classPlatform Mobile 产品报告

项目名称：classPlatform Mobile
开发者：胡傲东
提交日期：2026年1月19日
GitHub 仓库：https://github.com/JadeSnow7/graduationDesign

一、产品功能介绍

1.1 产品概述

classPlatform Mobile 是一款基于 React Native (Expo) 开发的跨平台 AI 智能辅导应用，支持 iOS、Android 和 Web 三端运行。该应用专注于为学生和教师提供完整的课程管理和智能辅导体验。

1.2 核心功能模块

功能模块	描述	用户角色
🔐 **用户认证**	账户登录、会话持久化	所有用户
📚 **课程管理**	课程列表浏览、课程详情查看	学生/教师
📖 **章节学习**	章节内容阅读、学习时长追踪	学生
💬 **AI 智能聊天**	三种对话模式：导师/解题/模拟	学生
📝 **作业管理**	作业列表、创建作业、提交作业	学生/教师
📋 **测验系统**	测验列表、创建测验	学生/教师
📁 **资源管理**	资源列表、创建资源	学生/教师
👤 **个人中心**	学习统计、缓存管理、退出登录	所有用户

1.3 功能详情

1.3.1 用户认证

- 支持用户名/密码登录
- JWT Token 认证机制
- 登录状态本地持久化（AsyncStorage）
- 自动恢复上次会话

1.3.2 课程管理

- 下拉刷新获取最新课程列表
- 展示课程名称、教师、学生数量
- 课程详情页包含四个 Tab：章节、作业、测验、资源

1.3.3 章节学习

- 章节内容渲染展示
- **学习时长追踪**：每 30 秒自动上报心跳数据

- 离开页面时自动提交最终学习时长

1.3.4 AI 智能聊天

- **导师模式**：苏格拉底式提问，引导学生自主思考
- **解题模式**：直接给出答案和详细解题步骤
- **模拟模式**：通过实验场景解释抽象概念
- 聊天记录本地缓存，支持离线查看

1.3.5 教师功能

- 创建作业（标题、描述、截止日期）
- 创建测验（标题、描述、时间限制）
- 创建资源（视频/文档/链接类型）

二、程序概要设计

2.1 项目目录结构

```
code/mobile/
├── App.tsx                      # 应用入口
├── app.json                     # Expo 配置
├── package.json                 # 依赖配置
└── src/
    ├── api.ts                   # 网络请求层（15+ 接口）
    ├── config.ts                # 配置常量
    ├── storage.ts               # 本地存储管理
    ├── types.ts                 # TypeScript 类型定义
    ├── navigation/
    │   └── AppNavigator.tsx     # 导航系统
    ├── screens/
    │   ├── LoginScreen.tsx      # 登录页面
    │   ├── CoursesScreen.tsx    # 课程列表
    │   ├── CourseDetailScreen.tsx   # 课程详情（含 Tabs）
    │   ├── ChapterContentScreen.tsx # 章节学习
    │   ├── ChatScreen.tsx       # AI 聊天
    │   ├── ProfileScreen.tsx    # 个人中心
    │   └── CreateItemScreen.tsx # 创建表单（教师）
    └── components/
        └── MessageBubble.tsx    # 消息气泡组件
```

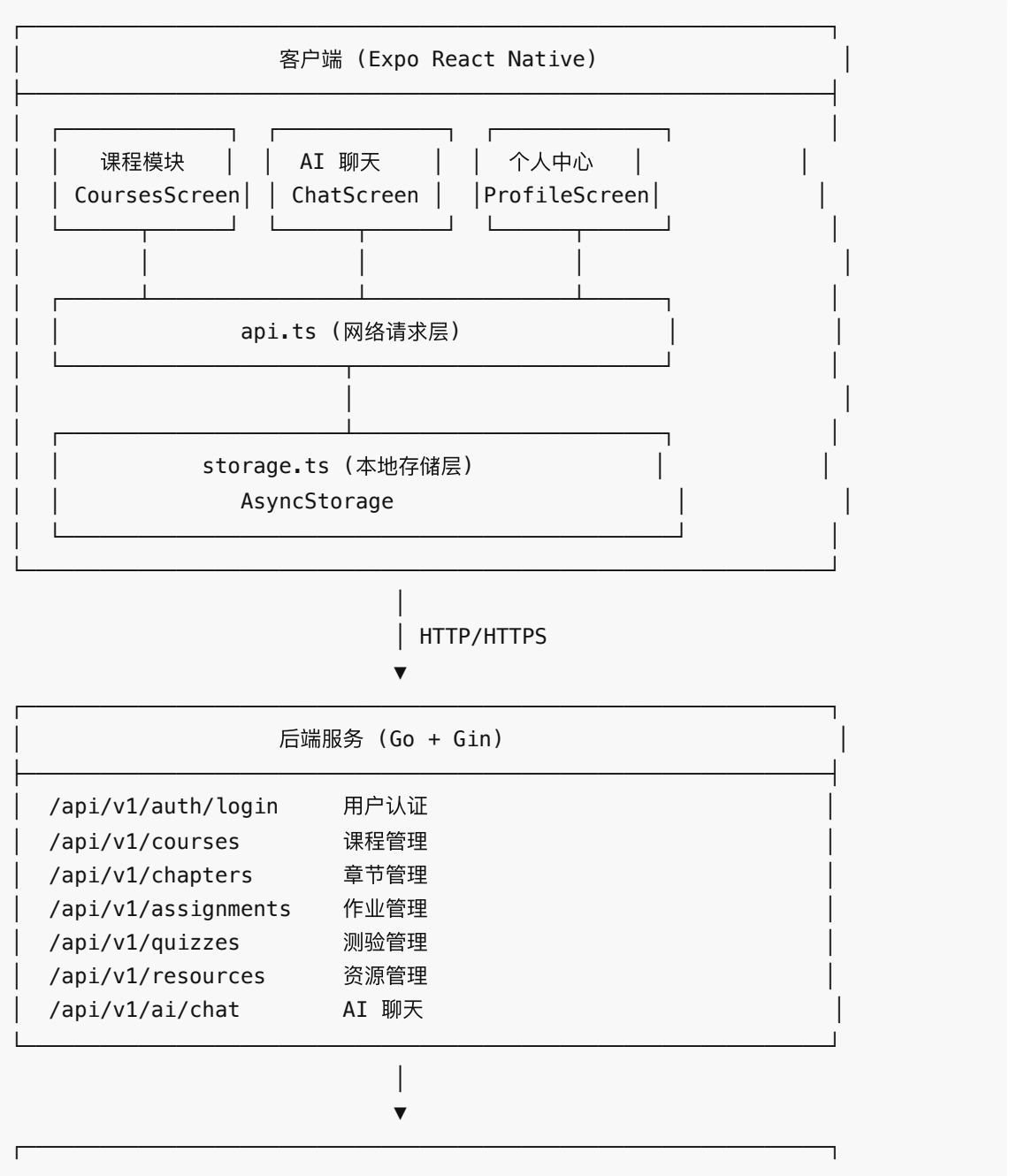
2.2 核心模块说明

模块	职责
App.tsx	应用入口，管理全局状态（会话、消息），包装 SafeAreaProvider
AppNavigator.tsx	导航系统，实现底部 Tab + Stack 嵌套导航
api.ts	封装 HTTP 请求，处理超时、取消、多种响应格式

storage.ts	AsyncStorage 封装，管理会话和消息持久化
CourseDetailScreen.tsx	课程详情，Tab 切换，教师可创建作业/测验/资源
ChapterContentScreen.tsx	章节学习，学习时长心跳上报
ChatScreen.tsx	AI 聊天，三种模式切换，消息列表

三、软件架构图

3.1 整体架构



```
│                          存储层                                    │
├──────────────────────────┬─────────────────────────────────────────┤
│    MySQL (用户/课程)      │      AI Service (Ollama + RAG)          │
└──────────────────────────┴─────────────────────────────────────────┘
```

3.2 导航架构

```
AppNavigator
├── AuthStack (未登录)
│   └── LoginScreen
└── MainTabs (已登录)
    ├── HomeTab (📚 课程)
    │   └── HomeStack
    │       ├── CoursesScreen
    │       ├── CourseDetailScreen
    │       └── ChapterContentScreen
    ├── ChatTab (💬 AI 助教)
    │   └── ChatScreen
    └── ProfileTab (👤 我的)
        └── ProfileScreen
```

四、技术亮点及实现原理

4.1 网络请求优化

请求超时与取消机制

```
async function request<T>(path: string, options: RequestInit = {}):
Promise<T> {
  const controller = new AbortController();
  const timeoutId = setTimeout(() => controller.abort(),
NETWORK_TIMEOUT_MS);

  // 支持外部取消信号
  if (externalSignal?.aborted) {
    controller.abort();
  }

  try {
    const response = await fetch(buildUrl(path), {
      signal: controller.signal,
      // ...
    });
    // 处理多种响应格式
  } finally {
    clearTimeout(timeoutId);
```

```
  }
}
```

技术亮点：

- 统一 15 秒超时控制
- 支持外部 AbortController 取消请求
- 兼容三种后端响应格式：`{success, data}`、`{data}`、直接返回

4.2 学习时长追踪

心跳上报机制

```
useEffect(() => {
  const sendHeartbeat = async () => {
    const elapsed = Math.floor((now - lastHeartbeatRef.current) / 1000);
    if (elapsed > 0 && isActiveRef.current) {
      await recordStudyTime(session.token, session.tokenType, chapterId,
elapsed);
      setStudySeconds((prev) => prev + elapsed);
      lastHeartbeatRef.current = now;
    }
  };

  const subscription = AppState.addEventListener('change',
handleAppStateChange);
  const intervalId = setInterval(sendHeartbeat, 30000); // 每30秒

  return () => {
    subscription.remove();
    clearInterval(intervalId);
    sendHeartbeat(); // 页面卸载时发送最终心跳
  };
}, [chapterId]);
```

技术亮点：

- 每 30 秒自动上报学习时长
- 监听 AppState 变化，后台时暂停计时
- 页面离开时发送最终心跳，确保数据不丢失

4.3 本地存储持久化

```
// 消息存储限制
export async function saveMessages(messages: ChatMessage[]): Promise<void>
{
  const trimmed = messages.slice(-MAX_HISTORY); // 限制最多 50 条
  await AsyncStorage.setItem(STORAGE_KEYS.messages,
```

```
JSON.stringify(trimmed));
}
```

技术亮点：

- 消息数量限制（MAX_HISTORY = 50），防止存储膨胀
- 会话与消息分离存储，便于独立管理
- 应用启动时自动恢复历史数据

4.4 竞态条件处理

```
const requestIdRef = useRef(0);
const mountedRef = useRef(true);

const handleSend = async () => {
  const requestId = requestIdRef.current + 1;
  requestIdRef.current = requestId;

  // 发送请求...

  if (!mountedRef.current || requestId !== requestIdRef.current) {
    return; // 忽略过期响应
  }
  // 更新状态...
};
```

技术亮点：

- requestId 追踪确保只处理最新请求的响应
- mountedRef 防止组件卸载后更新状态
- 组件卸载时自动取消进行中的请求

4.5 教师角色功能控制

```
const isTeacher = session.user.role === 'teacher' || session.user.role ===
'admin';

// 仅教师可见的 FAB 按钮
{isTeacher && activeTab !== 'chapters' && (
  <Pressable style={styles.fab} onPress={() => setShowCreateModal(true)}>
    <Text style={styles.fabText}>+</Text>
  </Pressable>
)}
```

技术亮点：

- 基于用户角色动态显示/隐藏功能

- Modal 表单支持创建作业、测验、资源

五、技术栈总结

类别	技术选型
框架	React Native (Expo SDK 54)
语言	TypeScript
导航	React Navigation 7.x
状态管理	React Hooks (useState, useEffect, useRef)
网络请求	Fetch API + AbortController
本地存储	@react-native-async-storage/async-storage
UI 组件	React Native 原生组件 + Modal
后端	Go + Gin + MySQL
AI 服务	Ollama + RAG (GraphRAG)

六、作业要求对照

要求项	实现情况	说明
✅ 功能要求	已满足	AI 智能聊天 + 课程管理 + 作业/测验/资源创建
✅ 技术要求 (网络)	已满足	HTTP API 调用（15+ 接口）
✅ 技术要求 (存储)	已满足	AsyncStorage 本地持久化
✅ 性能要求	已满足	FlatList 虚拟化、请求取消、心跳优化

七、提交内容

1. **产品报告**：本文档 (PDF)
2. **源代码**：https://github.com/JadeSnow7/graduationDesign
3. **演示录屏**：见附件

文档生成时间：2026-01-19