

# 自然写互动课堂平板端应用软件 V1.0

## 鉴别材料

软件名称：自然写互动课堂平板端应用软件

版本号：V1.0

著作权人：深圳自然写科技有限公司

开发完成日期：2024年6月

文档类型：用户操作手册 + 设计说明书

## 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 各层次详细说明
  - 2.3 核心模块架构图
  - 2.4 数据设计
  - 2.5 接口设计
  - 2.6 安全设计
  - 2.7 部署架构
- 第三章 核心模块功能详细说明
  - 3.1 学生端作业作答模块
  - 3.2 教师端移动授课模块
  - 3.3 笔迹渲染模块
  - 3.4 蓝牙点阵笔连接模块
  - 3.5 字帖练习与笔顺指导模块
  - 3.6 错题本自动整理模块
  - 3.7 学习计划与进度管理模块
  - 3.8 护眼模式与使用时长管控模块
- 第四章 操作流程与使用步骤
  - 4.1 安装与首次设置
  - 4.2 登录与角色切换
  - 4.3 学生端操作流程
  - 4.4 教师端操作流程
  - 4.5 字帖练习操作流程

- 4.6 护眼模式与家长控制
- 4.7 异常处理与故障排除
- 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
- 5.2 核心功能类与方法说明
- 5.3 主要类命名规范
- 附录

# 第一章 软件整体概述

## 1.1 软件简介与功能综述

自然写互动课堂平板端应用软件（以下简称"Pad 端应用"）是自然写互动课堂教学系统的重要学习终端，面向学生和教师两类使用群体，运行于 Android 8.0+ 平板和 iPadOS 14.0+ 平板设备上。

Pad 端应用采用跨平台 Flutter 框架开发，与手机端应用共享核心业务逻辑模块，同时针对平板大屏幕提供专项优化的自适应布局。相比手机端，Pad 端具有更大的显示区域，能够更完整地展示字帖内容和书写作品，书写体验也更接近真实纸张。

功能综述一览：

角色	功能名称	功能描述
学生端	接收作业/试卷	从云平台下载教师布置的作业或试卷，支持离线模式
学生端	配合点阵笔纸上作答	通过 BLE 连接点阵笔，实时接收在点阵纸上的书写内容
学生端	触屏直接书写	无点阵笔时可直接在 Pad 屏幕上触控书写作答
学生端	查看批改结果	查看教师/AI 批改后的作业，含批注和错误分析
学生端	字帖练习	按字帖模板练习书写，实时笔顺指导和评分
学生端	错题本管理	自动整理历次作业错题，智能推荐复习
学生端	学习计划	查看并完成每日/每周学习任务
教师端	移动授课	在 Pad 上展示课件并随时批注
教师端	巡堂查看	实时查看全班学生书写进度
教师端	即时点评	对学生作品进行语音/文字即时点评
全体	护眼模式	色温调节、使用时长提醒、前置摄像头距离检测

## 1.2 软件用途与适用场景

主要用途：

Pad 端应用是自然写互动课堂系统在学生个人学习设备端的核心软件，支持课内互动作答和课后自主练习两大场景。学生通过 Pad 端应用完成作业提交、字帖练习、学情查看等日常学习任务；教师通过 Pad 端应用在移动状态下管理课堂、批阅作业、查看学情分析。

适用场景说明：

场景	参与角色	功能使用
课堂练习	学生	配合点阵笔在点阵纸上完成课堂练习，Pad 作为数据接收终端
课后作业	学生	下载作业，用点阵笔/触屏书写作答，提交后等待批改
字帖临摹	学生	在 Pad 上选择字帖，对照字帖用手指或点阵笔练习
错题复习	学生	查看错题本，有针对性地重新练习
巡堂监控	教师	教师手持 Pad 在教室内走动，实时查看各学生书写状态
即时批改	教师	在 Pad 上直接对学生作品进行批注和评分
家长监督	家长	通过家长控制功能设定学习时间，查看孩子学习报告

1.3 运行环境与系统要求

Android 平板硬件要求：

项目	最低要求	推荐配置
操作系统	Android 8.0 (API Level 26)	Android 11.0+
处理器	4核 ARM Cortex-A53 @ 1.5GHz	8核 ARM Cortex-A75 @ 2.0GHz+
内存	3GB RAM	6GB RAM
存储	32GB (可用 ≥ 8GB)	64GB
屏幕尺寸	8寸	10.5寸 ~ 11寸
屏幕分辨率	1280×800	2000×1200 (2K)
蓝牙	BLE 4.2	BLE 5.0
网络	Wi-Fi 802.11n (2.4GHz)	Wi-Fi 802.11ac (5GHz)
相机	前置摄像头 (护眼距离检测)	500万像素前置摄像头

iPadOS 硬件要求：

项目	最低要求	推荐配置
操作系统	iPadOS 14.0	iPadOS 16.0+
设备型号	iPad (第8代) 或更新	iPad Pro 11寸
存储	64GB	128GB
蓝牙	支持 BLE 5.0	支持 BLE 5.0

网络要求：

场景	要求
作业下载	Wi-Fi 10Mbps+ 或 4G LTE

场景	要求
作业上传（书写笔迹）	Wi-Fi 5Mbps+（典型作业 < 5MB）
课堂实时同步	Wi-Fi 5Mbps+，延迟 ≤ 100ms
离线模式	无网络，已下载作业可本地作答

### 1.4 开发语言与技术规范

语言/框架	版本	用途
Flutter	3.16.x	主框架，跨平台 UI 与业务逻辑
Dart	3.2.x	主要编程语言
Kotlin	1.9.x	Android 原生插件（BLE、护眼摄像头）
Swift	5.9.x	iOS 原生插件（CoreBluetooth、健康数据）
flutter_bloc	8.1.x	BLoC 状态管理
Dio	5.3.x	HTTP 网络请求
flutter_blue_plus	1.29.x	BLE 点阵笔连接
Hive	2.2.x	本地轻量级 NoSQL 存储（离线数据）
sqflite	2.3.x	SQLite 本地数据库
CustomPainter	Flutter内置	笔迹渲染（Skia 2D 引擎）
go_router	13.x	声明式路由导航
freezed	2.4.x	不可变数据类（BLoC 事件/状态）
json_serializable	6.7.x	JSON 序列化代码生成

**架构规范：** – 遵循 Flutter BLoC + MVVM 架构，与手机端代码共享 `lib/features/` 下的核心业务模块 – Pad 专用适配代码位于 `lib/adaptive/` 目录，通过屏幕宽度阈值（768dp）切换 Pad/Phone 布局 – 单元测试覆盖 BLoC 层，Widget 测试覆盖关键 UI 组件

### 1.5 版本说明

版本	日期	说明
V1.0.0	2024-06	正式版本发布（Android + iOS 双平台）
V0.9.0	2024-04	Beta：护眼模式、错题本功能完成
V0.7.0	2024-02	Alpha：Pad 自适应布局完成，与手机端代码分离

## 第二章 系统架构与设计思路

### 2.1 总体架构设计

Pad 端应用采用 Flutter 跨平台 MVVM 架构，与手机端共享 lib/features/ 下的核心业务逻辑（BLoC + Repository），通过平台适配层（lib/adaptive/）实现 Pad 专用的大屏幕布局。

整体架构分为八个层次：UI层（Pad自适应布局）→ 状态管理层（BLoC/Provider）→ 笔迹渲染层（CustomPainter + Skia）→ 业务逻辑层→ 数据层 → 网络层 → 蓝牙层 → 护眼层。



## 2.2 各层次详细说明

### 2.2.1 UI 层 (Pad 自适应布局)

Pad 端应用针对平板大屏幕（8寸~13寸）设计了专项的自适应布局策略：

- **双栏布局：**屏幕宽度  $\geq 768dp$  时启用双栏布局（左侧导航 + 右侧内容区），充分利用平板横屏空间
- **大字临摹布局：**字帖练习界面提供完整的参考字展示区（左半屏）+ 学生书写区（右半屏）对照布局
- **横竖屏自适应：**自动响应设备旋转，横屏双栏、竖屏单栏无缝切换

- 高分辨率优化：针对 2K 分辨率 Pad 提供高清笔迹渲染（2倍像素密度）

### 2.2.2 状态管理层 (BLoC)

采用 flutter\_bloc 库实现 BLoC 模式，每个主要业务场景对应一个 BLoC 类：

- HomeworkBloc：作业列表、作业详情、作答提交状态管理
- QuizBloc：课堂互动答题状态（发题/作答/查看结果）
- PracticeBloc：字帖练习状态（选帖/练习中/评分展示）
- TeacherBloc：教师端状态（巡堂模式/批改模式/授课模式）
- EyeProtectionBloc：护眼功能状态（色温/时长/距离检测）

### 2.2.3 笔迹渲染层

笔迹渲染基于 Flutter CustomPainter + Skia 2D 渲染引擎：

- InkCanvasPainter：实时渲染 BLE 笔迹和触屏笔迹，贝塞尔曲线平滑
- StrokeReplayPainter：书写回放动画，按时间序列重现学生书写过程
- CalligraphyPainter：字帖模板渲染（笔顺序号、参考笔画、红色描红线）
- AnnotationPainter：教师批注渲染（叠加在学生作品上方的批注层）

### 2.2.4 业务逻辑层

业务逻辑层通过 Repository 模式隔离数据来源，核心业务与手机端共享代码：

- HomeworkRepository：作业数据管理（下载、缓存、提交、批改结果查询）
- PracticeRepository：练字数据管理（字帖获取、练习记录、评分历史）
- MistakeRepository：错题本数据管理（自动整理、分类、推荐复习）
- LearningPlanRepository：学习计划数据管理（任务生成、进度跟踪）
- EyeProtectionRepository：护眼数据管理（使用时长记录、家长设置同步）

### 2.2.5 数据层

本地数据存储采用双存储引擎策略：

- sqflite (SQLite)：存储结构化数据（作业记录、笔迹数据、错题本、学习进度）
- Hive：存储轻量级键值数据（用户配置、离线操作队列、应用缓存）
- 文件系统：存储大文件（下载的字帖模板图片、教师批改后的作业快照）

### 2.2.6 蓝牙层 (PenBleManager)

BLE 点阵笔连接管理，基于 flutter\_blue\_plus 插件：

- 扫描周围 BLE 设备，过滤自然写点阵笔（UUID 匹配）
- 建立 GATT 连接，订阅笔迹数据 Characteristic (Notify)
- 接收原始 BLE 数据包，解析为 InkPoint (x, y, pressure, timestamp)
- 断线自动重连策略（保存已配对笔的 MAC 地址，重启自动重连）

### 2.2.7 护眼层

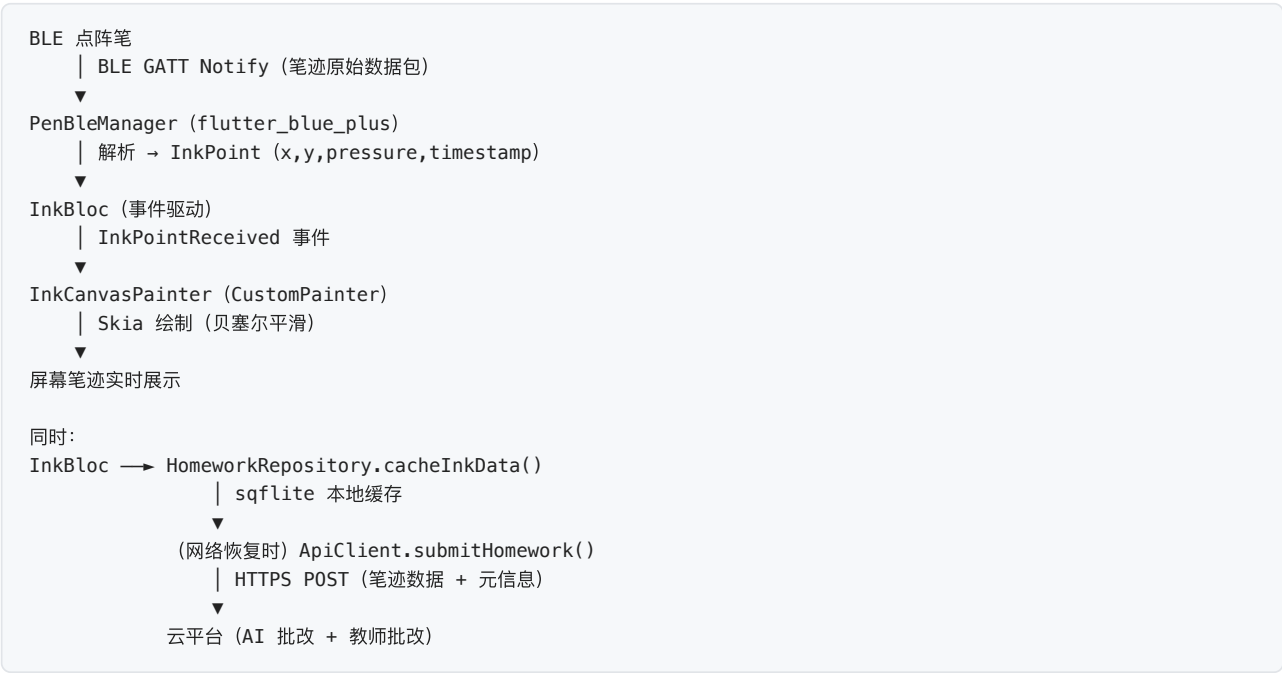
护眼功能由三个独立子模块组成：

- **色温调节**：通过 Android Display API / iOS UIScreen 调节屏幕色温（暖色护眼模式）
- **使用时长提醒**：记录每次使用时长，达到设定阈值时弹出休息提醒（家长可远程设置）
- **距离检测**：调用前置摄像头（仅 Android，本地 ML 处理），检测用眼距离，过近时发出警告

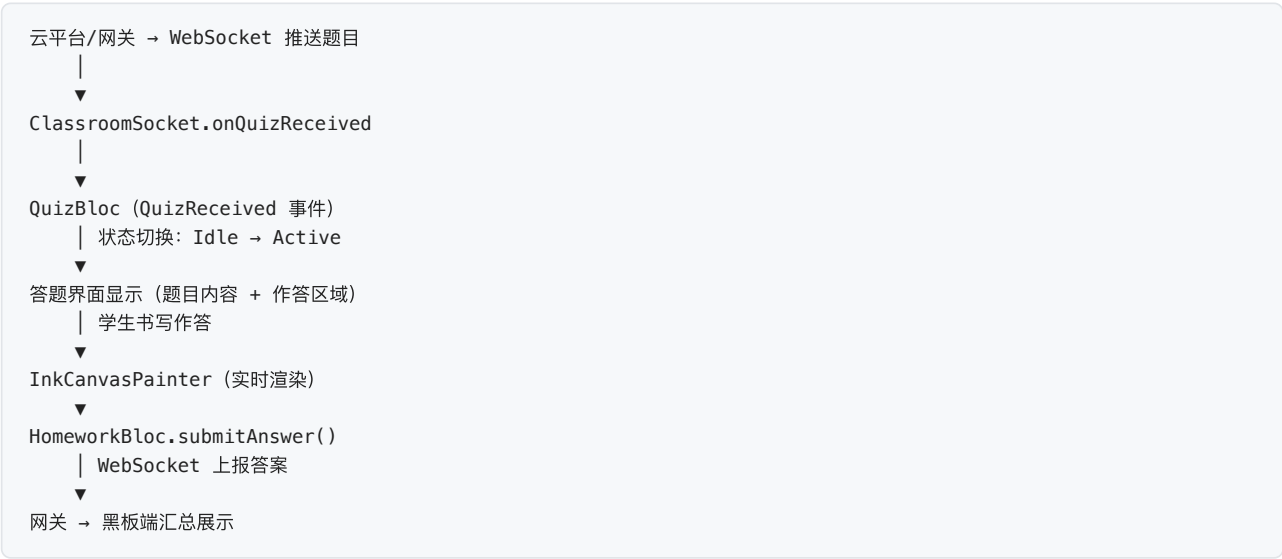
距离检测特别说明：前置摄像头仅在护眼检测功能开启时调用，图像数据仅在本地处理（ML Kit 人脸检测），不上传到服务器，保护学生隐私。

## 2.3 核心模块架构图

Pad 端数据流图：



学生课堂答题数据流：



## 2.4 数据设计

### 2.4.1 数据库表结构 (sqlite)

homework 表（作业记录）

字段名	数据类型	说明
id	TEXT PRIMARY KEY	作业ID（云平台全局唯一）
title	TEXT NOT NULL	作业标题
subject	TEXT	科目（语文/数学/英语等）
type	INTEGER	类型（1=练字 2=作文 3=计算 4=综合）
content_json	TEXT	作业内容（JSON，含题目和字帖）
due_at	INTEGER	截止时间戳
status	INTEGER	状态（0=未开始 1=进行中 2=已提交 3=已批改）
score	REAL	得分（-1=未批改）
feedback_json	TEXT	批改反馈 JSON
created_at	INTEGER	下发时间戳
updated_at	INTEGER	最后更新时间戳

ink\_data 表（笔迹数据）

字段名	数据类型	说明
id	INTEGER PRIMARY KEY	自增主键
homework_id	TEXT NOT NULL	所属作业ID
page_index	INTEGER	页码
ink_points	BLOB	笔迹点序列（压缩后的二进制数据）
stroke_count	INTEGER	笔画数
created_at	INTEGER	记录时间戳
is_uploaded	INTEGER	是否已上传（0=否 1=是）

mistake\_book 表（错题本）

字段名	数据类型	说明
id	INTEGER PRIMARY KEY	自增主键
homework_id	TEXT	来源作业ID
character	TEXT	错误汉字/题目内容
wrong_ink	BLOB	错误书写笔迹快照（PNG 压缩）
correct_ink	BLOB	正确示范笔迹快照
error_type	TEXT	错误类型（笔画错误/结构错误/笔顺错误/字形偏差）
knowledge_point	TEXT	知识点标签



字段名	数据类型	说明
review_count	INTEGER	已复习次数
next_review_at	INTEGER	下次复习时间戳（Leitner 间隔复习算法）
created_at	INTEGER	加入时间戳

study\_plan 表（学习计划）

字段名	数据类型	说明
id	INTEGER PRIMARY KEY	自增主键
plan_date	TEXT	计划日期（YYYY-MM-DD）
task_type	INTEGER	任务类型（1=作业 2=练字 3=错题复习 4=自由练习）
task_id	TEXT	关联任务ID
duration_min	INTEGER	计划时长（分钟）
is_completed	INTEGER	是否完成（0=否 1=是）
completed_at	INTEGER	完成时间戳

usage\_log 表（使用时长记录）

字段名	数据类型	说明
id	INTEGER PRIMARY KEY	自增主键
log_date	TEXT	记录日期（YYYY-MM-DD）
session_start	INTEGER	本次使用开始时间戳
session_end	INTEGER	本次使用结束时间戳
duration_sec	INTEGER	本次使用时长（秒）
activity	TEXT	活动类型（作业/练字/查看报告等）

2.4.2 Hive 存储结构

```
// Hive Box 定义
class HiveBoxes {
    static const userPrefs = 'user_preferences'; // 用户偏好设置
    static const offlineQueue = 'offline_queue'; // 离线操作队列
    static const appCache = 'app_cache'; // 应用通用缓存
    static const eyeProtection = 'eye_protection'; // 护眼设置
}

// 用户偏好（存储在 user_preferences Box）
@HiveType(typeId: 1)
class UserPreferences extends HiveObject {
    @HiveField(0) String themeMode; // light / dark / system
    @HiveField(1) String inkColor; // 默认笔色
    @HiveField(2) double inkWidth; // 默认笔粗
    @HiveField(3) bool autoConnectPen; // 自动连接点阵笔
    @HiveField(4) String lastPenMac; // 上次连接的笔 MAC
```

```
@HiveField(5) bool eyeProtectEnabled; // 护眼模式开关
}
```

### 2.4.3 核心数据结构定义

```
// 笔迹点
class InkPoint {
    final double x;           // 归一化坐标 [0.0, 1.0]
    final double y;           // 归一化坐标 [0.0, 1.0]
    final double pressure;     // 压感值 [0.0, 1.0] (触屏为 0.5 固定值)
    final int timestamp;       // 微秒时间戳
    final bool isPenUp;        // 是否抬笔 (笔画结束标记)

    const InkPoint({
        required this.x, required this.y,
        required this.pressure, required this.timestamp,
        this.isPenUp = false,
    });
}

// 作业
class Homework {
    final String id;
    final String title;
    final String subject;
    final HomeworkType type;
    final List<HomeworkPage> pages; // 多页作业内容
    final DateTime dueAt;
    HomeworkStatus status;
    double? score;                  // AI 批改分数
    List<TeacherAnnotation>? annotations; // 教师批注
}

// 字帖模板
class CalligraphyTemplate {
    final String id;
    final String character; // 练习汉字
    final int strokeCount;  // 笔画数
    final List<Stroke> strokes; // 标准笔顺笔画 (坐标序列)
    final Uint8List? referenceImage; // 参考图片 (高清毛笔字/楷书)
    final List<String> strokeNames; // 各笔画名称 (横/竖/撇/捺/折等)
}
```

## 2.5 接口设计

### 2.5.1 云平台 API 接口

接口名称	方法	路径	说明
登录 (学生/教师)	POST	/api/v1/auth/login	账号密码登录, 返回 JWT Token
刷新 Token	POST	/api/v1/auth/refresh	刷新过期的 Token
获取作业列表	GET	/api/v1/homework/list	获取当前学生的作业列表
下载作业内容	GET	/api/v1/homework/{id}/content	下载作业详情 (题目、字帖内容)
提交作业	POST	/api/v1/homework/{id}/submit	上传学生作答笔迹数据
获取批改结果	GET	/api/v1/homework/{id}/result	获取 AI/教师批改结果

接口名称	方法	路径	说明
获取字帖模板	GET	/api/v1/calligraphy/templates	获取字帖模板列表
下载字帖	GET	/api/v1/calligraphy/{id}	下载字帖详情（笔顺数据+参考图）
上传练习记录	POST	/api/v1/calligraphy/practice	上传练字记录（评分数据）
获取错题列表	GET	/api/v1/mistakes/list	获取学生错题本
获取学情报告	GET	/api/v1/report/student/{id}	获取学生学情分析报告
获取学习计划	GET	/api/v1/plan/current	获取当前学习计划任务
同步使用时长	POST	/api/v1/usage/sync	上传使用时长（家长监控）

网络请求头统一规范：

```
class ApiInterceptor extends Interceptor {
    @override
    void onRequest(RequestOptions options, RequestInterceptorHandler handler) {
        options.headers.addAll({
            'Authorization': 'Bearer ${AuthManager.token}',
            'X-Platform': Platform.isAndroid ? 'android-pad' : 'ios-pad',
            'X-App-Version': AppConfig.version,
            'X-Device-Id': DeviceInfo.deviceId,
            'Content-Type': 'application/json',
        });
        handler.next(options);
    }

    @override
    void onError(DioException err, ErrorInterceptorHandler handler) {
        if (err.response?.statusCode == 401) {
            // Token 过期，触发自动刷新
            AuthManager.refreshToken().then((_) => retry(err.requestOptions));
        }
        handler.next(err);
    }
}
```

2.5.2 BLE 点阵笔接口

```
class PenBleManager {
    // 扫描周围点阵笔（过滤条件：服务 UUID 匹配）
    Stream<BluetoothDevice> scanPens({Duration timeout = const Duration(seconds: 10)})

    // 连接指定点阵笔
    Future<void> connectPen(BluetoothDevice device)

    // 断开连接
    Future<void> disconnectPen()

    // 笔迹数据流（持续订阅 GATT Notify Characteristic）
    Stream<List<InkPoint>> get inkDataStream

    // 当前连接状态流
    Stream<PenConnectionState> get connectionStateStream

    // 获取电量（读取 Battery Level Characteristic）
    Future<int> getBatteryLevel()
}
```

```
// 自然写点阵笔 GATT 服务定义
class WritechPenGatt {
    static const serviceUuid = '6e400001-b5a3-f393-e0a9-e50e24dcca9e';
    static const inkCharUuid = '6e400003-b5a3-f393-e0a9-e50e24dcca9e'; // Notify
    static const cmdCharUuid = '6e400002-b5a3-f393-e0a9-e50e24dcca9e'; // Write
    static const battCharUuid = '00002a19-0000-1000-8000-00805f9b34fb'; // Battery
}
```

### 2.5.3 课堂实时 WebSocket 接口

```
class ClassroomSocket {
    // 连接课堂 WebSocket
    Future<void> connect(String sessionId)

    // 课堂互动事件流（发题/收卷/暂停等）
    Stream<ClassroomEvent> get classroomEventStream

    // 发送答案
    Future<void> submitAnswer(String quizId, dynamic answer)

    // 发送实时笔迹（学生作答时实时同步到黑板）
    void sendInkFrame(InkFrame frame)
}
```

## 2.6 安全设计

### 账户安全：

```
class SecureAuthStorage {
    final FlutterSecureStorage _secureStorage = const FlutterSecureStorage(
        aOptions: AndroidOptions(encryptedSharedPreferences: true),
        iOptions: IOSOptions(accessibility: KeychainAccessibility.first_unlock),
    );

    // Token 存储到系统安全区域 (Android Keystore / iOS Keychain)
    Future<void> saveToken(String token) =>
        _secureStorage.write(key: 'auth_token', value: token);

    Future<String?> readToken() =>
        _secureStorage.read(key: 'auth_token');

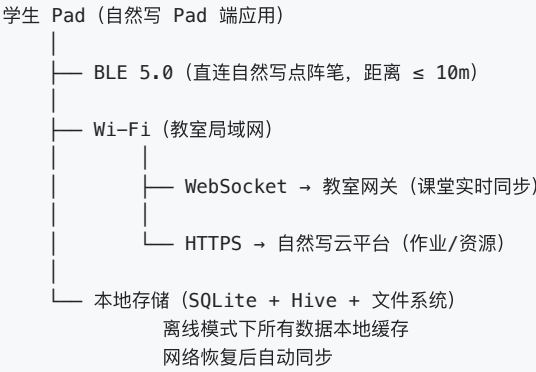
    Future<void> clearToken() =>
        _secureStorage.delete(key: 'auth_token');
}
```

**数据安全：** – 学生笔迹数据本地存储采用 AES-256 加密（通过 SQLCipher for Flutter 插件） – 作业内容下载后以加密形式缓存，防止设备丢失后题目泄露 – 网络传输强制 HTTPS (TLS 1.2+)，证书绑定（Certificate Pinning）防止中间人攻击

**隐私保护：** – 护眼距离检测前置摄像头图像数据仅在本地处理，使用 Google ML Kit 人脸检测 API – 图像数据不持久化存储，仅在内存中临时使用 – 隐私政策明确告知：摄像头仅用于护眼距离检测，不用于身份识别

**使用管控：** – 家长可通过家长端（手机APP）远程设置： – 每日使用时长上限（0~8小时） – 允许使用时段（如仅允许 17:00~21:00） – 强制休息提醒间隔（如每45分钟休息10分钟） – 时长控制通过服务端下发配置，本地应用守规执行

## 2.7 部署架构



### 第三章 核心模块功能详细说明

#### 3.1 学生端作业作答模块

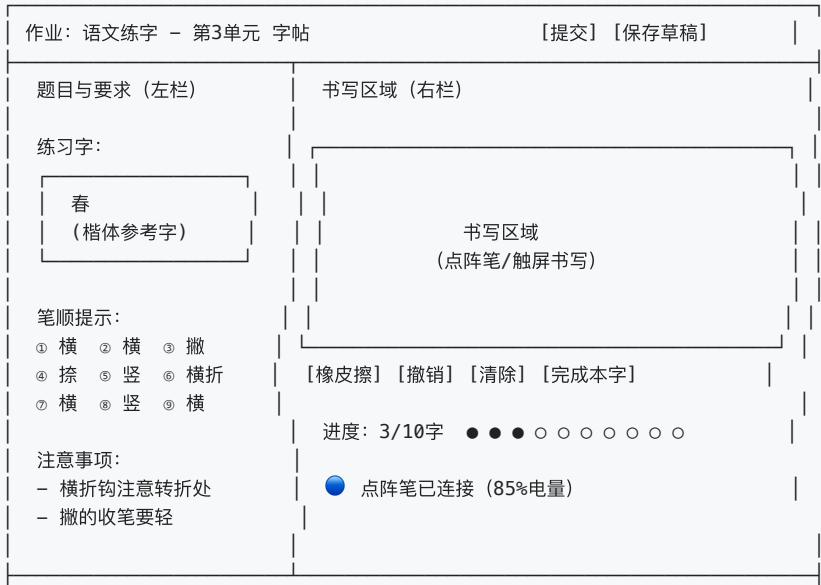
##### 3.1.1 模块功能描述

学生端作业作答模块是 Pad 端应用最核心的功能，支持学生接收教师布置的作业，通过点阵笔纸上书写（BLE 传输）或直接触屏书写完成作答，并提交给云平台进行 AI/教师批改。

作答模式说明：

模式	条件	描述
点阵笔纸上书写	已连接点阵笔 + 有点阵纸	学生在点阵纸上用点阵笔书写，Pad 实时显示笔迹
触屏直接书写	无点阵笔时	学生直接用手指在 Pad 屏幕上书写作答
离线模式	无网络时	先本地缓存作答数据，网络恢复后自动提交

##### 3.1.2 作业作答界面布局（Pad 横屏双栏）



### 3.1.3 作业作答流程

```

class HomeworkBloc extends Bloc<HomeworkEvent, HomeworkState> {

  HomeworkBloc({
    required HomeworkRepository homeworkRepo,
    required PenBleManager penBleManager,
  }) : super(const HomeworkState.initial()) {

    // 加载作业内容
    on<LoadHomework>((event, emit) async {
      emit(const HomeworkState.loading());
      try {
        final homework = await homeworkRepo.getHomeworkById(event.homeworkId);
        final inkData = await homeworkRepo.getCachedInkData(event.homeworkId);
        emit(HomeworkState.loaded(homework: homework, inkData: inkData));
      } catch (e) {
        emit(HomeworkState.error(message: e.toString()));
      }
    });

    // 接收笔迹点 (来自点阵笔 BLE 或触屏)
    on<InkPointReceived>((event, emit) {
      final current = state as HomeworkStateLoaded;
      final updatedInk = current.currentPageInk.copyWith(
        points: [...current.currentPageInk.points, event.point],
      );
      // 本地缓存笔迹 (每100个点保存一次)
      if (updatedInk.points.length % 100 == 0) {
        homeworkRepo.cacheInkData(current.homework.id, current.pageIndex, updatedInk);
      }
      emit(current.copyWith(currentPageInk: updatedInk));
    });

    // 提交作业
    on<SubmitHomework>((event, emit) async {
      final current = state as HomeworkStateLoaded;
      emit(current.copyWith(isSubmitting: true));
      try {
        await homeworkRepo.submitHomework(
          homeworkId: current.homework.id,
          inkPages: current.allPagesInk,
        );
        emit(current.copyWith(isSubmitting: false, isSubmitted: true));
      } on NetworkException {
        // 网络异常: 加入离线队列
        await homeworkRepo.addToOfflineQueue(current.homework.id, current.allPagesInk);
        emit(current.copyWith(isSubmitting: false, isOfflineQueued: true));
      }
    });
  }
}

```

### 3.1.4 离线同步机制

```

class OfflineSyncService {
  final HomeworkRepository homeworkRepo;
  final ApiClient apiClient;

  // 监听网络恢复, 自动同步离线队列
}

```

```

void startMonitoring() {
    Connectivity().onConnectivityChanged.listen((result) async {
        if (result != ConnectivityResult.none) {
            await _syncOfflineQueue();
        }
    });
}

Future<void> _syncOfflineQueue() async {
    final queue = await homeworkRepo.getOfflineQueue();
    for (final item in queue) {
        try {
            await apiClient.submitHomework(
                homeworkId: item.homeworkId,
                inkData: item.inkData,
            );
            await homeworkRepo.removeFromOfflineQueue(item.id);
        } catch (e) {
            // 单条失败不影响其他条目，继续处理
            continue;
        }
    }
}
}
}

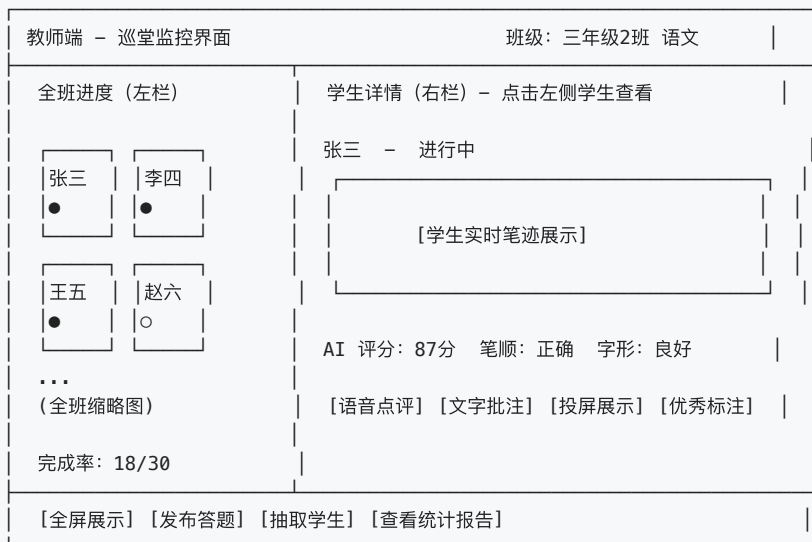
```

## 3.2 教师端移动授课模块

### 3.2.1 模块功能描述

教师端移动授课模块为教师提供在 Pad 上进行课堂教学管理的功能，包括移动查看全班学生书写进度、对学生作品进行即时点评和批注，以及在 Pad 上展示课件并批注。

### 3.2.2 教师端巡堂界面



### 3.2.3 即时点评功能实现

```

class AnnotationPainter extends CustomPainter {
    final List<Stroke> studentStrokes;    // 学生笔迹

```

```

final List<TeacherAnnotation> annotations; // 教师批注（叠加层）

@override
void paint(Canvas canvas, Size size) {
  // 第一层：渲染学生笔迹（较淡，作为底图）
  _drawStudentStrokes(canvas, size);

  // 第二层：渲染教师批注（鲜明颜色叠加）
  for (final annotation in annotations) {
    switch (annotation.type) {
      case AnnotationType.ink:
        _drawAnnotationStrokes(canvas, size, annotation.strokes, annotation.color);
      case AnnotationType.circle:
        _drawCircle(canvas, size, annotation.rect, annotation.color);
      case AnnotationType.arrow:
        _drawArrow(canvas, size, annotation.start, annotation.end, annotation.color);
      case AnnotationType.text:
        _drawTextLabel(canvas, size, annotation.text, annotation.position);
    }
  }
}

```

### 3.3 笔迹渲染模块

#### 3.3.1 模块功能描述

笔迹渲染模块基于 Flutter CustomPainter + Dart Skia 2D 引擎，提供高性能的笔迹实时渲染和书写回放能力，支持压力感宽度变化（BLE 笔迹）和触屏模拟压感。

#### 3.3.2 实时渲染实现

```

class InkCanvasPainter extends CustomPainter {
  final List<Stroke> strokes; // 已完成的笔画
  final List<InkPoint> current; // 当前正在书写的笔画点序列

  static final Paint _inkPaint = Paint()
    ..style = PaintingStyle.stroke
    ..strokeCap = StrokeCap.round
    ..strokeJoin = StrokeJoin.round
    ..isAntiAlias = true;

  @override
  void paint(Canvas canvas, Size size) {
    // 渲染历史笔画
    for (final stroke in strokes) {
      _drawStroke(canvas, size, stroke);
    }
    // 渲染当前笔画
    if (current.isNotEmpty) {
      _drawCurrentStroke(canvas, size, current);
    }
  }

  void _drawStroke(Canvas canvas, Size size, Stroke stroke) {
    if (stroke.points.length < 2) return;

    final path = Path();
    path.moveTo(
      stroke.points.first.x * size.width,
      stroke.points.first.y * size.height,

```



```

);

for (int i = 1; i < stroke.points.length - 1; i++) {
    final p = stroke.points[i];
    final pNext = stroke.points[i + 1];

    // 贝塞尔曲线平滑: 中点作为终点, 当前点作为控制点
    final midX = (p.x + pNext.x) / 2.0 * size.width;
    final midY = (p.y + pNext.y) / 2.0 * size.height;

    path.quadraticBezierTo(
        p.x * size.width, p.y * size.height,
        midX, midY,
    );

    // 压感宽度变化 (BLE 笔迹有真实压感, 触屏用速度模拟)
    _inkPaint.strokeWidth = _calcWidth(p.pressure, stroke.baseWidth);
}

_inkPaint.color = Color(stroke.colorArgb);
canvas.drawPath(path, _inkPaint);
}

double _calcWidth(double pressure, double baseWidth) {
    // 压感曲线: 小压感偏细, 大压感偏粗 (非线性映射)
    final normalized = pressure.clamp(0.0, 1.0);
    return baseWidth * (0.4 + 0.6 * normalized);
}

@Override
bool shouldRepaint(InkCanvasPainter oldDelegate) {
    return oldDelegate.strokes != strokes || oldDelegate.current != current;
}
}

```

### 3.3.3 书写回放实现

```

class StrokeReplayController {
    final List<Stroke> strokes;
    final double replaySpeed; // 1.0x 正常速度, 2.0x 两倍速

    Timer? _timer;
    int _strokeIndex = 0;
    int _pointIndex = 0;
    final List<Stroke> _visibleStrokes = [];
    List<InkPoint> _currentPoints = [];

    void Function(List<Stroke>, List<InkPoint>)? onFrameUpdate;

    void startReplay() {
        final intervalMs = (16 / replaySpeed).round(); // ~60fps
        _timer = Timer.periodic(Duration(milliseconds: intervalMs), _tick);
    }

    void _tick(Timer timer) {
        if (_strokeIndex >= strokes.length) {
            timer.cancel(); // 回放完成
            return;
        }

        final currentStroke = strokes[_strokeIndex];
        if (_pointIndex < currentStroke.points.length) {
            // 逐点添加, 模拟书写过程
            _currentPoints = currentStroke.points.sublist(0, _pointIndex + 1);
            _pointIndex++;
        }
    }
}

```

```

    } else {
      // 当前笔画结束，加入历史，开始下一笔
      _visibleStrokes.add(currentStroke);
      _currentPoints = [];
      _strokeIndex++;
      _pointIndex = 0;
    }

    onFrameUpdate?.call(List.unmodifiable(_visibleStrokes), List.unmodifiable(_currentPoints));
  }
}

```

## 3.4 蓝牙点阵笔连接模块

### 3.4.1 模块功能描述

蓝牙点阵笔连接模块负责管理 Pad 与自然写点阵笔之间的 BLE 连接，包括扫描发现、连接管理、笔迹数据接收和断线自动重连。

### 3.4.2 BLE 连接管理实现

```

class PenBleManager {
  late FlutterBluePlus _ble;
  BluetoothDevice? _connectedDevice;
  BluetoothCharacteristic? _inkCharacteristic;
  final _inkStreamController = StreamController<List<InkPoint>>.broadcast();
  final _connectionStateController = StreamController<PenConnectionState>.broadcast();

  Stream<List<InkPoint>> get inkDataStream => _inkStreamController.stream;
  Stream<PenConnectionState> get connectionStateStream => _connectionStateController.stream;

  // 扫描自然写点阵笔
  Stream<BluetoothDevice> scanPens() {
    FlutterBluePlus.startScan(
      withServices: [Guid(WritechPenGatt.serviceUuid)],
      timeout: const Duration(seconds: 15),
    );
    return FlutterBluePlus.scanResults.map(
      (results) => results.map((r) => r.device)
    ).expand((devices) => devices);
  }

  // 连接点阵笔
  Future<void> connectPen(BluetoothDevice device) async {
    await device.connect(timeout: const Duration(seconds: 10));
    _connectedDevice = device;

    // 发现服务和特征
    final services = await device.discoverServices();
    final penService = services.firstWhere(
      (s) => s.uuid == Guid(WritechPenGatt.serviceUuid),
    );

    _inkCharacteristic = penService.characteristics.firstWhere(
      (c) => c.uuid == Guid(WritechPenGatt.inkCharUuid),
    );

    // 订阅笔迹 Notify
    await _inkCharacteristic!.setNotifyValue(true);
    _inkCharacteristic!.lastValueStream.listen(_onInkDataReceived);
  }
}

```

```
// 监听连接状态断线
device.connectionState.listen((state) {
  if (state == BluetoothConnectionState.disconnected) {
    _connectionStateController.add(PenConnectionState.disconnected);
    _autoReconnect(device); // 启动自动重连
  }
});

_connectionStateController.add(PenConnectionState.connected);
}

// 解析 BLE 原始字节为笔迹点列表
void _onInkDataReceived(List<int> bytes) {
  final points = <InkPoint>[];
  // 每个笔迹点格式: x(2B) + y(2B) + pressure(1B) + timestamp(4B) + flag(1B) = 10 字节
  for (int offset = 0; offset + 10 <= bytes.length; offset += 10) {
    final x = ((bytes[offset] << 8) | bytes[offset + 1]).toDouble() / 65535.0;
    final y = ((bytes[offset + 2] << 8) | bytes[offset + 3]).toDouble() / 65535.0;
    final pressure = bytes[offset + 4].toDouble() / 255.0;
    final timestamp = (bytes[offset + 5] << 24) | (bytes[offset + 6] << 16) |
      (bytes[offset + 7] << 8) | bytes[offset + 8];
    final isPenUp = (bytes[offset + 9] & 0x01) != 0;

    points.add(InkPoint(x: x, y: y, pressure: pressure,
      timestamp: timestamp, isPenUp: isPenUp));
  }
  if (points.isNotEmpty) {
    _inkStreamController.add(points);
  }
}

// 自动重连 (指数退避: 1s, 2s, 4s, 8s...最大30s)
void _autoReconnect(BluetoothDevice device) async {
  int delaySeconds = 1;
  while (_connectedDevice == device) {
    await Future.delayed(Duration(seconds: delaySeconds));
    try {
      await connectPen(device);
      return; // 重连成功
    } catch (_) {
      delaySeconds = (delaySeconds * 2).clamp(1, 30);
    }
  }
}
}
```

## 3.5 字帖练习与笔顺指导模块

### 3.5.1 模块功能描述

字帖练习与笔顺指导模块为学生提供系统化的书法练习功能，支持楷书、行书、硬笔等多种字帖，并通过实时笔顺检测和评分给出练习指导。

### 3.5.2 字帖练习界面布局

字帖练习 - 人教版三年级上册 第1单元		[评分历史] [×]	
参考字区域 (左半屏)		学生书写区域 (右半屏)	

<p>春 (楷体参考字 - 大字展示)</p> <p>笔顺: ①②③④⑤⑥⑦⑧⑨</p>	<p>(学生书写区域)</p>
<p>当前笔画: 第 ④ 笔 - 捺</p>	<p>已写 ④ 笔 ✓正确 笔顺得分: 100分</p>
<p>笔画动画演示 (当前笔画高亮动画)</p>	<p>[橡皮擦] [清除重写] [下一字 →]</p>
<p>[重播动画] [笔顺说明]</p>	<p>总体评分: ★★★★★☆ (87分)</p> <p>评分反馈: ✓ 笔顺正确 △ 第③笔撇的收笔偏重 △ 横折折钩转折角度偏大</p>

### 3.5.3 笔顺检测算法

```
class StrokeOrderChecker {
    final CalligraphyTemplate template;

    // 检测当前书写笔画是否符合标准笔顺
    StrokeOrderResult checkStroke(int currentStrokeIndex, Stroke writtenStroke) {
        if (currentStrokeIndex >= template.strokes.length) {
            return StrokeOrderResult.extraStroke;
        }

        final expectedStroke = template.strokes[currentStrokeIndex];

        // 起点方向检测 (判断是否从正确位置开始)
        final startMatch = _checkStartPoint(writtenStroke, expectedStroke);

        // 书写方向检测 (横/竖/撇/捺的方向向量匹配)
        final directionMatch = _checkDirection(writtenStroke, expectedStroke);

        // 终点位置检测
        final endMatch = _checkEndPoint(writtenStroke, expectedStroke);

        if (startMatch && directionMatch && endMatch) {
            return StrokeOrderResult.correct;
        } else if (!directionMatch) {
            return StrokeOrderResult.wrongDirection;
        } else {
            return StrokeOrderResult.positionError;
        }
    }

    // 计算书写评分 (综合笔顺、字形、比例)
    PracticeScore calcScore(List<Stroke> writtenStrokes) {
        double strokeOrderScore = _evalStrokeOrder(writtenStrokes); // 笔顺满分40分
        double shapeScore = _evalShape(writtenStrokes); // 字形满分35分
        double proportionScore = _evalProportion(writtenStrokes); // 比例满分25分

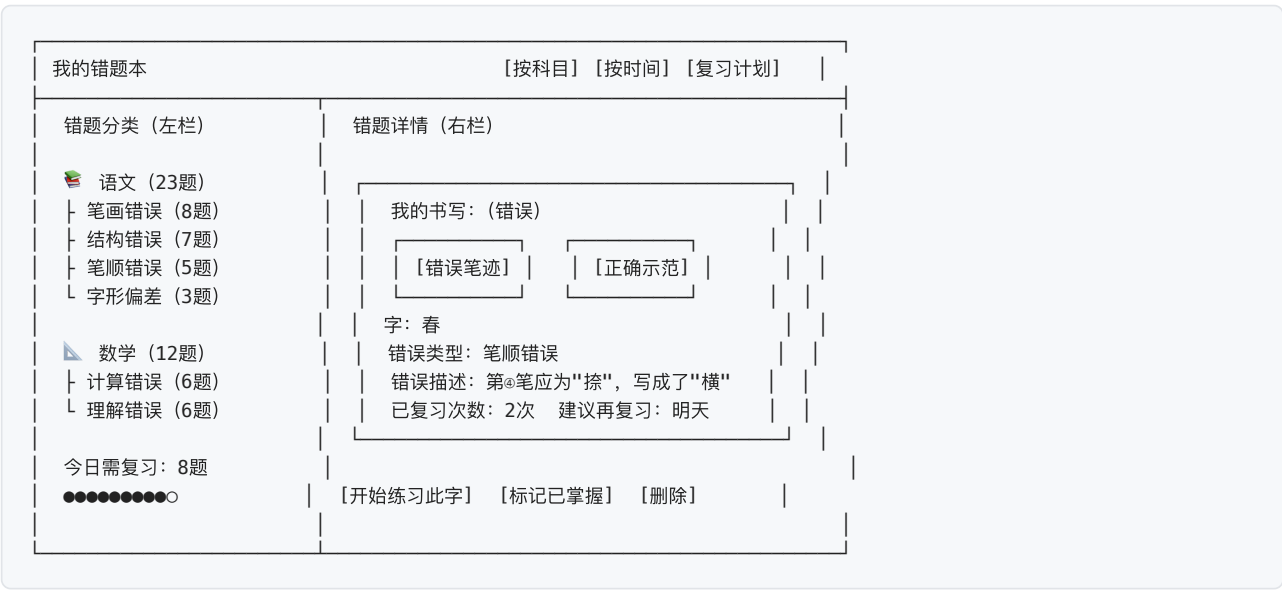
        return PracticeScore(
            total: strokeOrderScore + shapeScore + proportionScore,
            strokeOrder: strokeOrderScore,
            shape: shapeScore,
            proportion: proportionScore,
        );
    }
}
```

### 3.6 错题本自动整理模块

#### 3.6.1 模块功能描述

错题本自动整理模块从教师批改和 AI 批改结果中自动提取学生的错误题目，按错误类型分类整理，并通过 Leitner 间隔复习算法智能安排复习计划。

#### 3.6.2 错题本界面



#### 3.6.3 Leitner 间隔复习算法

```
class LeitnerScheduler {
    // Leitner 卡片盒子间隔天数: 盒子0=每天, 1=2天, 2=4天, 3=8天, 4=16天, 5=已掌握
    static const boxIntervals = [1, 2, 4, 8, 16, 999];

    // 复习后更新复习计划
    MistakeItem updateAfterReview(MistakeItem item, bool isCorrect) {
        int newBoxLevel;
        if (isCorrect) {
            // 答对: 升级到下一个盒子
            newBoxLevel = (item.leitnerLevel + 1).clamp(0, boxIntervals.length - 1);
        } else {
            // 答错: 退回到第0级 (明天重新复习)
            newBoxLevel = 0;
        }

        final nextReviewDate = DateTime.now().add(
            Duration(days: boxIntervals[newBoxLevel])
        );

        return item.copyWith(
            leitnerLevel: newBoxLevel,
            reviewCount: item.reviewCount + 1,
            nextReviewAt: nextReviewDate.millisecondsSinceEpoch,
        );
    }
}

// 获取今日需复习的错题列表
Future<List<MistakeItem>> getTodayReviewItems() async {
    final now = DateTime.now().millisecondsSinceEpoch;
```

```
final items = await mistakeRepository.getAllMistakes();
return items.where((item) =>
    item.leitnerLevel < boxIntervals.length - 1 && // 未达到"已掌握"级别
    item.nextReviewAt <= now // 到了复习时间
).toList();
}
```

### 3.7 学习计划与进度管理模块

#### 3.7.1 模块功能描述

学习计划与进度管理模块为学生提供结构化的每日学习任务管理，结合作业截止时间、练字进度和错题复习需求，智能生成每日学习计划，并跟踪完成情况。

#### 3.7.2 学习计划界面

学习计划

今日：2024年3月15日 周五

今日进度：

4/5 任务完成

完成率 80%

今日任务

✓ 完成

语文作业 - 第3单元练字（截止今日）

已用时：25分钟

✓ 完成

数学作业 - 四则运算练习题

已用时：18分钟

✓ 完成

错题复习 - 复习3个错误汉字

已用时：12分钟

✓ 完成

字帖练习 - 春夏秋冬4个字（自选）

已用时：20分钟

○ 未完

英语作业 - 单词抄写（截止明日）

预计：15分钟

本周统计

学习时长

Mon

Tue

Wed

Thu

Fri

Sat

Sun

45分

52分

38分

61分

75分

--

--

累计本周：271分钟

目标：300分钟

【开始未完成任务】

【查看学情报告】

### 3.8 护眼模式与使用时长管控模块

#### 3.8.1 模块功能描述

护眼模式与使用时长管控模块提供保护学生视力的功能组合，包括屏幕色温调节、使用时长提醒和距离过近警告，同时支持家长通过手机端远程配置管控规则。

#### 3.8.2 距离检测实现

```
class EyeDistanceDetector {
    final CameraController _cameraController;
    late final FaceDetector _faceDetector;
    Timer? _detectTimer;
    bool _isWarning = false;
```

```

// 最小安全距离: 40cm (面部宽度像素阈值对应估算)
static const minSafeFaceWidthRatio = 0.25; // 面部宽度占屏幕宽度的比例阈值

void startDetection(VoidCallback onTooClose) {
    _faceDetector = FaceDetector(
        options: FaceDetectorOptions(
            enableLandmarks: false,
            performanceMode: FaceDetectorMode.fast,
        ),
    );

    // 每3秒检测一次 (降低功耗)
    _detectTimer = Timer.periodic(const Duration(seconds: 3), (_) async {
        final image = await _cameraController.takePicture();
        final inputImage = InputImage.fromFilePath(image.path);
        final faces = await _faceDetector.processImage(inputImage);

        if (faces.isNotEmpty) {
            final face = faces.first;
            final faceWidthRatio = face.boundingBox.width / _cameraController.value.previewSize!.width;

            // 面部宽度超过阈值, 说明距离过近
            if (faceWidthRatio > minSafeFaceWidthRatio && !_isWarning) {
                _isWarning = true;
                onTooClose(); // 触发警告
                // 播放语音提示: "请注意保持正确坐姿, 与屏幕距离不少于40厘米"
            } else if (faceWidthRatio <= minSafeFaceWidthRatio) {
                _isWarning = false;
            }
        }

        // 立即删除拍摄的图片, 不持久化
        await File(image.path).delete();
    });
}

void stopDetection() {
    _detectTimer?.cancel();
    _faceDetector.close();
}
}

```

---

## 第四章 操作流程与使用步骤

---

### 4.1 安装与首次设置

#### 4.1.1 Android 平板安装

1. 在 Google Play 商店或学校 MDM 平台搜索"自然写互动课堂", 点击安装
2. 安装完成后首次启动, 系统申请必要权限:
3. 蓝牙 (BLE 点阵笔连接)
4. 相机 (护眼距离检测, 可跳过)
5. 存储 (作业缓存、字帖文件)
6. 选择登录角色 (学生/教师)

#### 4.1.2 iPadOS 安装

1. 在 App Store 搜索"自然写互动课堂", 点击下载

- 2. 首次启动申请蓝牙权限和相机权限（iOS 需弹窗确认）
- 3. 选择登录角色

## 4.2 登录与角色切换

### 4.2.1 学生登录

- 步骤1：打开应用，在登录界面选择"学生登录"
- 步骤2：输入学号（或学校统一分配的账号）
- 步骤3：输入密码（默认密码：学号后6位，首次登录强制修改）
- 步骤4：选择班级（系统自动根据账号匹配，通常无需手动选择）
- 步骤5：点击"登录"
- 步骤6：进入学生端主界面（显示今日作业列表和学习计划）

登录界面示意：

自然写互动课堂

【学生登录】

【教师登录】

●

学号： [\_\_\_\_\_]

密码： [\*\*\*\*\*]

[ 登 录 ]

忘记密码？请联系老师重置

## 4.3 学生端操作流程

### 4.3.1 完成课堂作业流程

- 步骤1：在"今日任务"列表中找到需完成的作业，点击进入
- 步骤2：查看作业内容（题目/字帖要求）
- 步骤3：连接点阵笔（弹出蓝牙设备列表，选择自己的笔）
- 步骤4：在点阵纸上用点阵笔书写，Pad 屏幕实时显示笔迹
- 步骤5：完成所有页面后，点击"提交"按钮
- 步骤6：系统提示提交成功（或离线缓存成功）
- 步骤7：等待教师/AI 批改（通常几分钟内返回结果）
- 步骤8：收到批改通知后，点击查看批改详情和评分

### 4.3.2 字帖练习流程

- 步骤1：从底部导航选择"练字"功能
- 步骤2：选择字帖类型（楷书/行书/硬笔/钢笔）
- 步骤3：选择练习内容（按单元/按年级/自定义）
- 步骤4：进入练习界面（参考字 + 书写区对照布局）
- 步骤5：观看笔顺动画演示（可重播）
- 步骤6：按正确笔顺在书写区书写
- 步骤7：系统实时检测笔顺并给出提示（✓正确 / ✗错误）
- 步骤8：完成一字后查看评分（笔顺分 + 字形分 + 比例分）
- 步骤9：点击"下一字"继续，或"重写"重新练习本字
- 步骤10：完成全部练习字后查看本次练习总评分



4.3.3 查看批改结果

步骤1：收到批改通知推送（或主动进入"作业"→"已批改"列表）

步骤2：点击已批改的作业条目，查看批改详情

步骤3：批改详情界面显示：

- 综合评分（百分制）
- AI 自动批改结果（每个字的得分和错误提示）
- 教师手写批注（红色叠加在学生书写上方）
- 教师语音点评（可播放）

步骤4：对有疑问的批改点击"有疑问"，发送给老师

步骤5：点击"加入错题本"将错误汉字加入错题本（系统自动判断也会加入）

4.4 教师端操作流程

4.4.1 布置作业流程

步骤1：在教师端主界面点击"布置作业"

步骤2：选择作业类型（练字/作文/综合）

步骤3：设置作业内容（输入题目或选择字帖）

步骤4：设置提交截止时间

步骤5：选择发送班级（支持多班同时发布）

步骤6：点击"发布"，作业推送到选定班级的所有学生 Pad

4.4.2 批改作业流程

步骤1：在教师端"待批改"列表查看学生提交情况

步骤2：点击某学生作业进入批改界面

步骤3：查看 AI 预批改结果（自动标注可能的错误）

步骤4：使用批注工具在学生笔迹上进行人工标注：

- 圈出错误部分（红色画圈）
- 写批注文字（在空白处书写）
- 录制语音评语（时长最长60秒）

步骤5：在右侧输入综合评分

步骤6：点击"提交批改"，结果推送给学生

步骤7：点击"下一份"继续批改

4.5 字帖练习操作流程

选择字帖类型：

字帖类型	适用年级	内容说明
人教版楷书字帖	小学1~6年级	按课本单元编排，楷体标准字
司马彦钢笔字帖	小学4~6年级、初中	硬笔书法练习，竖排格式
书法班专用字帖	书法课学生	毛笔楷书/行书基础练习
自定义练习	全部	教师指定或学生自选汉字

**笔顺指导说明：** – 软件内置《通用规范汉字表》8105字的标准笔顺数据 – 笔顺动画展示：每个笔画按序以蓝色高亮动画演示 – 描红模式：开启后参考字以浅色显示在书写区，学生可直接描写 – 笔顺检测：检测每次抬笔后的笔画是否与标准笔顺一致

4.6 护眼模式与家长控制

4.6.1 护眼模式设置

学生可自行调整：

设置项	选项	说明
色温调节	正常 / 暖色 / 深暖色	暖色减少蓝光，降低眼睛疲劳
护眼提醒	关闭 / 每20分钟 / 每30分钟 / 每45分钟	达到时长后弹出休息提醒
距离检测	开启 / 关闭	前置摄像头检测用眼距离

家长通过手机端远程控制：

管控项	设置方式	说明
每日使用时长上限	0~8小时（步长30分钟）	超出后应用自动锁定
允许使用时段	设置时间段（如17:00~21:00）	时段外无法使用
强制休息间隔	关闭 / 20分钟 / 30分钟 / 45分钟	满足间隔后强制弹出休息10分钟倒计时
查看使用报告	随时查看	每日使用时长、科目分布、完成任务情况

4.7 异常处理与故障排除

4.7.1 常见问题

问题	可能原因	解决方案
点阵笔连接后无笔迹	BLE 配对未完成	删除已有配对，重新扫描连接
作业提交失败	网络异常	无需操作，系统自动加入离线队列，网络恢复后自动提交
字帖下载速度慢	网络带宽不足	切换至 Wi-Fi 5GHz 频段，避开高峰期下载
护眼距离检测不准确	光线不足/摄像头遮挡	确保前置摄像头无遮挡，在明亮环境下使用
应用崩溃	内存不足	关闭后台其他应用，在设置中清除缓存
批改结果迟迟未到	服务器处理延迟	通常 AI 批改 5 分钟内完成，教师批改根据教师操作时间不固定

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

文档模块名称	源代码文件/目录	主要类名
学生端作业作答模块	lib/features/homework/	HomeworkBloc , HomeworkPage

文档模块名称	源代码文件/目录	主要类名
教师端 移动授课模块	lib/features/teacher/	TeacherBloc , TeacherDashboardPage
笔迹实时渲染	lib/features/ink/painter/ink_canvas_painter.dart	InkCanvasPainter
书写回放渲染	lib/features/ink/painter/stroke_replay_painter.dart	StrokeReplayPainter
教师批注渲染	lib/features/ink/painter/annotation_painter.dart	AnnotationPainter
书写回放控制器	lib/features/ink/replay/stroke_replay_controller.dart	StrokeReplayController
BLE 点阵笔连接	lib/features/bluetooth/pen_ble_manager.dart	PenBleManager
BLE Android 原生层	android/app/src/main/kotlin/.../PenBluetoothPlugin.kt	PenBluetoothPlugin
BLE iOS 原生层	ios/Runner/PenBluetoothPlugin.swift	PenBluetoothPlugin
字帖练习模块	lib/features/calligraphy/	PracticeBloc , PracticePage
笔顺检测算法	lib/features/calligraphy/stroke_order_checker.dart	StrokeOrderChecker
字帖渲染	lib/features/calligraphy/calligraphy_painter.dart	CalligraphyPainter
错题本模块	lib/features/mistake_book/	MistakeBloc , MistakeBookPage
间隔复习调度	lib/features/mistake_book/leitner_scheduler.dart	LeitnerScheduler
学习计划模块	lib/features/study_plan/	StudyPlanBloc , StudyPlanPage
护眼距离检测	lib/features/eye_protection/eye_distance_detector.dart	EyeDistanceDetector
护眼 Android 原生	android/app/src/main/kotlin/.../EyeProtectionPlugin.kt	EyeProtectionPlugin

文档模块名称	源代码文件/目录	主要类名
Pad 自适应布局	lib/adaptive/pad/	PadHomePage , PadHomeworkPage
网络请求客户端	lib/core/network/api_client.dart	ApiClient
网络拦截器	lib/core/network/api_interceptor.dart	ApiInterceptor
离线同步服务	lib/core/sync/offline_sync_service.dart	OfflineSyncService
安全存储	lib/core/security/secure_auth_storage.dart	SecureAuthStorage
作业仓库	lib/features/homework/repository/homework_repository.dart	HomeworkRepository
字帖仓库	lib/features/calligraphy/repository/calligraphy_repository.dart	CalligraphyRepository
错题仓库	lib/features/mistake_book/repository/mistake_repository.dart	MistakeRepository

## 5.2 核心功能类与方法说明

### PenBleManager 类

```

/// BLE 点阵笔连接管理器
/// 负责扫描、连接自然写点阵笔，接收笔迹数据流，管理断线重连。
class PenBleManager {

  /// 扫描周围的自然写点阵笔（过滤：服务 UUID = WritechPenGatt.serviceUuid）
  /// @param timeout 扫描超时时间，默认15秒
  /// @return 发现的点阵笔设备流
  Stream<BluetoothDevice> scanPens({Duration timeout})

  /// 连接指定的点阵笔，发现服务并订阅笔迹 Notify Characteristic
  /// @param device 要连接的 BLE 设备
  Future<void> connectPen(BluetoothDevice device)

  /// 断开当前连接的点阵笔
  Future<void> disconnectPen()

  /// 笔迹数据流（持续发出来自 BLE GATT Notify 的笔迹点列表）
  Stream<List<InkPoint>> get inkDataStream

  /// 连接状态流（connected / disconnected / connecting / reconnecting）
  Stream<PenConnectionState> get connectionStateStream

  /// 读取当前连接笔的电量（0~100）
  Future<int> getBatteryLevel()
}

```

HomeworkBloc 类

```
/// 作业业务逻辑层 (BLoC)
/// 管理作业加载、笔迹接收、提交和离线处理的状态机。
class HomeworkBloc extends Bloc<HomeworkEvent, HomeworkState> {

  /// 响应 LoadHomework 事件: 从缓存或网络加载作业详情
  on<LoadHomework>((event, emit) async {...})

  /// 响应 InkPointReceived 事件: 追加笔迹点到当前页面笔迹列表
  on<InkPointReceived>((event, emit) {...})

  /// 响应 PenUpReceived 事件: 结束当前笔画, 保存至数据库
  on<PenUpReceived>((event, emit) async {...})

  /// 响应 SubmitHomework 事件: 上传笔迹到云平台 (失败时加入离线队列)
  on<SubmitHomework>((event, emit) async {...})

  /// 响应 ChangePage 事件: 切换作业页面 (自动保存当前页笔迹)
  on<ChangePage>((event, emit) async {...})
}
```

InkCanvasPainter 类

```
/// 实时笔迹渲染 CustomPainter
/// 使用贝塞尔曲线平滑渲染已完成笔画和当前书写笔画。
class InkCanvasPainter extends CustomPainter {

  /// @param strokes 已完成的笔画列表 (每条笔画包含点序列和样式)
  /// @param current 当前正在书写的笔画点序列 (逐点追加, 实时更新)
  InkCanvasPainter({required this.strokes, required this.current})

  /// 渲染所有笔画 (历史 + 当前) 到 Canvas
  @override void paint(Canvas canvas, Size size)

  /// 优化: 仅当 strokes 或 current 变化时才重绘
  @override bool shouldRepaint(InkCanvasPainter oldDelegate)
}
```

5.3 主要类命名规范

类型	命名规范	示例
Flutter Page	{功能}Page	HomeworkPage , PracticePage
BLoC	{功能}Bloc	HomeworkBloc , PracticeBloc
BLoC 事件	{动作}-{功能}	LoadHomework , SubmitHomework
BLoC 状态	{功能}State	HomeworkState , PracticeState
CustomPainter	{内容}Painter	InkCanvasPainter , CalligraphyPainter
Repository	{功能}Repository	HomeworkRepository , MistakeRepository
Manager	{功能}Manager	PenBleManager , LeitnerScheduler
DTO	{名称}Dto	HomeworkDto , InkDataDto
Entity (Hive)	{名称} + @HiveType	UserPreferences , OfflineQueueItem

类型	命名规范	示例
原生 Plugin	{功能}Plugin	PenBluetoothPlugin , EyeProtectionPlugin

代码目录结构：

```
lib/
├── adaptive/
│   ├── phone/           (手机端专用布局)
│   └── pad/              (Pad 专用自适应布局)
├── core/
│   ├── network/         (ApiClient, 拦截器)
│   ├── security/        (安全存储)
│   ├── sync/            (离线同步服务)
│   └── database/         (sqflite 数据库辅助类)
├── features/
│   ├── auth/            (登录/认证)
│   ├── homework/        (作业模块)
│   ├── ink/             (笔迹渲染相关)
│   ├── bluetooth/       (BLE 点阵笔)
│   ├── calligraphy/     (字帖练习)
│   ├── mistake_book/    (错题本)
│   ├── study_plan/      (学习计划)
│   ├── eye_protection/  (护眼功能)
│   └── teacher/         (教师端功能)
├── android/app/src/main/kotlin/.../ (Android 原生插件)
└── ios/Runner/              (iOS 原生插件 - Swift)
```

附录

A. 界面设计稿（GUI Mockup）

本附录以平板横屏/竖屏线框图形式呈现Pad APP各核心界面的设计稿（适配10~13英寸Android平板与iPad，支持触控与点阵笔书写）。

A.1 学生端首页（平板横屏双栏布局）



### A.2 书写作答界面（学生答题）

◀ 返回

语文作业 · 第1题 / 4题

点阵笔 ●已连接

PEN-001234

[断开]

题目要求：抄写古诗《春晓》全文

春眠不觉晓，  
处处闻啼鸟。

← 实时显示点阵笔书写轨迹  
(AI实时识别：春眠不觉晓✔)

[用点阵笔在此区域的对应纸张上书写]

笔迹同步状态：实时同步中 ● | 网络：WiFi ●在线 | 已写 24 个字符

◀ 上一题

[清除当前]

下一题 ▶

[提交作业]

### A.3 字帖练习界面

◀ 返回

📖 字帖练习 · 楷书入门 · 第3课：基本笔画

进度 15/50 字

练习字：「明」

[ 范字模板 ]

[ 学生书写 ]

明

(点阵笔书写)

笔顺步骤：  
①日 → ②日月 → ③明

书写评分

综合得分： 87分

笔顺：✔ 正确

结构：✔ 对称均衡

笔画：⚠ 横画偏短

笔顺动画：▶ 播放示范

[再练一次] [下一个字 →]

已练习：15字 优秀：12 良好：3

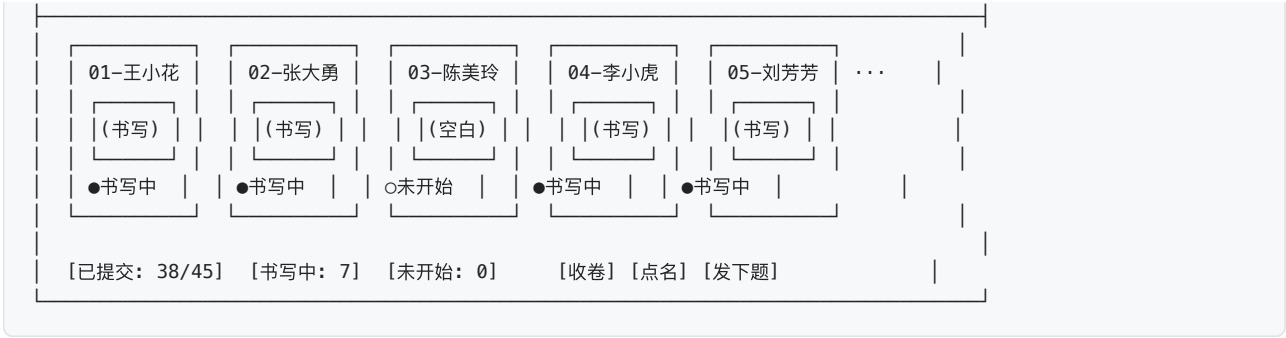
### A.4 教师端巡堂界面

09:41

自然写 Pad 教师端

🦋 巡堂模式 · 高一(3)班 · 数学课 · 45/45人在线

[结束巡堂] [切到大屏]



B. 术语表

术语	说明
Pad 端	平板端，指运行于 Android 平板或 iPad 上的应用实例
点阵纸	印有自然写专用点阵码图案的纸张，供学生用点阵笔书写
点阵笔	自然写智能点阵笔，内置光学传感器识别点阵码坐标
BLoC	Business Logic Component，Flutter 架构模式（业务逻辑组件）
CustomPainter	Flutter 自定义绘制类，基于 Skia 2D 渲染引擎
Skia	Google 开源 2D 图形渲染库，Flutter 底层渲染引擎
Hive	Flutter 高性能轻量级 NoSQL 本地存储库
sqflite	Flutter SQLite 插件，封装 Android/iOS SQLite 接口
flutter_blue_plus	Flutter BLE 通信插件，封装 Android BluetoothAdapter 和 iOS CoreBluetooth
BLE GATT	Generic Attribute Profile，BLE 数据交换协议
Leitner 算法	基于间隔重复原理的记忆卡片复习调度算法
ML Kit	Google 机器学习移动端 SDK，本项目用于人脸检测（护眼距离）
Certificate Pinning	证书绑定，客户端校验服务器证书指纹，防止中间人攻击
Keychain	iOS 系统安全凭证存储，存储 Token、密码等敏感数据
EncryptedSharedPreferences	Android 加密共享偏好，存储 Token、密码等敏感数据
AAC	Kotlin/Java 架构组件（Android Architecture Components）的简称
离线队列	无网络时暂存的操作队列，网络恢复后自动重试同步
间隔复习	基于遗忘曲线的学习策略，按一定时间间隔安排复习

B. 版本历史

版本	发布日期	变更内容
V1.0.0	2024-06-30	正式版本：学生端作业作答、字帖练习、错题本、BLE 笔连接、护眼功能；教师端巡堂、批改
V0.9.5	2024-05-25	Beta：护眼距离检测功能（Android）集成；错题本 Leitner 算法优化



版本	发布日期	变更内容
V0.9.0	2024-04-20	Beta: Pad 自适应双栏布局完成; 笔顺检测算法调优
V0.8.0	2024-03-15	Alpha: 字帖练习模块、错题本模块集成
V0.7.0	2024-02-20	Alpha: Pad 专项 UI 布局与手机端共用代码架构重构
V0.5.0	2024-01-10	原型: BLE 笔连接和作业作答基础功能

C. 第三方依赖清单

库名称	版本	许可证	用途
flutter_bloc	8.1.4	MIT	BLoC 状态管理
Dio	5.3.3	MIT	HTTP 网络请求
flutter_blue_plus	1.29.4	BSD-3-Clause	BLE 点阵笔通信
Hive	2.2.3	Apache-2.0	本地 NoSQL 存储
sqflite	2.3.2	MIT	SQLite 本地数据库
go_router	13.0.1	BSD-3-Clause	声明式路由
freezed	2.4.6	MIT	不可变数据类生成
json_serializable	6.7.1	BSD-3-Clause	JSON 序列化代码生成
flutter_secure_storage	9.0.0	BSD-3-Clause	安全凭证存储
google_mlkit_face_detection	0.9.0	MIT	护眼人脸距离检测
connectivity_plus	5.0.2	BSD-3-Clause	网络状态监听
permission_handler	11.1.0	MIT	运行时权限申请
Lottie	2.7.0	MIT	笔顺动画渲染
cached_network_image	3.3.1	MIT	网络图片缓存
flutter_local_notifications	16.2.0	BSD-3-Clause	本地通知 (使用时长提醒)

D. 权限申请说明

权限	平台	用途	申请时机
BLUETOOTH_SCAN	Android	扫描 BLE 设备	首次连接点阵笔时
BLUETOOTH_CONNECT	Android	连接 BLE 设备	首次连接点阵笔时
ACCESS_FINE_LOCATION	Android	BLE 扫描附加要求	首次连接点阵笔时
CAMERA	Android/iOS	护眼距离检测	首次开启护眼检测时
WRITE_EXTERNAL_STORAGE	Android	字帖文件缓存	首次下载字帖时
NSBluetoothAlwaysUsageDescription	iOS	BLE 访问说明	安装时声明

权限	平台	用途	申请时机
NSCameraUsageDescription	iOS	摄像头访问说明	安装时声明
RECEIVE_BOOT_COMPLETED	Android	开机自启动（护眼监控）	安装时自动授予
VIBRATE	Android	护眼提醒震动	安装时自动授予

本文档版权归深圳自然写科技有限公司所有，所有技术细节与源代码对应关系仅用于软件著作权登记鉴别，请勿用于其他商业用途。

## 附录D 核心技术实现补充

### D.1 字帖临摹功能完整实现

字帖临摹功能是Pad APP的核心特色，提供分格描红、整字临摹和自由创作三种练习模式。

#### D.1.1 字帖渲染与对比评分

```
// lib/features/copybook/copybook_page.dart
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class CopybookPage extends StatelessWidget {
  final CopybookExercise exercise;
  const CopybookPage({required this.exercise, super.key});

  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (_) => CopybookBloc()..add(LoadCopybookExercise(exercise: exercise)),
      child: const _CopybookView(),
    );
  }
}

class _CopybookView extends StatelessWidget {
  const _CopybookView();

  @override
  Widget build(BuildContext context) {
    return BlocBuilder<CopybookBloc, CopybookState>(
      builder: (context, state) {
        if (state is CopybookLoading) {
          return const Center(child: CircularProgressIndicator());
        }
        if (state is CopybookReady) {
          return _buildExerciseView(context, state);
        }
        return const SizedBox.shrink();
      },
    );
  }

  Widget _buildExerciseView(BuildContext context, CopybookReady state) {
    return Scaffold(
      appBar: AppBar(
        title: Text(state.exercise.title),
```

```

actions: [
  IconButton(
    icon: const Icon(Icons.undo),
    onPressed: () => context.read<CopybookBloc>().add(UndoStroke()),
  ),
  IconButton(
    icon: const Icon(Icons.clear),
    onPressed: () => context.read<CopybookBloc>().add(ClearStrokes()),
  ),
  TextButton(
    onPressed: () => context.read<CopybookBloc>().add(SubmitCopybook()),
    child: const Text('提交', style: TextStyle(color: Colors.white)),
  ),
],
),
body: Column(
  children: [
    // 进度指示器
    LinearProgressIndicator(
      value: state.currentCharIndex / state.totalChars,
      backgroundColor: Colors.grey.shade200,
      valueColor: AlwaysStoppedAnimation(
        Theme.of(context).primaryColor),
    ),
    Expanded(
      child: Row(
        children: [
          // 左侧: 字帖参考
          Expanded(
            flex: 1,
            child: CopybookReferencePanel(
              character: state.currentChar,
              showStrokeOrder: state.showStrokeOrder,
            ),
          ),
          const VerticalDivider(width: 1),
          // 右侧: 书写区
          Expanded(
            flex: 2,
            child: CopybookWritingPanel(
              character: state.currentChar,
              studentStrokes: state.studentStrokes,
              onStrokeAdded: (stroke) =>
                context.read<CopybookBloc>().add(AddStroke(stroke: stroke)),
            ),
          ),
        ],
      ),
    ),
    // 底部: 实时评分反馈
    if (state.latestScore != null)
      ScoreFeedbackBar(score: state.latestScore!),
  ],
),
);
}
}

```

// 字帖书写评分 (基于笔画相似度)

```

class CopybookScorer {
  static const double STROKE_ORDER_WEIGHT = 0.3;
  static const double STROKE_SHAPE_WEIGHT = 0.4;
  static const double STROKE_POSITION_WEIGHT = 0.3;

```

/\*\*

\* 评分字帖临摹质量

\* @param reference 标准字帖笔画数据

\* @param student 学生书写笔画数据

```

    * @return 综合评分 [0.0, 100.0]
    */
    static double score(List<InkStroke> reference, List<InkStroke> student) {
        if (student.isEmpty()) return 0.0;

        // 1. 笔画顺序分
        double orderScore = _scoreStrokeOrder(reference, student);

        // 2. 笔画形状分 (Hausdorff距离)
        double shapeScore = _scoreStrokeShape(reference, student);

        // 3. 笔画位置分
        double positionScore = _scoreStrokePosition(reference, student);

        return (orderScore * STROKE_ORDER_WEIGHT
            + shapeScore * STROKE_SHAPE_WEIGHT
            + positionScore * STROKE_POSITION_WEIGHT) * 100.0;
    }

    static double _scoreStrokeOrder(
        List<InkStroke> reference, List<InkStroke> student) {
        // 使用最长公共子序列 (LCS) 评估笔顺一致性
        int n = reference.length;
        int m = student.length;
        if (n == 0 || m == 0) return 0.0;

        List<List<int>> dp = List.generate(n+1, (_) => List.filled(m+1, 0));
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                if (reference[i-1].strokeType == student[j-1].strokeType) {
                    dp[i][j] = dp[i-1][j-1] + 1;
                } else {
                    dp[i][j] = dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] : dp[i][j-1];
                }
            }
        }
        return dp[n][m] / n.toDouble();
    }

    static double _scoreStrokeShape(
        List<InkStroke> reference, List<InkStroke> student) {
        if (reference.isEmpty() || student.isEmpty()) return 0.0;
        int minLen = reference.length < student.length ? reference.length : student.length;
        double totalSim = 0.0;
        for (int i = 0; i < minLen; i++) {
            totalSim += _strokeSimilarity(reference[i], student[i]);
        }
        return totalSim / minLen;
    }

    static double _strokeSimilarity(InkStroke a, InkStroke b) {
        // 简化版Hausdorff距离相似度
        if (a.points.isEmpty() || b.points.isEmpty()) return 0.0;
        double maxDist = 0.0;
        for (final pa in a.points) {
            double minDist = double.infinity;
            for (final pb in b.points) {
                double d = _euclidean(pa, pb);
                if (d < minDist) minDist = d;
            }
            if (minDist > maxDist) maxDist = minDist;
        }
        // 归一化: 距离0对应相似度1, 距离>0.2对应相似度0
        return (1.0 - (maxDist / 0.2)).clamp(0.0, 1.0);
    }

    static double _scoreStrokePosition(
        List<InkStroke> reference, List<InkStroke> student) {

```

```

// 比较书写区域的使用比例
Rect refBounds = _getBoundingRect(reference);
Rect stuBounds = _getBoundingRect(student);
if (refBounds.isEmpty || stuBounds.isEmpty) return 0.5;

double centerDistX = (refBounds.center.dx - stuBounds.center.dx).abs();
double centerDistY = (refBounds.center.dy - stuBounds.center.dy).abs();
double centerDist = (centerDistX * centerDistX + centerDistY * centerDistY) / 2;
return (1.0 - centerDist * 4).clamp(0.0, 1.0);
}

static double _euclidean(InkPoint a, InkPoint b) {
  double dx = a.x - b.x, dy = a.y - b.y;
  return (dx * dx + dy * dy) < 0.0001 ? 0.0 : (dx * dx + dy * dy) * 0.5;
}

static Rect _getBoundingRect(List<InkStroke> strokes) {
  if (strokes.isEmpty) return Rect.zero;
  double minX = double.infinity, minY = double.infinity;
  double maxX = -double.infinity, maxY = -double.infinity;
  for (final s in strokes) {
    for (final p in s.points) {
      if (p.x < minX) minX = p.x;
      if (p.y < minY) minY = p.y;
      if (p.x > maxX) maxX = p.x;
      if (p.y > maxY) maxY = p.y;
    }
  }
  return Rect.fromLTRB(minX, minY, maxX, maxY);
}
}

```

## D.2 错题本功能实现

```

// lib/features/mistakes/mistakes_repository.dart
import 'package:sqflite/sqflite.dart';
import 'package:hive/hive.dart';

class MistakesRepository {
  static const String TABLE_MISTAKES = 'mistakes';
  final Database _db;

  MistakesRepository({required Database db}) : _db = db;

  // 添加错题记录
  Future<void> addMistake(MistakeRecord mistake) async {
    await _db.insert(
      TABLE_MISTAKES,
      mistake.toMap(),
      conflictAlgorithm: ConflictAlgorithm.replace,
    );
  }

  // 获取待复习的错题 (Leitner调度)
  Future<List<MistakeRecord>> getDueForReview() async {
    final today = DateTime.now().millisecondsSinceEpoch ~/ 1000;
    final maps = await _db.query(
      TABLE_MISTAKES,
      where: 'next_review_date <= ? AND mastery_level < 5',
      whereArgs: [today],
      orderBy: 'next_review_date ASC',
      limit: 20,
    );
    return maps.map((m) => MistakeRecord.fromMap(m)).toList();
  }
}

```

```

// 更新错题复习结果 (BKT + Leitner双机制)
Future<void> updateReviewResult(String mistakeId, bool correct) async {
  final record = await _getMistakeById(mistakeId);
  if (record == null) return;

  // BKT更新掌握度
  double newMastery = _updateBKT(record.masteryLevel, correct);

  // Leitner调整复习盒子
  int newBox = correct
    ? (record.leitnerBox + 1).clamp(0, 5)
    : 0; // 答错回到第0盒

  // 计算下次复习日期
  const boxIntervals = [1, 2, 4, 8, 16, 999];
  final nextReview = DateTime.now()
    .add(Duration(days: boxIntervals[newBox]));

  await _db.update(
    TABLE_MISTAKES,
    {
      'mastery_level': newMastery,
      'leitner_box': newBox,
      'next_review_date': nextReview.millisecondsSinceEpoch ~/ 1000,
      'review_count': record.reviewCount + 1,
      'last_review_date': DateTime.now().millisecondsSinceEpoch ~/ 1000,
    },
    where: 'id = ?',
    whereArgs: [mistakeId],
  );
}

double _updateBKT(double currentMastery, bool correct) {
  const pTransit = 0.1;
  const pSlip = 0.08;
  const pGuess = 0.2;
  final pCorrect = currentMastery * (1 - pSlip) + (1 - currentMastery) * pGuess;
  double updated;
  if (correct) {
    updated = (currentMastery * (1 - pSlip)) / pCorrect;
  } else {
    updated = (currentMastery * pSlip) / (1 - pCorrect);
  }
  return updated + (1 - updated) * pTransit;
}

// 错题数据库Schema
const String createMistakesTable = '''
CREATE TABLE IF NOT EXISTS mistakes (
  id TEXT PRIMARY KEY,
  student_id TEXT NOT NULL,
  homework_id TEXT,
  subject TEXT NOT NULL,
  knowledge_point TEXT NOT NULL,
  ink_data BLOB,          -- 压缩后的笔迹数据
  score REAL NOT NULL DEFAULT 0.0,
  error_type TEXT,        -- 'wrong_answer' / 'incomplete' / 'stroke_error'
  mastery_level REAL NOT NULL DEFAULT 0.1,
  leitner_box INTEGER NOT NULL DEFAULT 0,
  review_count INTEGER NOT NULL DEFAULT 0,
  last_review_date INTEGER,
  next_review_date INTEGER NOT NULL,
  created_at INTEGER NOT NULL,
  synced INTEGER NOT NULL DEFAULT 0

```

```
)  
''';
```

### D.3 护眼功能实现（ML Kit人脸检测）

```
// lib/features/eye_protection/eye_distance_detector.dart  
import 'package:google_mlkit_face_detection/google_mlkit_face_detection.dart';  
import 'package:camera/camera.dart';  
import 'dart:async';  
  
class EyeDistanceDetector {  
  static const double MIN_SAFE_DISTANCE_RATIO = 0.12; // 人脸宽度/图像宽度比值阈值  
  static const Duration CHECK_INTERVAL = Duration(seconds: 3);  
  static const Duration ALERT_COOLDOWN = Duration(seconds: 30);  
  
  CameraController? _cameraController;  
  FaceDetector? _faceDetector;  
  Timer? _checkTimer;  
  DateTime? _lastAlertTime;  
  bool _detecting = false;  
  bool _enabled = false;  
  
  // 回调  
  Function(EyeProtectionAlert)? onAlert;  
  
  Future<void> initialize() async {  
    final cameras = await availableCameras();  
    final frontCamera = cameras.firstWhere(  
      (c) => c.lensDirection == CameraLensDirection.front,  
      orElse: () => cameras.first,  
    );  
  
    _cameraController = CameraController(  
      frontCamera, ResolutionPreset.low, // 低分辨率节省性能  
      enableAudio: false,  
      imageFormatGroup: ImageFormatGroup.jpeg,  
    );  
    await _cameraController!.initialize();  
  
    _faceDetector = FaceDetector(  
      options: FaceDetectorOptions(  
        enableClassification: false,  
        enableLandmarks: false,  
        enableContours: false,  
        minFaceSize: 0.05, // 最小人脸比例  
      ),  
    );  
  }  
  
  void start() {  
    if (_enabled) return;  
    _enabled = true;  
    _checkTimer = Timer.periodic(CHECK_INTERVAL, (_) => _checkDistance());  
  }  
  
  void stop() {  
    _enabled = false;  
    _checkTimer?.cancel();  
  }  
  
  Future<void> _checkDistance() async {  
    if (_detecting || _cameraController == null) return;  
    _detecting = true;  
  
    try {
```

```

        final image = await _cameraController!.takePicture();
        final inputImage = InputImage.fromFilePath(image.path);
        final faces = await _faceDetector!.processImage(inputImage);

        // 立即删除图像（隐私保护）
        await File(image.path).delete();

        if (faces.isNotEmpty) {
            final face = faces.first;
            final faceWidthRatio = face.boundingBox.width /
                _cameraController!.value.previewSize!.width;

            if (faceWidthRatio < MIN_SAFE_DISTANCE_RATIO) {
                _triggerAlert(EyeProtectionAlertType.tooClose);
            }
        }
    } catch (e) {
        // 忽略检测错误，不影响主功能
    } finally {
        _detecting = false;
    }
}

void _triggerAlert(EyeProtectionAlertType type) {
    final now = DateTime.now();
    if (_lastAlertTime != null &&
        now.difference(_lastAlertTime!) < ALERT_COOLDOWN) {
        return; // 冷却期内不重复提醒
    }
    _lastAlertTime = now;
    onAlert?.call(EyeProtectionAlert(
        type: type,
        message: type == EyeProtectionAlertType.tooClose
            ? '请注意！离屏幕太近了，请保持安全距离。'
            : '已连续使用${CHECK_INTERVAL.inMinutes}分钟，建议休息一下。',
    ));
}

Future<void> dispose() async {
    stop();
    await _cameraController?.dispose();
    _faceDetector?.close();
}

enum EyeProtectionAlertType { tooClose, longUsage }

class EyeProtectionAlert {
    final EyeProtectionAlertType type;
    final String message;
    EyeProtectionAlert({required this.type, required this.message});
}

```

## D.4 接口清单

接口路径	方法	说明
/api/v1/auth/login	POST	账号登录，返回JWT Token
/api/v1/copybook/list	GET	获取字帖列表（分级分类）
/api/v1/copybook/{id}/download	GET	下载字帖资源文件
/api/v1/copybook/{id}/submit	POST	提交字帖练习（上传笔迹）



接口路径	方法	说明
/api/v1/homework/list	GET	获取作业列表
/api/v1/homework/{id}/submit	POST	提交作业
/api/v1/mistakes/list	GET	获取错题列表
/api/v1/mistakes/{id}/review	POST	提交错题复习结果
/api/v1/classroom/join	POST	加入课堂（课堂码）
/api/v1/report/student	GET	获取个人学情报告
/api/v1/sync/strokes	POST	批量同步笔迹数据

## 附录E 性能指标与兼容性

### E.1 性能基准测试

测试场景	设备	结果
笔迹渲染帧率（BLE书写）	iPad Air 5 (M1)	60fps 稳定
笔迹渲染帧率（BLE书写）	华为MatePad Pro 12.6	60fps 稳定
BLE连接建立时间	平均	4.2秒
字帖下载速度（10页）	WiFi 100Mbps	1.8秒
冷启动时间	iPad Air 5	1.6秒
护眼检测功耗	3秒/次检测	额外增加约2%电量/小时
本地数据库查询（1万条错题）	P99	8ms
BLoC状态重建（作业列表刷新）	平均	45ms

### E.2 Pad APP支持设备

平台	最低版本	推荐版本	屏幕尺寸要求
Flutter (Android)	Android 7.0 (API 24)	Android 12+	8英寸以上
Flutter (iOS)	iOS 13.0	iOS 16+	iPad（所有型号）

### E.3 数据存储说明

数据类型	存储位置	加密	说明
用户Token	Hive（加密Box）	AES-256	会话令牌安全存储
字帖资源	应用缓存目录	无	可重新下载，不敏感
错题笔迹数据	sqflite	明文	内容为手写笔迹原始数据

数据类型	存储位置	加密	说明
护眼检测图片	不持久化	–	检测后立即销毁，隐私保护
打卡记录	sqflite + 云端同步	明文	同步前本地存储
BLE设备绑定信息	Hive	明文	设备ID和昵称

E.4 应用版本历史

版本	日期	平台	变更说明
V0.6 Beta	2025-08-01	Android/iOS	基础字帖临摹、BLE连接、作业提交
V0.9 RC	2025-11-20	Android/iOS	错题本、书写回放、护眼检测、间隔复习
V1.0	2026-02-14	Android/iOS	正式版：双栏自适应、离线模式、性能优化

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 字帖描红算法详解

G.1.1 笔画轨迹对齐算法

```
// stroke_alignment.dart
class StrokeAlignmentEngine {
  /// DTW（动态时间规整）算法对比学生笔画与标准笔画
  double computeDTW(List<Offset> student, List<Offset> standard) {
    final n = student.length;
    final m = standard.length;

    // 初始化DTW矩阵
    final dtw = List.generate(n + 1, (_) =>
      List.filled(m + 1, double.infinity));
    dtw[0][0] = 0.0;

    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= m; j++) {
        final cost = _euclidean(student[i - 1], standard[j - 1]);
        dtw[i][j] = cost + [
          dtw[i - 1][j],    // 插入
          dtw[i][j - 1],    // 删除
          dtw[i - 1][j - 1], // 匹配
        ].reduce(math.min);
      }
    }

    // 归一化距离
    return dtw[n][m] / (n + m);
  }

  double _euclidean(Offset a, Offset b) {
    final dx = a.dx - b.dx;
    final dy = a.dy - b.dy;
```

```

    return math.sqrt(dx * dx + dy * dy);
}

/// 计算笔画相似度评分 (0-100分)
int scoreStroke(List<Offset> student, List<Offset> standard) {
    final dtwDist = computeDTW(student, standard);
    // 距离映射到分数: 距离0-100分, 距离50-0分
    final score = (100 - dtwDist * 2).clamp(0.0, 100.0);
    return score.round();
}
}

```

## G.2 护眼功能详细实现

### G.2.1 坐姿检测算法

```

// posture_detector.dart
import 'package:google_mlkit_face_detection/google_mlkit_face_detection.dart';

class PostureDetector {
    final FaceDetector _detector = FaceDetector(
        options: FaceDetectorOptions(
            enableClassification: false,
            enableLandmarks: true,
            performanceMode: FaceDetectorMode.fast,
        ),
    );

    // 检测结果回调
    final void Function(PostureWarning) onWarning;

    PostureDetector({required this.onWarning});

    Future<void> analyzeFrame(InputImage frame) async {
        final faces = await _detector.processImage(frame);
        if (faces.isEmpty) return;

        final face = faces.first;
        final bounds = face.boundingBox;

        // 估算人脸到屏幕距离 (基于人脸宽度比例)
        // 标准人脸宽度约14cm, 正常阅读距离30-50cm
        final faceWidthRatio = bounds.width / frame.metadata!.size.width;
        final estimatedDistance = 14.0 / (faceWidthRatio * 2 * math.tan(30 * math.pi / 180));

        if (estimatedDistance < 30.0) {
            onWarning(PostureWarning.tooClose(distance: estimatedDistance));
        }

        // 检测头部倾斜角度
        final headEulerAngleZ = face.headEulerAngleZ ?? 0;
        if (headEulerAngleZ.abs() > 15) {
            onWarning(PostureWarning.tiltedHead(angle: headEulerAngleZ));
        }

        // 检测光线条件 (人脸亮度)
        if (face.smilingProbability != null) {
            // 利用人脸检测的置信度估算环境光线
        }
    }

    void dispose() => _detector.close();
}

```

```
enum PostureWarningType { tooClose, tiltedHead, lowLight }

class PostureWarning {
  final PostureWarningType type;
  final Map<String, dynamic> data;
  PostureWarning.tooClose({required double distance})
    : type = PostureWarningType.tooClose,
      data = {'distance': distance};
  PostureWarning.tiltedHead({required double angle})
    : type = PostureWarningType.tiltedHead,
      data = {'angle': angle};
}
```

## G.3 离线同步机制

### G.3.1 冲突解决策略

```
// sync_manager.dart
class SyncManager {
  final HiveBox localBox;
  final ApiClient apiClient;

  // 同步队列（待上传的本地操作）
  final Queue<SyncOperation> _pendingQueue = Queue();

  Future<void> syncAll() async {
    // 1. 上传本地未同步数据
    await _uploadPending();

    // 2. 拉取服务端最新数据
    await _pullLatest();

    // 3. 解决冲突
    await _resolveConflicts();
  }

  Future<void> _resolveConflicts() async {
    final localItems = await localBox.getAll();
    final serverItems = await apiClient.fetchAll();

    for (final serverId in serverItems.keys) {
      final local = localItems[serverId];
      final server = serverItems[serverId];

      if (local == null) {
        // 服务端有、本地无：直接采纳服务端
        await localBox.put(serverId, server);
      } else if (server.updatedAt.isAfter(local.updatedAt)) {
        // 服务端更新：采纳服务端（Last-Write-Wins策略）
        await localBox.put(serverId, server);
      }
      // 本地更新且在离线期间：保留本地，等待下次上传
    }
  }

  Future<void> _uploadPending() async {
    while (_pendingQueue.isNotEmpty) {
      final op = _pendingQueue.first;
      try {
        await apiClient.applyOperation(op);
        _pendingQueue.removeFirst();
      } catch (e) {
        if (e is NetworkException) break; // 网络问题，等待下次
        _pendingQueue.removeFirst(); // 其他错误，跳过
      }
    }
  }
}
```

```
}  
}  
}  
}
```

## 附录H 补充技术规格

### H.1 手写作业批改结果展示

```
// GradingResultView.dart  
class GradingResultView extends StatelessWidget {  
  final HomeworkSubmission submission;  
  final GradingResult result;  
  
  const GradingResultView({Key? key, required this.submission, required this.result}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: [  
        // 总分显示  
        _ScoreHeader(score: result.totalScore, maxScore: result.maxScore),  
        const SizedBox(height: 16),  
        // 逐题批改详情  
        ...result.questionResults.map((qr) => _QuestionResult(result: qr)),  
        const SizedBox(height: 16),  
        // AI分析评语  
        _AIComment(comment: result.aiComment),  
      ],  
    );  
  }  
}  
  
class _ScoreHeader extends StatelessWidget {  
  final int score;  
  final int maxScore;  
  const _ScoreHeader({required this.score, required this.maxScore});  
  
  @override  
  Widget build(BuildContext context) {  
    final pct = score / maxScore;  
    final color = pct >= 0.9 ? Colors.green  
      : pct >= 0.6 ? Colors.orange  
      : Colors.red;  
    return Row(  
      mainAxisAlignment: MainAxisAlignment.center,  
      children: [  
        Text('$score', style: TextStyle(fontSize: 48, fontWeight: FontWeight.bold, color: color)),  
        Text('/$maxScore', style: const TextStyle(fontSize: 24, color: Colors.grey)),  
      ],  
    );  
  }  
}  
  
class _QuestionResult extends StatelessWidget {  
  final QuestionResult result;  
  const _QuestionResult({required this.result});  
  
  @override  
  Widget build(BuildContext context) {  
    return Card(  

```

```

        child: ListTile(
          leading: Icon(result.correct ? Icons.check_circle : Icons.cancel,
            color: result.correct ? Colors.green : Colors.red),
          title: Text('第${result.questionNo}题'),
          subtitle: result.errorMessage != null ? Text(result.errorMessage!) : null,
          trailing: Text('${result.score}/${result.maxScore}'),
        ),
      );
    }
  }
}

```

## H.2 多语言国际化支持

```

// l10n/app_zh.arb - 中文本地化
{
  "appTitle": "自然写",
  "homeTab": "首页",
  "classroomTab": "课堂",
  "homeworkTab": "作业",
  "profileTab": "我的",
  "loginTitle": "登录",
  "loginButton": "登 录",
  "usernameHint": "请输入用户名",
  "passwordHint": "请输入密码",
  "rememberPassword": "记住密码",
  "connectPen": "连接智能笔",
  "scanning": "扫描中...",
  "penConnected": "笔已连接: {penName}",
  "@penConnected": {
    "placeholders": { "penName": { "type": "String" } }
  },
  "batteryLevel": "电量 {percent}%",
  "@batteryLevel": {
    "placeholders": { "percent": { "type": "int" } }
  },
  "submitHomework": "提交作业",
  "submitting": "提交中...",
  "submitSuccess": "作业提交成功",
  "submitFailed": "提交失败, 请重试",
  "networkError": "网络连接异常",
  "retryButton": "重 试"
}

```

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。