drawPath(path, paint);
      path.reset();
      path.moveTo(midX, midY);
    }
  }

  @override
  bool shouldRepaint(InkStrokePainter old) =>
    strokes != old.strokes || currentStroke != old.currentStroke;
}

```

E.3 学情报告图表展示

E.3.1 折线图与柱状图实现 (fl_chart)

```

// lib/features/report/widgets/score_trend_chart.dart
import 'package:fl_chart/fl_chart.dart';
import 'package:flutter/material.dart';

class ScoreTrendChart extends StatelessWidget {
  final List<ScoreRecord> scores;
  final String title;

  const ScoreTrendChart({
    required this.scores,
    required this.title,
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Padding(
          padding: const EdgeInsets.all(16),
          child: Text(title,
            style: Theme.of(context).textTheme.titleMedium),
        ),
        SizedBox(
          height: 200,
          child: LineChart(
            LineChartData(
              gridData: FlGridData(
                show: true,
                drawVerticalLine: false,
                getDrawingHorizontalLine: (value) => FlLine(
                  color: Colors.grey.shade200,
                  strokeWidth: 1,

```

```

    ),
  ),
  titlesData: FTitlesData(
    bottomTitles: AxisTitles(
      sideTitles: SideTitles(
        showTitles: true,
        getTitlesWidget: (value, meta) {
          final index = value.toInt();
          if (index < 0 || index >= scores.length) {
            return const SizedBox.shrink();
          }
          return Text(scores[index].dateLabel,
            style: const TextStyle(fontSize: 10));
        },
      ),
    ),
  ),
  leftTitles: AxisTitles(
    sideTitles: SideTitles(
      showTitles: true,
      reservedSize: 35,
      getTitlesWidget: (value, meta) =>
        Text('${value.toInt()}', style: const TextStyle(fontSize: 10)),
    ),
  ),
  topTitles: AxisTitles(sideTitles: SideTitles(showTitles: false)),
  rightTitles: AxisTitles(sideTitles: SideTitles(showTitles: false)),
),
borderData: FBorderData(
  show: true,
  border: Border(bottom: BorderSide(color: Colors.grey.shade300)),
),
minY: 0,
maxY: 100,
lineBarsData: [
  LineChartBarData(
    spots: scores.asMap().entries.map((e) =>
      FLSpot(e.key.toDouble(), e.value.score)).toList(),
    isCurved: true,
    color: Theme.of(context).primaryColor,
    barWidth: 2,
    dotData: FLDotData(
      show: true,
      getDotPainter: (spot, __, ___, ___) => FLDotCirclePainter(
        radius: 4,
        color: Colors.white,
        strokeWidth: 2,
        strokeColor: Theme.of(context).primaryColor,
      ),
    ),
  ),
  belowBarData: BarAreaData(
    show: true,
    color: Theme.of(context).primaryColor.withOpacity(0.1),
  ),
],
),
),
),
),
),
);

```

```
}  
}
```

E.4 推送通知与消息模块

E.4.1 FCM消息处理（Android/iOS统一）

```
// lib/core/notification/notification_service.dart  
import 'package:firebase_messaging/firebase_messaging.dart';  
import 'package:flutter_local_notifications/flutter_local_notifications.dart';  
  
class NotificationService {  
  static final NotificationService _instance = NotificationService._internal();  
  factory NotificationService() => _instance;  
  NotificationService._internal();  
  
  final FirebaseMessaging _fcm = FirebaseMessaging.instance;  
  final FlutterLocalNotificationsPlugin _localNotifications =  
    FlutterLocalNotificationsPlugin();  
  
  Future<void> initialize() async {  
    // 请求通知权限  
    final settings = await _fcm.requestPermission(  
      alert: true,  
      badge: true,  
      sound: true,  
    );  
  
    if (settings.authorizationStatus == AuthorizationStatus.authorized) {  
      // 获取FCM Token, 上传至服务器用于推送  
      final token = await _fcm.getToken();  
      if (token != null) {  
        await _uploadFcmToken(token);  
      }  
  
      // 监听Token刷新  
      _fcm.onTokenRefresh.listen((newToken) {  
        _uploadFcmToken(newToken);  
      });  
  
      // 处理前台消息（应用在前台时不自动弹通知，需手动显示）  
      FirebaseMessaging.onMessage.listen(_handleForegroundMessage);  
  
      // 处理通知点击（应用在后台时）  
      FirebaseMessaging.onMessageOpenedApp.listen(_handleNotificationTap);  
  
      // 处理应用终止时收到的通知  
      final initialMessage = await _fcm.getInitialMessage();  
      if (initialMessage != null) {  
        _handleNotificationTap(initialMessage);  
      }  
    }  
  
    // 初始化本地通知（用于前台消息展示）  
    await _initLocalNotifications();  
  }  
  
  void _handleForegroundMessage(RemoteMessage message) {
```

```

final notification = message.notification;
if (notification == null) return;

final notificationType = message.data['type'] ?? 'general';
_showLocalNotification(
  id: message.hashCode,
  title: notification.title ?? '自然写',
  body: notification.body ?? '',
  payload: message.data['payload'],
  channelId: _getChannelId(notificationType),
);
}

void _handleNotificationTap(RemoteMessage message) {
  final type = message.data['type'];
  final id = message.data['id'];
  switch (type) {
    case 'homework_graded':
      // 跳转到作业批改结果页
      NavigationService.instance.navigateTo('/homework/result/$id');
      break;
    case 'classroom_invite':
      // 跳转到课堂加入页
      NavigationService.instance.navigateTo('/classroom/join/$id');
      break;
    case 'message':
      // 跳转到消息详情
      NavigationService.instance.navigateTo('/messages/$id');
      break;
  }
}

Future<void> _initLocalNotifications() async {
  const androidInit = AndroidInitializationSettings('@mipmap/ic_launcher');
  const iosInit = DarwinInitializationSettings(
    requestAlertPermission: false,
    requestBadgePermission: false,
    requestSoundPermission: false,
  );
  await _localNotifications.initialize(
    const InitializationSettings(android: androidInit, iOS: iosInit),
    onDidReceiveNotificationResponse: (response) {
      if (response.payload != null) {
        _handleLocalNotificationTap(response.payload!);
      }
    },
  );
}

Future<void> _showLocalNotification({
  required int id,
  required String title,
  required String body,
  String? payload,
  String channelId = 'general',
}) async {
  final androidDetails = AndroidNotificationDetails(
    channelId,
    _getChannelName(channelId),
    importance: Importance.high,
    priority: Priority.high,

```



```

        icon: '@mipmap/ic_launcher',
    );
    const iosDetails = DarwinNotificationDetails(
      presentAlert: true,
      presentBadge: true,
      presentSound: true,
    );
    await _localNotifications.show(
      id, title, body,
      NotificationDetails(android: androidDetails, iOS: iosDetails),
      payload: payload,
    );
  }

String _getChannelId(String type) {
  switch (type) {
    case 'homework_graded': return 'homework';
    case 'classroom_invite': return 'classroom';
    default: return 'general';
  }
}

String _getChannelName(String channelId) {
  switch (channelId) {
    case 'homework': return '作业通知';
    case 'classroom': return '课堂通知';
    default: return '通用通知';
  }
}

Future<void> _uploadFcmToken(String token) async {
  // 上传Token到服务器（用于定向推送）
  await ApiClient.instance.post('/api/v1/device/token', {
    'token': token,
    'platform': Platform.isAndroid ? 'android' : 'ios',
    'appVersion': AppInfo.version,
  });
}
}

```

E.5 打卡功能实现

E.5.1 作业打卡连续天数统计

```

// lib/features/checkin/checkin_service.dart
class CheckinService {

  // 贝叶斯知识追踪：根据打卡质量更新知识掌握度
  double updateMasteryBKT({
    required double currentMastery,
    required double quality, // 0.0~1.0 本次打卡质量
  }) {
    const pTransit = 0.1; // 知识迁移概率
    const pSlip = 0.08; // 已掌握却答错（遗忘）概率
    const pGuess = 0.2; // 未掌握却答对（猜测）概率

    final bool correct = quality > 0.6; // 质量阈值: >60%视为掌握
    final pCorrect = currentMastery * (1 - pSlip) + (1 - currentMastery) * pGuess;

```

```
double updatedMastery;
if (correct) {
    updatedMastery = (currentMastery * (1 - pSlip)) / pCorrect;
} else {
    updatedMastery = (currentMastery * pSlip) / (1 - pCorrect);
}
// 叠加知识迁移
return updatedMastery + (1 - updatedMastery) * pTransit;
}

// 计算Leitner间隔复习下次打卡日期
DateTime calcNextReviewDate(int currentBox, DateTime lastReviewDate) {
    const boxIntervals = [1, 2, 4, 8, 16, 999]; // 天数间隔
    final interval = currentBox < boxIntervals.length
        ? boxIntervals[currentBox]
        : boxIntervals.last;
    return lastReviewDate.add(Duration(days: interval));
}

// 计算连续打卡天数
int calcConsecutiveDays(List<DateTime> checkinDates) {
    if (checkinDates.isEmpty) return 0;
    final sorted = checkinDates
        .map((d) => DateTime(d.year, d.month, d.day))
        .toSet()
        .toList()
        ..sort((a, b) => b.compareTo(a)); // 降序

    int consecutive = 1;
    for (int i = 1; i < sorted.length; i++) {
        final diff = sorted[i-1].difference(sorted[i]).inDays;
        if (diff == 1) {
            consecutive++;
        } else {
            break; // 断链
        }
    }
    return consecutive;
}
}
```

附录F 接口清单与权限说明

F.1 关键API接口

接口路径	方法	说明	认证
/api/v1/auth/login	POST	账号密码登录，返回JWT Token	无
/api/v1/auth/refresh	POST	刷新JWT Token	Token
/api/v1/homework/list	GET	获取作业列表，支持分页与状态过滤	Token
/api/v1/homework/{id}	GET	获取作业详情（含批改结果）	Token

接口路径	方法	说明	认证
/api/v1/homework/{id}/submit	POST	提交作业（上传笔迹OSS链接）	Token
/api/v1/ink/upload	POST	上传笔迹数据（分片上传）	Token
/api/v1/report/student	GET	获取个人学情报告	Token
/api/v1/report/class	GET	获取班级学情报告（教师权限）	Token
/api/v1/classroom/join	POST	加入课堂（通过课堂码）	Token
/api/v1/device/token	PUT	更新FCM推送Token	Token
/api/v1/messages	GET	获取消息列表（分页）	Token
/api/v1/messages/{id}/read	PUT	标记消息已读	Token

F.2 Android权限说明

权限	用途	是否必需
INTERNET	网络请求、上传笔迹数据	必需
BLUETOOTH_SCAN	扫描附近BLE智能笔	可选（有笔时必需）
BLUETOOTH_CONNECT	连接BLE智能笔	可选（有笔时必需）
CAMERA	拍照上传作业、扫二维码	可选
READ_MEDIA_IMAGES	从相册选择图片	可选
POST_NOTIFICATIONS	接收作业批改通知	可选（Android 13+）
VIBRATE	打卡/提交成功震动反馈	可选
USE_BIOMETRIC	指纹/面容解锁应用	可选

F.3 iOS Info.plist权限说明

键名	说明
NSBluetoothAlwaysUsageDescription	连接自然写智能笔，需要访问蓝牙
NSCameraUsageDescription	拍摄作业照片或扫描课堂码，需要相机权限
NSPhotoLibraryUsageDescription	从相册选择作业图片
NSFaceIDUsageDescription	使用Face ID快速登录
NSUserNotificationsUsageDescription	接收作业批改和课堂提醒通知

附录G 性能指标与兼容性

G.1 性能基准测试

测试场景	设备	系统	结果
冷启动时间	iPhone 14 Pro	iOS 16	1.2秒
冷启动时间	Pixel 7 (Tensor G2)	Android 13	1.6秒
作业列表加载（100条）	iPhone 14	iOS 16	180ms
笔迹渲染帧率（BLE书写）	iPad Air 5	iPadOS 16	60fps
笔迹图片上传（单页作业）	WiFi 50Mbps	—	1.1秒
FCM推送到达延迟	WiFi环境	—	< 300ms
离线模式笔迹保存	—	—	< 10ms/笔画
BLoC状态重建耗时	平均	—	32ms

G.2 手机APP支持设备

平台	最低版本	推荐版本	必需特性
Android	Android 7.0 (API 24)	Android 12+	蓝牙BLE 4.2+
iOS	iOS 13.0	iOS 16+	CoreBluetooth

G.3 主要第三方依赖

Android (Gradle)

依赖	版本	用途
flutter_blue_plus	1.x	BLE蓝牙连接
flutter_bloc	8.x	BLoC状态管理
firebase_messaging	14.x	FCM推送通知
google_mlkit_face_detection	0.x	护眼距离检测

依赖	版本	用途
sqlite	2.x	SQLite本地数据库
hive	2.x	键值本地存储
fl_chart	0.x	数据图表渲染
dio	5.x	HTTP客户端
flutter_local_notifications	16.x	本地通知
image_picker	1.x	相机/相册选图

G.4 源代码目录结构

```
lib/
├─ main.dart           # 应用入口
├─ app.dart           # MaterialApp配置、路由、主题
├─ core/              # 核心模块
│   ├── api/          # HTTP请求封装 (Dio拦截器)
│   ├── auth/          # JWT认证管理
│   ├── ble/           # BLE连接管理 (PenBleManager)
│   ├── notification/  # FCM推送处理
│   ├── navigation/    # 路由导航服务
│   └─ storage/        # 本地存储 (Hive + sqflite)
├─ features/          # 功能模块 (BLoC分层)
│   ├── auth/          # 登录/登出
│   │   ├── bloc/      # LoginBloc, AuthState, AuthEvent
│   │   ├── repository/ # AuthRepository
│   │   └─ pages/      # LoginPage, SplashPage
│   ├── homework/      # 作业功能
│   │   ├── bloc/      # HomeworkBloc
│   │   ├── repository/ # HomeworkRepository
│   │   └─ pages/      # HomeworkListPage, HomeworkDetailPage
│   ├── classroom/     # 课堂功能
│   │   ├── bloc/      # ClassroomBloc
│   │   └─ pages/      # JoinClassroomPage, ClassroomPage
│   ├── report/        # 学情报告
│   │   └─ pages/      # ReportPage, StudentReportPage
│   ├── checkin/       # 打卡功能
│   │   └─ pages/      # CheckinPage, CheckinHistoryPage
│   ├── message/       # 消息中心
│   │   └─ pages/      # MessageListPage, MessageDetailPage
└─ shared/             # 通用组件
    ├── widgets/       # InkCanvas, ScoreChart, AvatarWidget...
    └─ utils/          # 工具函数、常量定义
```

本文档版权归深圳自然写科技有限公司所有，技术细节仅用于软件著作权登记鉴别，请勿用于其他商业用途。

附录G 补充技术规格

G.1 iOS端Swift实现

G.1.1 CoreBluetooth智能笔连接

```
// PenBLEManager.swift
import CoreBluetooth

class PenBLEManager: NSObject, CBCentralManagerDelegate, CBPeripheralDelegate {
    private var centralManager: CBCentralManager!
    private var connectedPen: CBPeripheral?
    private var inkCharacteristic: CBCharacteristic?

    // WrittechPen GATT服务UUID
    private let SERVICE_UUID = CBUUID(string: "12345678-1234-5678-1234-56789ABCDEF0")
    private let INK_CHAR_UUID = CBUUID(string: "12345678-1234-5678-1234-56789ABCDEF1")

    var onInkData: (([InkPoint]) -> Void)?
    var onConnectionChanged: ((Bool) -> Void)?

    override init() {
        super.init()
        centralManager = CBCentralManager(delegate: self, queue: .main)
    }

    func startScan() {
        guard centralManager.state == .poweredOn else { return }
        centralManager.scanForPeripherals(
            withServices: [SERVICE_UUID],
            options: [CBCentralManagerScanOptionAllowDuplicatesKey: false]
        )
    }

    func centralManagerDidUpdateState(_ central: CBCentralManager) {
        if central.state == .poweredOn { startScan() }
    }

    func centralManager(_ central: CBCentralManager,
                        didDiscover peripheral: CBPeripheral,
                        advertisementData: [String: Any], rssi RSSI: NSNumber) {
        guard let name = peripheral.name, name.hasPrefix("WrittechPen") else { return }
        central.stopScan()
        connectedPen = peripheral
        connectedPen?.delegate = self
        central.connect(peripheral, options: nil)
    }

    func centralManager(_ central: CBCentralManager,
                        didConnect peripheral: CBPeripheral) {
        peripheral.discoverServices([SERVICE_UUID])
        onConnectionChanged?(true)
    }

    func peripheral(_ peripheral: CBPeripheral,
                    didDiscoverServices error: Error?) {
        peripheral.services?.first?.let { service in
```

```

        peripheral.discoverCharacteristics([INK_CHAR_UUID], for: service)
    }
}

func peripheral(_ peripheral: CBPeripheral,
               didDiscoverCharacteristicsFor service: CBService, error: Error?) {
    if let char = service.characteristics?.first(where: { $0.uuid == INK_CHAR_UUID }) {
        inkCharacteristic = char
        peripheral.setNotifyValue(true, for: char)
    }
}

func peripheral(_ peripheral: CBPeripheral,
               didUpdateValueFor characteristic: CBCharacteristic, error: Error?) {
    guard let data = characteristic.value else { return }
    let points = parseInkData(data)
    onInkData?(points)
}

private func parseInkData(_ data: Data) -> [InkPoint] {
    var points: [InkPoint] = []
    var offset = 0
    while offset + 10 <= data.count {
        let x = Float(UInt16(data[offset]) << 8 | UInt16(data[offset+1])) / 65535.0
        let y = Float(UInt16(data[offset+2]) << 8 | UInt16(data[offset+3])) / 65535.0
        let pressure = Float(data[offset+4]) / 255.0
        points.append(InkPoint(x: x, y: y, pressure: pressure))
        offset += 10
    }
    return points
}
}

```

G.2 推送通知实现

```

// NotificationManager.swift
import UserNotifications

class NotificationManager {
    func requestPermission() async -> Bool {
        let center = UNUserNotificationCenter.current()
        do {
            return try await center.requestAuthorization(
                options: [.alert, .badge, .sound]
            )
        } catch {
            return false
        }
    }

    func scheduleHomeworkReminder(homework: Homework) {
        let content = UNMutableNotificationContent()
        content.title = "作业提醒"
        content.body = "《\((homework.title)》 截止时间: \((homework.deadline.formatted())"
        content.sound = .default
        content.badge = 1

        // 截止前2小时提醒
        let triggerDate = homework.deadline.addingTimeInterval(-7200)
    }
}

```

```

        let components = Calendar.current.dateComponents(
            [.year, .month, .day, .hour, .minute], from: triggerDate)
        let trigger = UNCalendarNotificationTrigger(
            dateMatching: components, repeats: false)

        let request = UNNotificationRequest(
            identifier: "homework_\(homework.id)",
            content: content,
            trigger: trigger
        )

        UNUserNotificationCenter.current().add(request)
    }
}

```

G.3 Android端ViewModel架构

```

// HomeworkViewModel.kt
class HomeworkViewModel(
    private val homeworkRepo: HomeworkRepository
) : ViewModel() {

    private val _homeworkList = MutableStateFlow<List<Homework>>(emptyList())
    val homeworkList: StateFlow<List<Homework>> = _homeworkList.asStateFlow()

    private val _uiState = MutableStateFlow<UiState>(UiState.Idle)
    val uiState: StateFlow<UiState> = _uiState.asStateFlow()

    fun loadHomework(courseId: String) {
        viewModelScope.launch {
            _uiState.value = UiState.Loading
            try {
                homeworkRepo.getHomeworkList(courseId)
                    .collect { list ->
                        _homeworkList.value = list
                        _uiState.value = UiState.Success
                    }
            } catch (e: Exception) {
                _uiState.value = UiState.Error(e.message ?: "加载失败")
            }
        }
    }

    fun submitHomework(homeworkId: String, inkData: ByteArray) {
        viewModelScope.launch {
            _uiState.value = UiState.Uploading
            try {
                homeworkRepo.submitHomework(homeworkId, inkData)
                _uiState.value = UiState.Submitted
            } catch (e: Exception) {
                _uiState.value = UiState.Error(e.message ?: "提交失败")
            }
        }
    }

    sealed class UiState {
        object Idle : UiState()
        object Loading : UiState()
        object Uploading : UiState()
    }
}

```



```

        object Success : UiState()
        object Submitted : UiState()
        data class Error(val message: String) : UiState()
    }
}

```

附录H 补充技术规格

H.1 图片压缩上传

```

// ImageCompressUploader.kt
class ImageCompressUploader(private val apiService: ApiService) {

    companion object {
        const val MAX_SIZE_BYTES = 2 * 1024 * 1024 // 2MB
        const val INITIAL_QUALITY = 90
    }

    suspend fun compressAndUpload(
        uri: Uri,
        context: Context,
        targetUrl: String
    ): UploadResult = withContext(Dispatchers.IO) {
        val bitmap = BitmapFactory.decodeStream(
            context.contentResolver.openInputStream(uri))

        var quality = INITIAL_QUALITY
        var compressedBytes: ByteArray

        // 循环压缩直到文件大小≤2MB
        do {
            val baos = ByteArrayOutputStream()
            bitmap.compress(Bitmap.CompressFormat.JPEG, quality, baos)
            compressedBytes = baos.toByteArray()
            quality -= 10
        } while (compressedBytes.size > MAX_SIZE_BYTES && quality > 10)

        // 上传
        val requestBody = compressedBytes.toRequestBody("image/jpeg".toMediaType())
        val part = MultipartBody.Part.createFormData("file", "homework.jpg", requestBody)
        apiService.uploadImage(targetUrl, part)
    }
}

```

H.2 深色模式适配

```

// ThemeManager.kt
object ThemeManager {
    fun applyTheme(context: Context) {
        val nightMode = AppCompatDelegate.getDefaultNightMode()
        val sharedPrefs = PreferenceManager.getDefaultSharedPreferences(context)
        val setting = sharedPrefs.getString("theme", "system")
    }
}

```

```

        val mode = when (setting) {
            "light" -> AppCompatDelegate.MODE_NIGHT_NO
            "dark" -> AppCompatDelegate.MODE_NIGHT_YES
            else -> AppCompatDelegate.MODE_NIGHT_FOLLOW_SYSTEM
        }
        AppCompatDelegate.setDefaultNightMode(mode)
    }

    fun isDarkMode(context: Context): Boolean {
        return (context.resources.configuration.uiMode
            and Configuration.UI_MODE_NIGHT_MASK) == Configuration.UI_MODE_NIGHT_YES
    }
}

```

H.3 无障碍支持

```

// AccessibilityHelper.kt
object AccessibilityHelper {

    fun setupContentDescriptions(views: Map<View, String>) {
        views.forEach { (view, description) ->
            view.contentDescription = description
            // 对于自定义View额外设置accessibility delegate
            ViewCompat.setAccessibilityDelegate(view, object : AccessibilityDelegateCompat() {
                override fun onInitializeAccessibilityNodeInfo(
                    host: View, info: AccessibilityNodeInfoCompat) {
                    super.onInitializeAccessibilityNodeInfo(host, info)
                    info.contentDescription = description
                }
            })
        }
    }

    fun announceForAccessibility(view: View, message: String) {
        view.announceForAccessibility(message)
    }
}

```

本文档版权归深圳自然写科技有限公司所有，技术细节仅用于软件著作权登记鉴别，请勿用于其他商业用途。