心音智鉴小程序 - 功能设计方案

概述

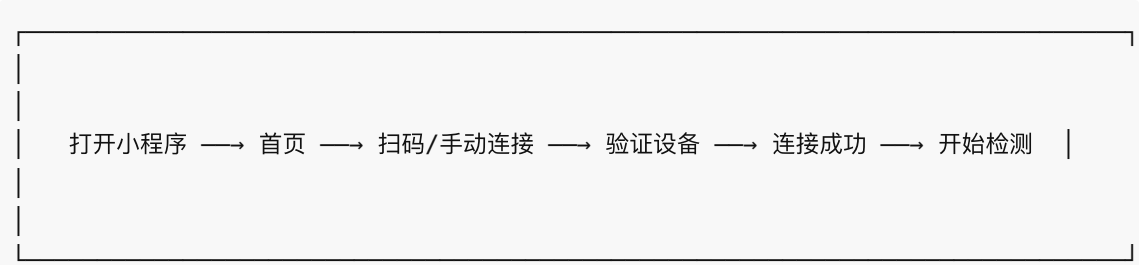
本文档综合三个专业角色的分析成果，形成小程序功能设计方案，包含：

- **首页模块**：设备连接（扫码/手动）
- **检测中心模块**：检测流程（准备→录制→分析→结果）

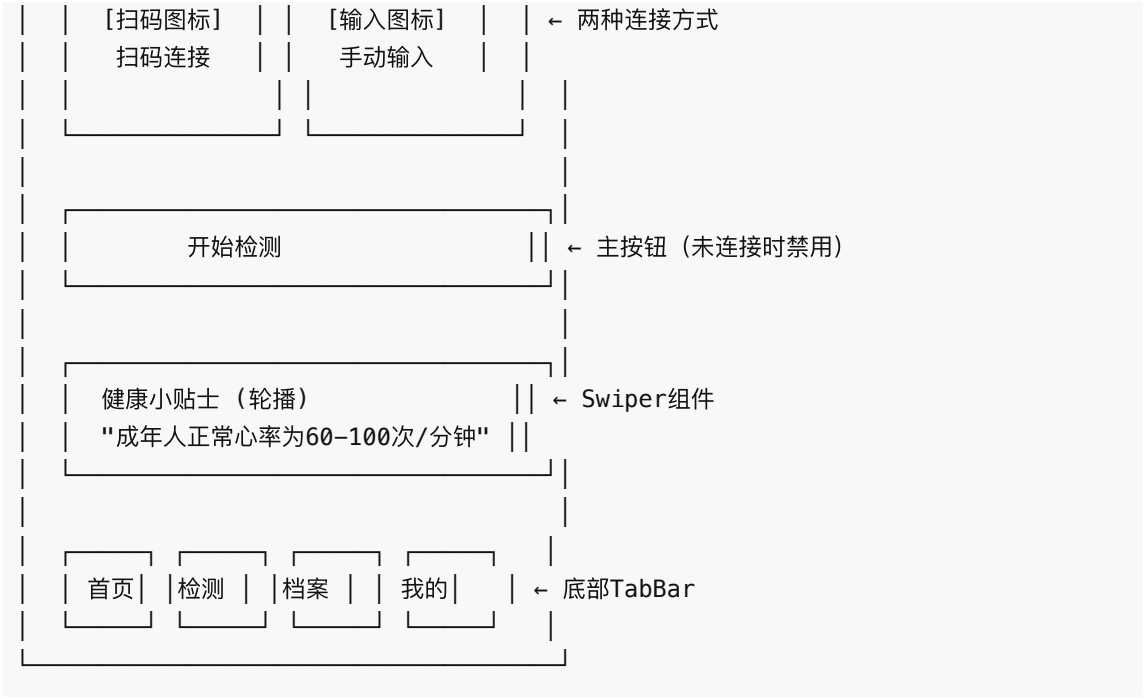
角色	贡献内容
产品经理	用户故事、功能优先级、信息架构、验收标准
UI设计师	页面布局、组件规格、视觉规范、响应式适配
领域专家	科普术语、健康建议文案、检测引导、免责声明

一、首页模块 - 设备连接

1.1 完整用户流程

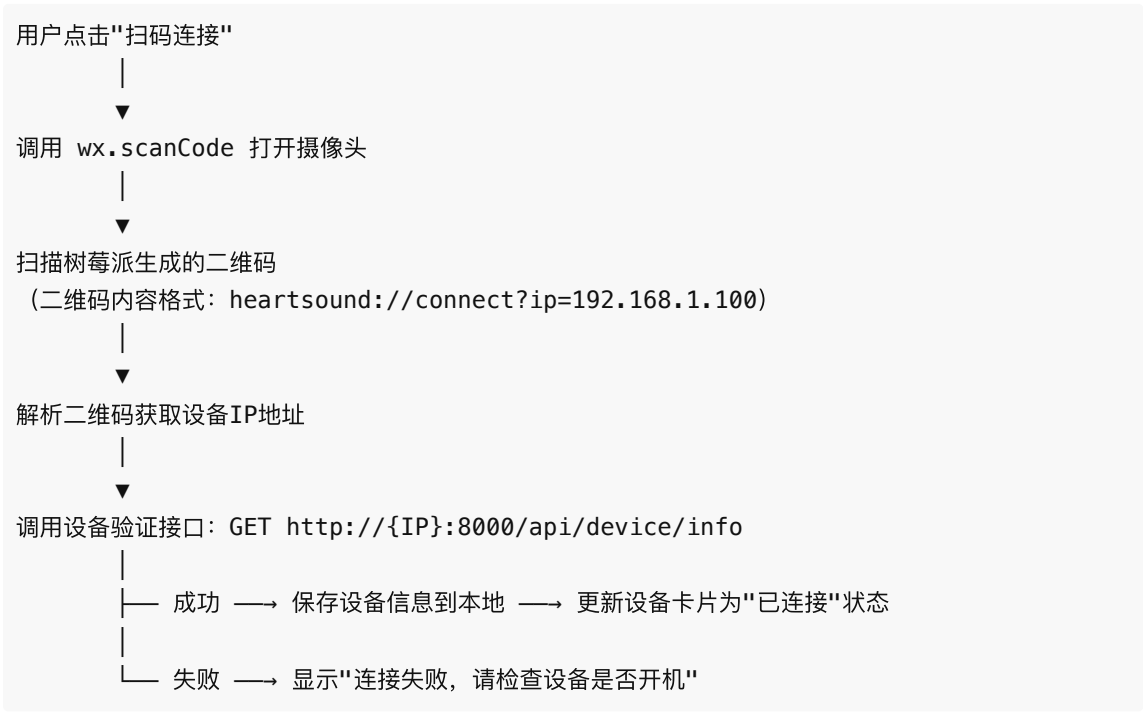


1.2 首页布局



1.3 扫码连接功能

功能流程



二维码规格

项目	规格
二维码内容格式	`heartsound://connect?ip={设备IP地址}`

二维码示例	`heartsound://connect?ip=192.168.1.100`
生成方式	树莓派启动时自动生成并显示在LCD屏幕上
刷新机制	IP变化时自动更新二维码

页面交互

状态	界面表现
扫码中	相机预览界面，顶部提示"请扫描设备上的二维码"
解析中	显示loading，提示"正在连接设备..."
连接成功	Toast提示"连接成功"，返回首页更新设备卡片
连接失败	弹窗提示错误原因，提供"重试"按钮
无效二维码	Toast提示"无效的设备二维码"

1.4 手动输入连接功能

功能流程

```
用户点击"手动输入"
        │
        ▼
跳转到连接页面，显示IP输入框
        │
        ▼
用户输入树莓派IP地址（如：192.168.1.100）
        │
        ▼
点击"连接"按钮
        │
        ▼
调用设备验证接口：GET http://{IP}:8000/api/device/info
        │
        ├── 成功 ──→ 保存设备信息 ──→ 返回首页，更新设备卡片
        │
        └── 失败 ──→ 显示"连接失败，请检查IP地址是否正确"
```

连接页面布局

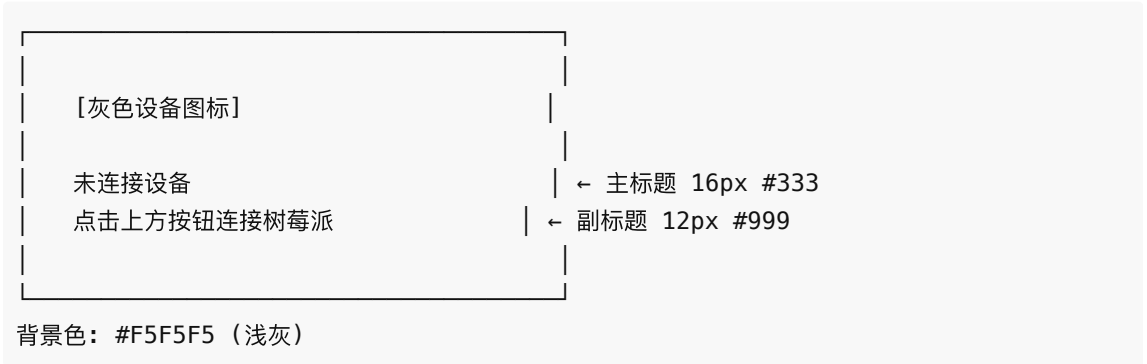
```
┌──────────────────────────────────────┐
│            状态栏（44px）              │
├──────────────────────────────────────┤
│  [返回]      手动连接设备              │
├──────────────────────────────────────┤
│                                      │
```

输入验证

验证项	规则	错误提示
格式检查	符合IP地址格式 (x.x.x.x)	"请输入有效的IP地址格式"
非空检查	不能为空	"请输入IP地址"
连接超时	3秒无响应	"连接超时，请检查网络"

1.5 设备状态卡片

未连接状态



已连接状态

```
┌──────────────────────────────────────┐
│                                      │
│   [绿色设备图标] ● 已连接              │ ← 绿色圆点表示在线
│                                      │
│   心音智鉴设备                         │ ← 设备名称 16px #333
│   IP: 192.168.1.100                  │ ← 设备IP 12px #666
│   信号强度: 良好                       │ ← 信号状态 12px #52C41A
│                                      │
└──────────────────────────────────────┘
背景色: #E8F5E9 (浅绿)
边框: 1px solid #4CAF50
```

连接断开状态

```
┌──────────────────────────────────────┐
│                                      │
│   [红色设备图标] ● 连接断开            │ ← 红色圆点
│                                      │
│   心音智鉴设备                         │
│   IP: 192.168.1.100                  │
│   点击重新连接                         │ ← 可点击重连
│                                      │
└──────────────────────────────────────┘
背景色: #FFEBEE (浅红)
边框: 1px solid #F44336
```

1.6 设备连接API

设备信息接口

```
GET http://{IP}:8000/api/device/info

响应示例:
{
  "device_id": "RPi-HS-001",
  "device_name": "心音智鉴设备",
  "status": "ready",
  "ip_address": "192.168.1.100",
  "firmware_version": "1.0.0",
  "model_version": "v2.1"
}
```

连接状态检查（心跳）

```
GET http://{IP}:8000/api/device/ping

响应: { "status": "ok" }
```

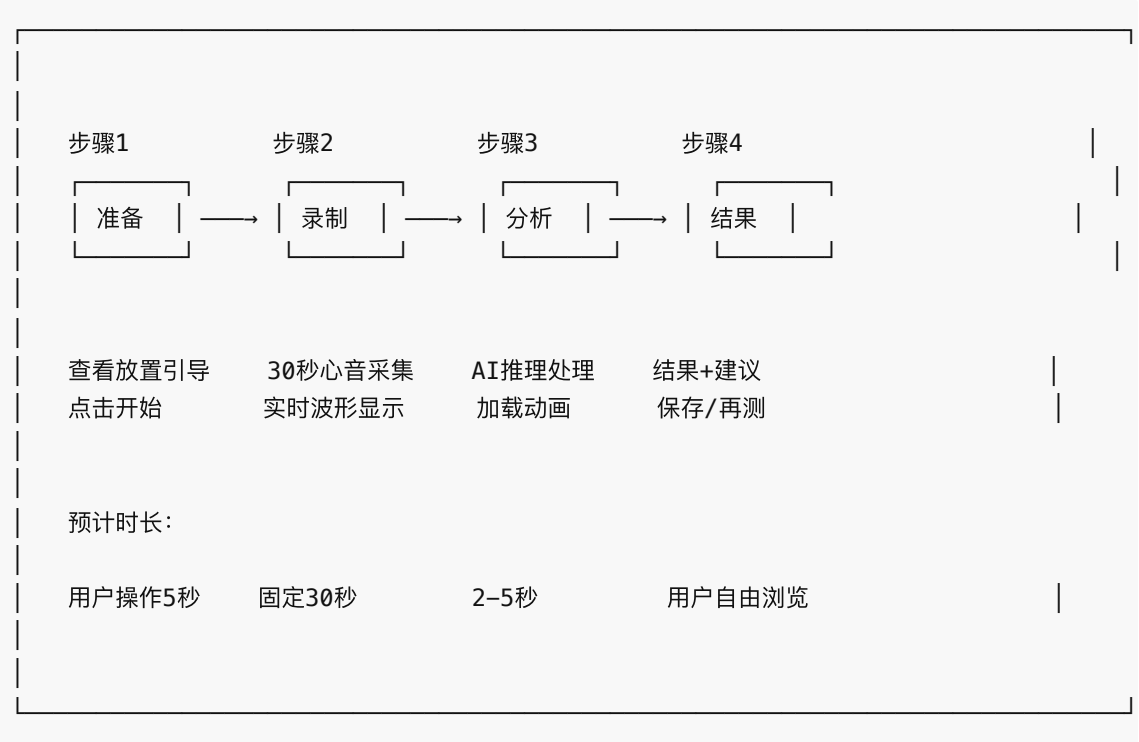
```
检查频率：每10秒检查一次
超时判定：连续3次无响应则判定断连
```

1.7 功能验收标准

验收项	标准
AC-C01	点击"扫码连接"后1秒内打开摄像头
AC-C02	扫描有效二维码后3秒内完成连接验证
AC-C03	手动输入IP后点击连接，3秒内返回结果
AC-C04	设备断连后10秒内更新卡片状态
AC-C05	连接成功后"开始检测"按钮变为可用状态
AC-C06	无效IP或二维码给出明确错误提示

二、检测中心模块 - 检测流程总览

1.1 四步标准流程



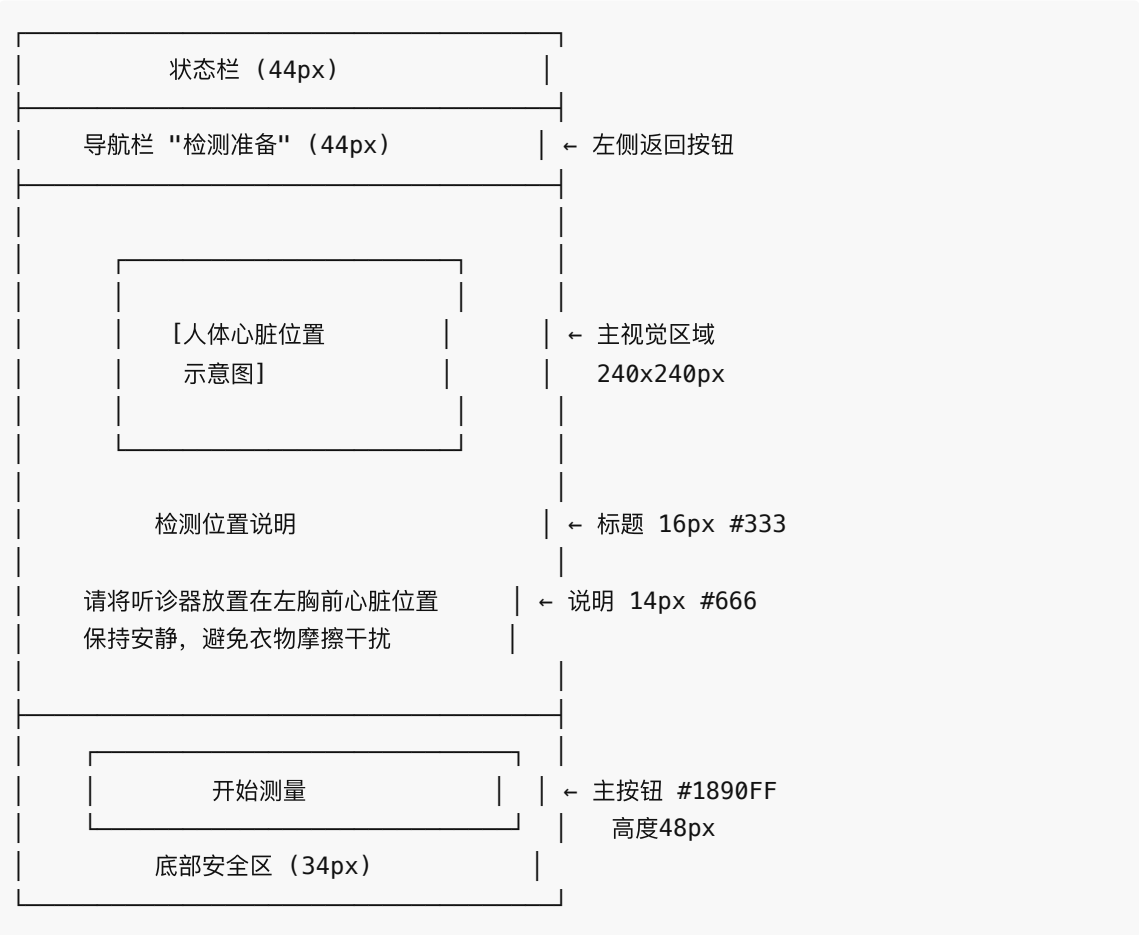
1.2 核心用户故事（产品经理）

阶段	核心用户故事
准备	作为家庭用户，我希望看到清晰的听诊器放置示意图，以便正确放置设备

录制	作为家庭用户，我希望看到实时波形和倒计时，以便知道设备正在正常工作
分析	作为家庭用户，我希望看到加载动画，以便知道系统正在处理我的数据
结果	作为家庭用户，我希望第一眼就看到正常/异常结果，并获得通俗的健康建议

二、检测准备页

2.1 页面布局（UI设计师）



2.2 检测引导文案（领域专家）

标题: 准备开始检测

说明文案:

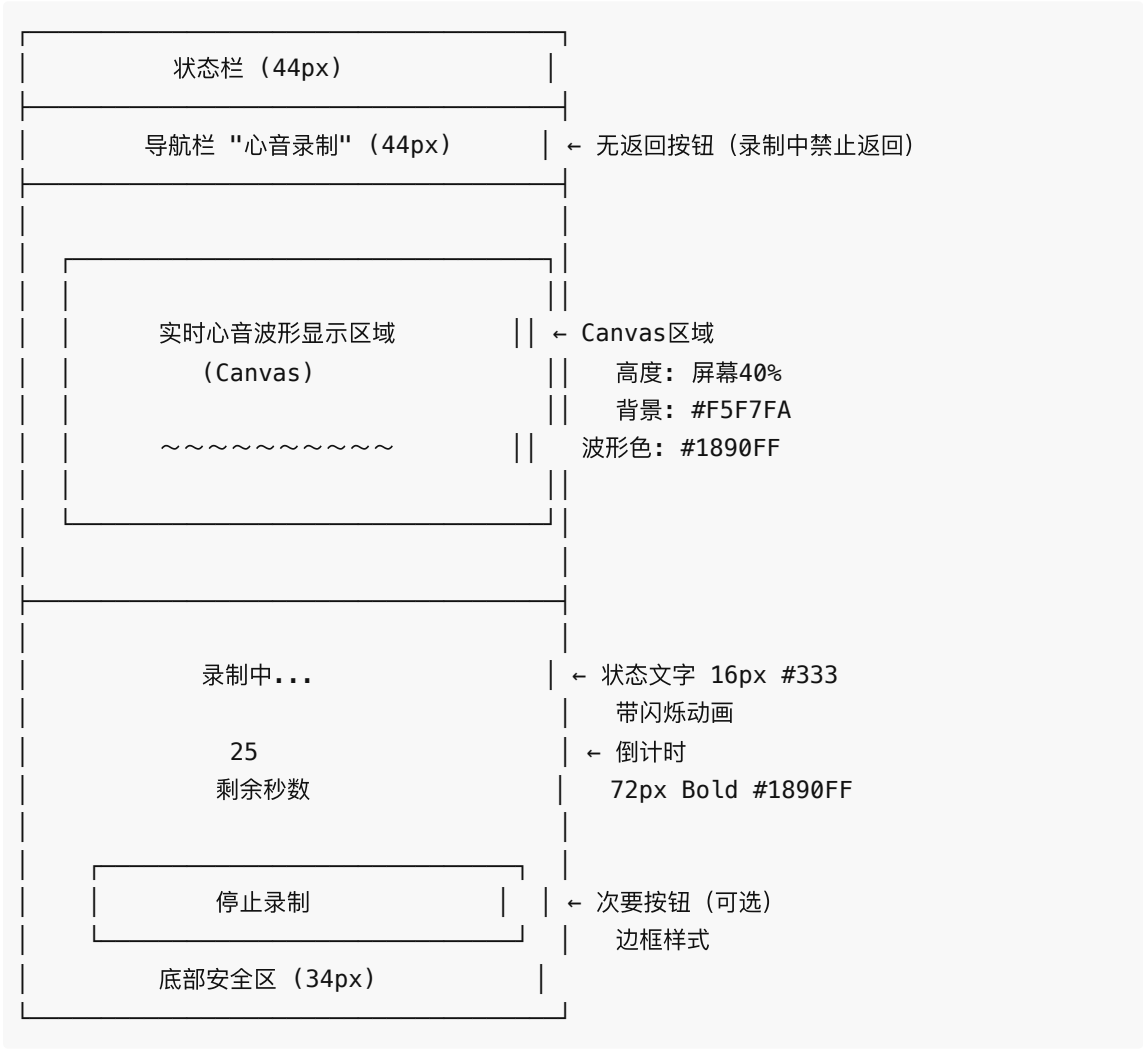
> *请按照以下步骤准备：*
>
> 1. *找一个安静的环境，避免周围有噪音干扰*
> 2. *坐下或躺下，保持身体放松*
> 3. *将听诊器头部（圆形金属面）贴在左胸前心脏位置*
> 4. *确保听诊器与皮肤紧密贴合，没有衣物阻隔*

2.3 功能验收标准（产品经理）

验收项	标准
AC-P01	页面加载后3秒内完整显示示意图
AC-P02	示意图清晰展示心脏位置，分辨率不低于750x750
AC-P03	文字说明不超过2行，使用12号以上字体
AC-P04	点击"开始测量"后0.5秒内跳转录制页

三、录制页面

3.1 页面布局（UI设计师）



3.2 波形显示规格（UI设计师）

规格项	数值

刷新率	30fps
波形颜色	#1890FF（品牌蓝）
波形线宽	2px
背景色	#F5F7FA
中心线	1px虚线 #DDD
振幅范围	中心线上下各40%区域高度

3.3 检测过程提示（领域专家）

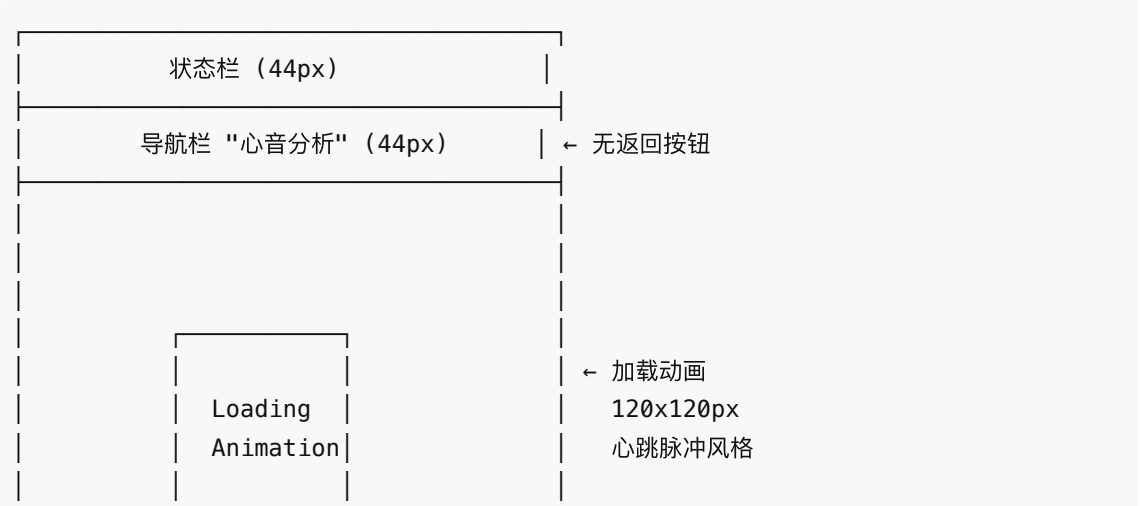
时机	提示文案
录制开始	检测已开始，请保持安静和放松
录制进行中	正在采集心音，请勿移动听诊器
剩余5秒	即将完成，请继续保持

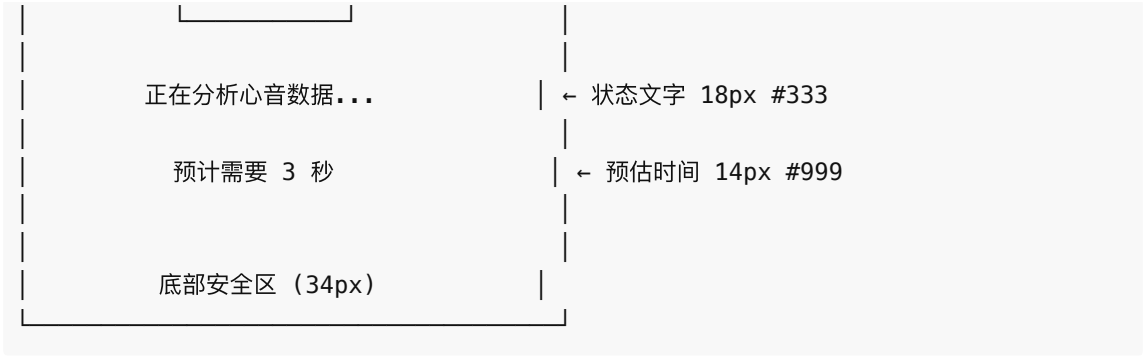
3.4 功能验收标准（产品经理）

验收项	标准
AC-R01	进入页面后2秒内建立WebSocket连接
AC-R02	波形刷新率不低于24fps，无明显卡顿
AC-R03	倒计时从30开始，每秒递减，准确无误差
AC-R05	30秒结束后自动跳转分析页

四、分析中页面

4.1 页面布局（UI设计师）





4.2 加载动画方案（UI设计师）

推荐方案: 心跳脉冲动画

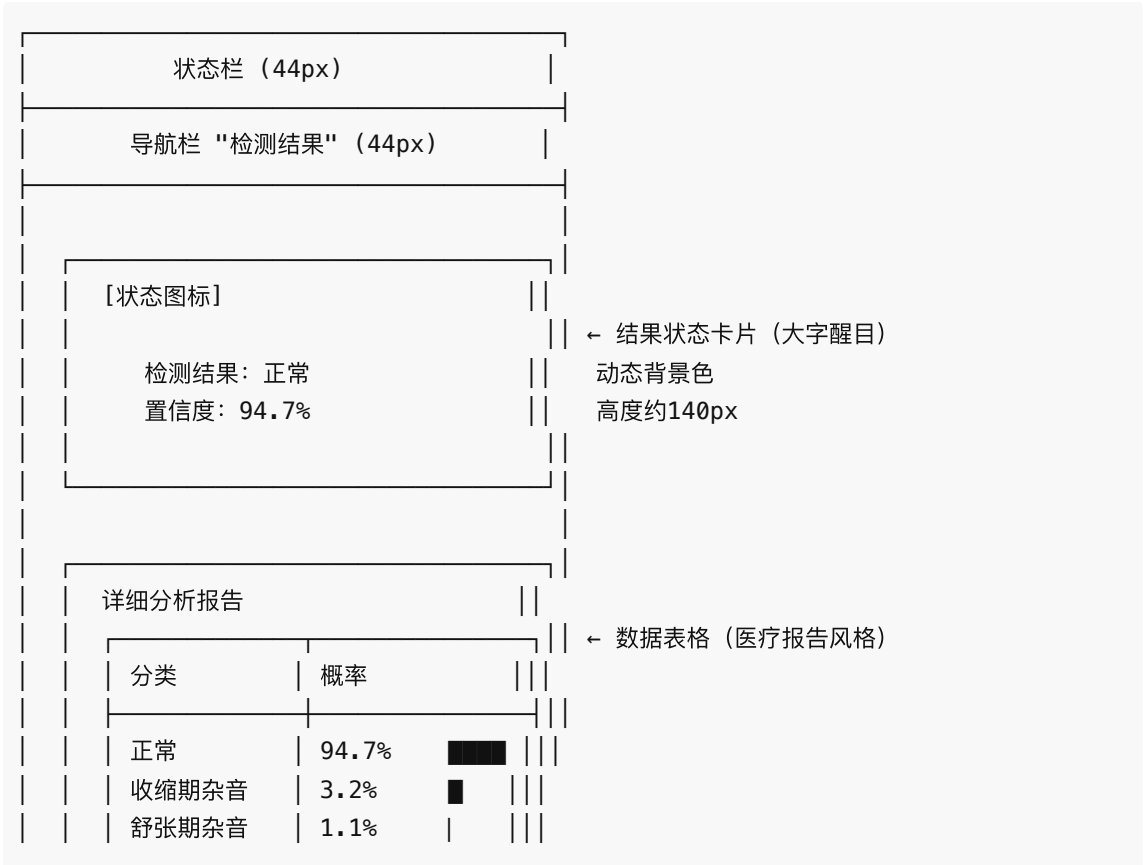
- 心形图标缩放动画，周期800ms
- 底部配合波形线条流动效果
- 使用Lottie实现，尺寸120x120px

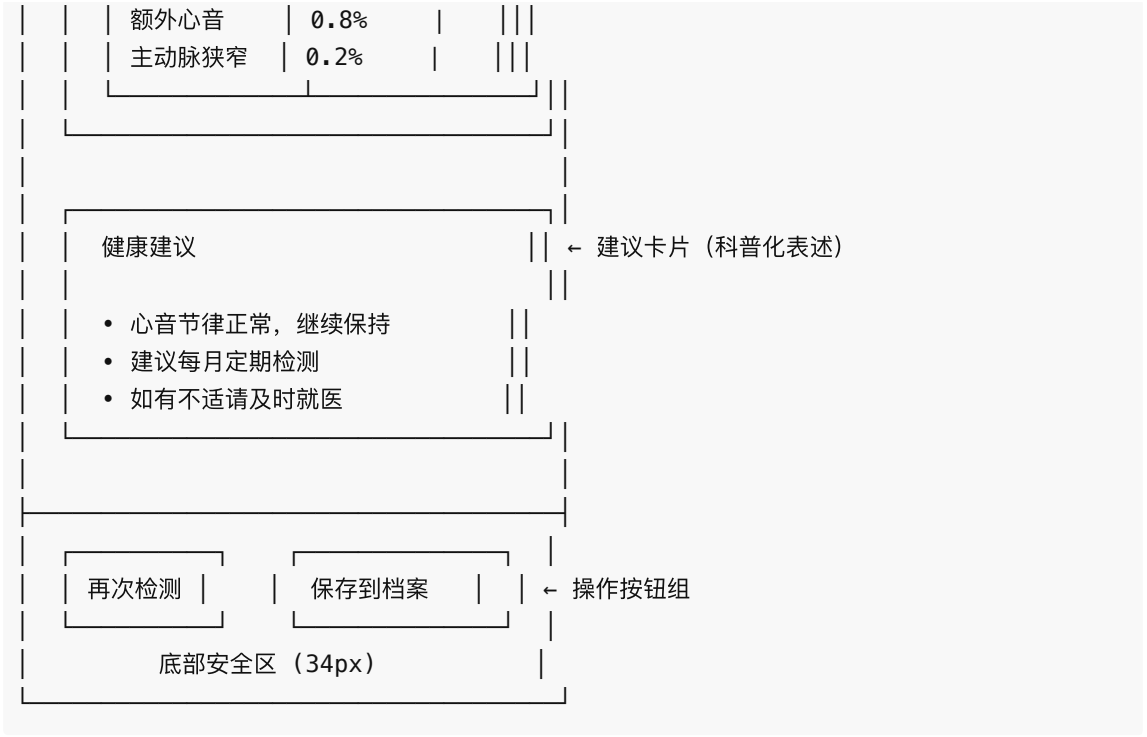
4.3 分析中提示（领域专家）

> *正在分析您的心音数据，请稍候...*

五、结果展示页

5.1 页面布局（UI设计师 - 分区融合设计）





5.2 三种风险状态视觉设计（UI设计师）

风险等级	主色	背景渐变	图标	文字色
安全 (Safe)	#4CAF50	#E8F5E9 → #C8E6C9	勾选圆形	#2E7D32
中等 (Warning)	#FF9800	#FFF8E1 → #FFECB3	警告三角	#E65100
高风险 (Danger)	#F44336	#FFEBEE → #FFCDD2	感叹圆形	#C62828

5.3 科普化术语对照（领域专家）

专业分类	用户展示文案
正常 (Normal)	心脏节律正常
收缩期杂音 (Systolic Murmur)	心脏收缩时有轻微异常声音
舒张期杂音 (Diastolic Murmur)	心脏舒张时有异常声音
额外心音 (Extra Heart Sound)	检测到额外的心跳声音
主动脉瓣狭窄 (Aortic Stenosis)	心脏瓣膜可能存在问题

5.4 分级健康建议示例（领域专家）

安全级（绿色）- 正常心音

> *恭喜您，本次检测显示心音正常！*
>
> - *您的心脏跳动节奏规律，没有检测到明显异常*
> - *建议保持良好的生活习惯，定期进行健康检测*

> - *适度运动、均衡饮食、充足睡眠有助于心脏健康*

行动建议: 建议每月进行1-2次自我检测

中等风险（黄色）- 收缩期杂音

> *本次检测到心脏收缩时有轻微的额外声音。*
>
> - *这种情况在健康人群中也可能出现，不必过于担心*
> - *情绪紧张、运动后、发热时都可能影响检测结果*
> - *建议您在安静状态下重新检测一次*

行动建议: 如果多次检测均显示异常，建议近期咨询医生

高风险（红色）- 主动脉瓣狭窄

> *本次检测发现心脏瓣膜区域存在明显异常特征。*
>
> - *检测结果提示可能存在主动脉瓣狭窄*
> - *这是一种需要医学关注的心脏瓣膜问题*
> - *请尽快前往医院心内科进行专业诊断*

行动建议: 建议尽快（一周内）前往三甲医院心内科就诊

5.5 免责声明（领域专家）

结果页底部简洁版:

> *本检测结果仅供健康参考，不能替代医生诊断。如有不适，请及时就医。*

六、功能优先级矩阵（产品经理）

Must Have（MVP必须）

功能	页面	说明
听诊器放置示意图	准备页	用户必须知道如何正确放置
开始测量按钮	准备页	启动检测的核心入口
实时波形显示	录制页	视觉反馈确认设备工作正常
倒计时显示	录制页	用户需要知道等待时间
加载动画	分析页	防止用户误以为页面卡死
结果状态大字显示	结果页	最核心的用户诉求
风险颜色标识	结果页	快速传达风险等级
健康建议列表	结果页	指导用户下一步行动
再次检测按钮	结果页	允许用户重新测量

Should Have（应该有）

功能	页面	说明
置信度显示	结果页	增加结果可信度
详细概率表格	结果页	满足想了解更多的用户
保存记录功能	结果页	构建用户健康档案
分析状态文字	分析页	解释系统正在做什么

Won't Have（本期不做）

功能	排除理由
可变录制时长	增加用户决策负担
S1/S2心音标注	普通用户无法理解
视频引导教程	开发成本高

七、设计规范汇总（UI设计师）

7.1 颜色规范

类型	颜色	色值	用途
主色	品牌蓝	#1890FF	按钮、波形、强调
安全色	绿色	#4CAF50	正常结果
警告色	黄色	#FF9800	中等风险
危险色	红色	#F44336	高风险
主文字	深灰	#333333	标题
次文字	中灰	#666666	正文
辅助文字	浅灰	#999999	提示

7.2 字体规范

层级	字号	字重	用途
Display	72px	Bold	倒计时数字
H1	24px	Bold	结果状态
H2	18px	Medium	卡片标题
Body	14px	Regular	正文说明

Caption	12px	Regular	辅助标签

7.3 间距系统

等级	数值	用途
xs	4px	图标与文字
sm	8px	紧凑元素
md	16px	标准间距、卡片内边距
lg	24px	区块间距
xl	32px	页面边距

八、性能指标

指标	目标值
波形渲染帧率	≥ 30fps
页面切换时间	≤ 300ms
WebSocket连接	≤ 2秒
AI推理时间	≤ 5秒
整体流程耗时	≤ 50秒

九、设计交付清单

9.1 页面设计

页面	状态变体
检测准备页	默认态
录制页面	录制中态
分析中页面	加载态
结果展示页	安全/中等/高风险 三种状态

9.2 组件设计

组件	状态变体
主按钮	default/pressed/disabled/loading
次要按钮	default/pressed/disabled

结果卡片	三种风险色
数据表格	表头/奇数行/偶数行
状态图标	对勾/警告/感叹号

9.3 动画资源

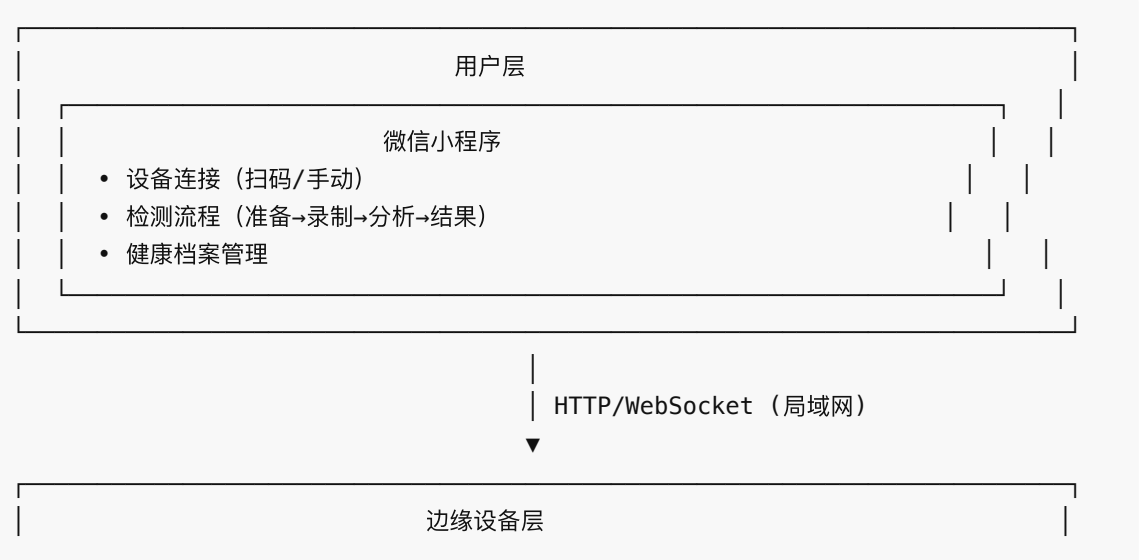
动画	格式	用途
心跳脉冲动画	Lottie JSON	分析中页面
波形绘制	Canvas代码	录制页面
状态文字闪烁	CSS动画	录制页面

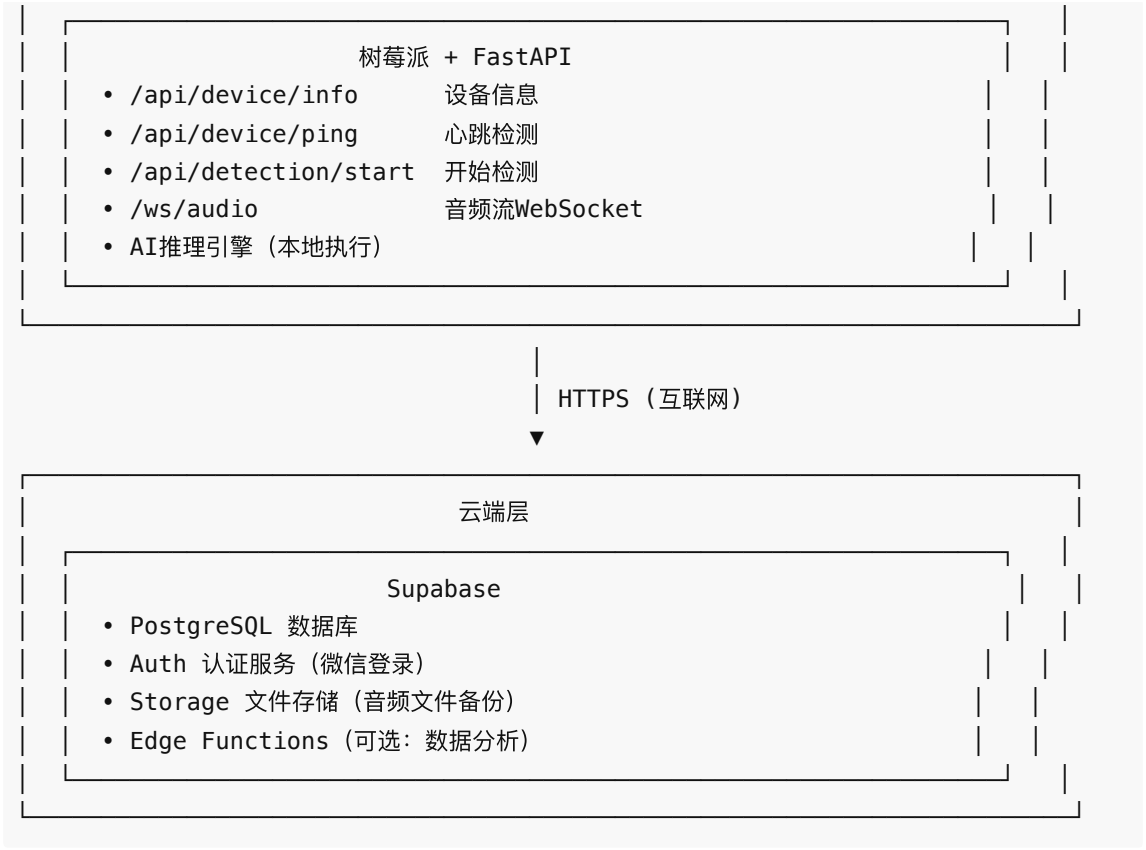
十、技术架构设计

10.1 技术栈概览

层级	技术选型	说明
前端	微信小程序原生	WXML + WXSS + JavaScript
后端（树莓派）	FastAPI + Python 3.11	心音采集、AI推理、WebSocket
云端数据库	Supabase (PostgreSQL)	用户数据、检测记录、健康档案
认证服务	Supabase Auth	微信登录集成
实时通信	WebSocket	心音波形实时传输
AI模型	PyTorch / ONNX	心音分类模型

10.2 系统架构图





十一、数据库设计（Supabase）

11.1 数据库表结构

users 用户表

```sql
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    openid VARCHAR(64) UNIQUE NOT NULL,          -- 微信openid
    nickname VARCHAR(50),                        -- 微信昵称
    avatar_url TEXT,                             -- 头像URL
    phone VARCHAR(20),                           -- 手机号（可选）
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- 索引
CREATE INDEX idx_users_openid ON users(openid);
```

devices 设备表

```sql
CREATE TABLE devices (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    device_id VARCHAR(50) UNIQUE NOT NULL,        -- 设备唯一标识 RPi-HS-001
    device_name VARCHAR(100) DEFAULT '心音智鉴设备',
    firmware_version VARCHAR(20),                  -- 固件版本
    model_version VARCHAR(20),                     -- AI模型版本
    last_ip VARCHAR(45),                           -- 最后连接IP
    last_seen_at TIMESTAMPTZ,                      -- 最后在线时间
    created_at TIMESTAMPTZ DEFAULT NOW()
);
```

user_devices 用户设备绑定表

```
CREATE TABLE user_devices (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    device_id UUID REFERENCES devices(id) ON DELETE CASCADE,
    bindAt TIMESTAMPTZ DEFAULT NOW(),
    is_primary BOOLEAN DEFAULT FALSE,             -- 是否为主设备
    UNIQUE(user_id, device_id)
);

-- 索引
CREATE INDEX idx_user_devices_user ON user_devices(user_id);
```

detection_records 检测记录表

```
CREATE TABLE detection_records (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    device_id UUID REFERENCES devices(id),

    -- 检测结果
    result_category VARCHAR(50) NOT NULL,         -- 'normal',
'systolic_murmur', etc.
    result_label VARCHAR(100) NOT NULL,           -- 显示文案: '心脏节律正常'
    confidence DECIMAL(5,2) NOT NULL,             -- 置信度 0-100
    risk_level VARCHAR(20) NOT NULL,              -- 'safe', 'warning',
'danger'

    -- 详细概率分布 (JSONB)
    probabilities JSONB NOT NULL,
    /* 示例:
    {
        "normal": 94.7,
        "systolic_murmur": 3.2,
        "diastolic_murmur": 1.1,
        "extra_heart_sound": 0.8,
```

```
        "aortic_stenosis": 0.2
    }
    */

    -- 健康建议（JSONB）
    health_advice JSONB,
    /* 示例:
    {
        "summary": "心音正常，继续保持良好生活习惯",
        "suggestions": ["每月定期检测", "适度运动"],
        "action": "建议每月进行1-2次自我检测"
    }
    */

    -- 元数据
    duration_seconds INTEGER DEFAULT 30,          -- 录制时长
    audio_file_path TEXT,                         -- 音频文件路径（Supabase
Storage）

    created_at TIMESTAMPTZ DEFAULT NOW(),

    -- 索引
    CONSTRAINT valid_risk_level CHECK (risk_level IN ('safe', 'warning',
'danger'))
);

-- 索引
CREATE INDEX idx_records_user ON detection_records(user_id);
CREATE INDEX idx_records_created ON detection_records(created_at DESC);
CREATE INDEX idx_records_risk ON detection_records(risk_level);
```

health_tips 健康小贴士表

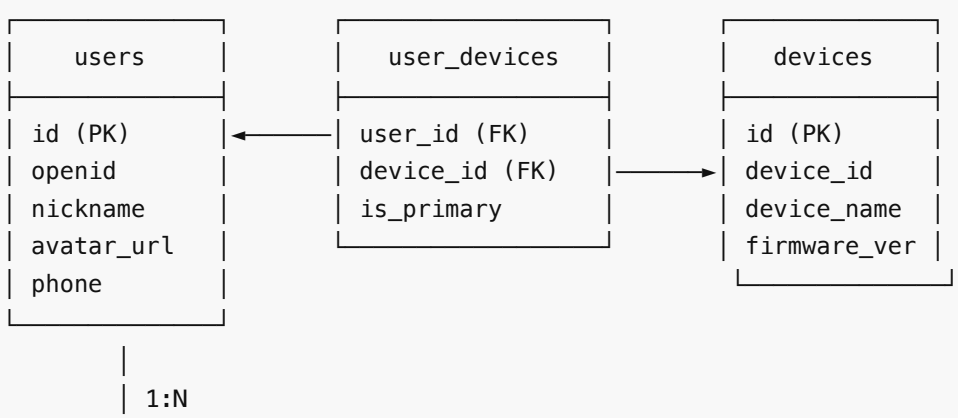
```
CREATE TABLE health_tips (
    id SERIAL PRIMARY KEY,
    content TEXT NOT NULL,                        -- 贴士内容
    category VARCHAR(50) DEFAULT 'general',       -- 分类
    is_active BOOLEAN DEFAULT TRUE,
    display_order INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- 预置数据
INSERT INTO health_tips (content, category) VALUES
('成年人正常心率为60-100次/分钟', 'heart_rate'),
('保持规律作息有助于心脏健康', 'lifestyle'),
```

```
('适度运动可以增强心肺功能', 'exercise'),
('戒烟限酒对心血管健康至关重要', 'lifestyle');
```

11.2 Row Level Security (RLS) 策略

```sql
-- 启用RLS
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
ALTER TABLE detection_records ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_devices ENABLE ROW LEVEL SECURITY;

-- 用户只能访问自己的数据
CREATE POLICY "Users can view own profile"
    ON users FOR SELECT
    USING (auth.uid() = id);

CREATE POLICY "Users can update own profile"
    ON users FOR UPDATE
    USING (auth.uid() = id);

-- 检测记录：用户只能访问自己的记录
CREATE POLICY "Users can view own records"
    ON detection_records FOR SELECT
    USING (auth.uid() = user_id);

CREATE POLICY "Users can insert own records"
    ON detection_records FOR INSERT
    WITH CHECK (auth.uid() = user_id);

-- 健康小贴士：所有人可读
CREATE POLICY "Health tips are public"
    ON health_tips FOR SELECT
    USING (is_active = TRUE);
```

11.3 数据库关系图



```
        ▼
┌──────────────────────┐        ┌──────────────────────┐
│ detection_records    │        │   health_tips        │
├──────────────────────┤        ├──────────────────────┤
│ id (PK)              │        │ id (PK)              │
│ user_id (FK)         │        │ content              │
│ device_id (FK)       │        │ category             │
│ result_category      │        │ is_active            │
│ confidence           │        └──────────────────────┘
│ risk_level           │
│ probabilities        │
│ health_advice        │
└──────────────────────┘
```

十二、API 详细设计

12.1 树莓派端 API（FastAPI）

设备信息接口

```
GET /api/device/info

Response 200:
{
    "device_id": "RPi-HS-001",
    "device_name": "心音智鉴设备",
    "status": "ready",                    // ready | recording | analyzing
    "ip_address": "192.168.1.100",
    "firmware_version": "1.0.0",
    "model_version": "v2.1",
    "uptime_seconds": 3600
}

Response 503 (设备忙):
{
    "error": "device_busy",
    "message": "设备正在录制中，请稍后重试"
}
```

心跳检测接口

```
GET /api/device/ping

Response 200:
{
    "status": "ok",
```

```
    "timestamp": "2024-01-15T10:30:00Z"
}
```

开始检测接口

```
POST /api/detection/start

Request:
{
    "user_id": "uuid-string",            // 可选，用于记录关联
    "duration": 30                        // 录制时长（秒），默认30
}

Response 200:
{
    "session_id": "sess_abc123",
    "websocket_url": "ws://192.168.1.100:8000/ws/audio/sess_abc123",
    "duration": 30,
    "started_at": "2024-01-15T10:30:00Z"
}

Response 400:
{
    "error": "invalid_duration",
    "message": "录制时长必须在10-60秒之间"
}
```

获取检测结果接口

```
GET /api/detection/{session_id}/result

Response 200 (分析完成):
{
    "session_id": "sess_abc123",
    "status": "completed",
    "result": {
        "category": "normal",
        "label": "心脏节律正常",
        "confidence": 94.7,
        "risk_level": "safe",
        "probabilities": {
            "normal": 94.7,
            "systolic_murmur": 3.2,
            "diastolic_murmur": 1.1,
            "extra_heart_sound": 0.8,
            "aortic_stenosis": 0.2
        },
```

```
        "health_advice": {
            "summary": "恭喜您，本次检测显示心音正常！",
            "suggestions": [
                "您的心脏跳动节奏规律，没有检测到明显异常",
                "建议保持良好的生活习惯，定期进行健康检测",
                "适度运动、均衡饮食、充足睡眠有助于心脏健康"
            ],
            "action": "建议每月进行1-2次自我检测"
        }
    },
    "duration_seconds": 30,
    "analyzed_at": "2024-01-15T10:30:35Z"
}

Response 202 (分析中):
{
    "session_id": "sess_abc123",
    "status": "analyzing",
    "progress": 60,
    "message": "正在分析心音数据..."
}

Response 404:
{
    "error": "session_not_found",
    "message": "检测会话不存在"
}
```

12.2 WebSocket 音频流协议

```
WebSocket: ws://{IP}:8000/ws/audio/{session_id}

# 连接建立后，服务端推送音频数据帧

# 音频数据帧格式 (Binary)
{
    "type": "audio_frame",
    "timestamp": 1234567890,
    "data": [0.1, 0.2, -0.1, ...]    // 采样点数组，用于波形绘制
}

# 状态消息 (Text/JSON)
{
    "type": "status",
    "status": "recording",           // recording | completed | error
    "remaining_seconds": 25,
    "message": "正在采集心音..."
```

```
}

# 录制完成消息
{
    "type": "recording_complete",
    "session_id": "sess_abc123",
    "duration": 30,
    "message": "录制完成，开始分析..."
}

# 分析完成消息
{
    "type": "analysis_complete",
    "session_id": "sess_abc123",
    "result_url": "/api/detection/sess_abc123/result"
}

# 错误消息
{
    "type": "error",
    "code": "microphone_error",
    "message": "麦克风连接异常，请检查设备"
}
```

12.3 小程序端 Supabase API 调用

用户登录/注册

```
// 微信登录后，使用openid注册/登录
const { data, error } = await supabase
    .from('users')
    .upsert({
        openid: wxOpenid,
        nickname: userInfo.nickName,
        avatar_url: userInfo.avatarUrl
    }, {
        onConflict: 'openid'
    })
    .select()
    .single();
```

保存检测记录

```
const { data, error } = await supabase
    .from('detection_records')
    .insert({
```

```
        user_id: currentUser.id,
        device_id: connectedDevice.id,
        result_category: result.category,
        result_label: result.label,
        confidence: result.confidence,
        risk_level: result.risk_level,
        probabilities: result.probabilities,
        health_advice: result.health_advice,
        duration_seconds: 30
    })
    .select()
    .single();
```

获取历史记录

```
const { data, error } = await supabase
    .from('detection_records')
    .select('*')
    .eq('user_id', currentUser.id)
    .order('created_at', { ascending: false })
    .limit(20);
```

获取健康小贴士

```
const { data, error } = await supabase
    .from('health_tips')
    .select('content')
    .eq('is_active', true)
    .order('display_order');
```

十三、代码结构设计

13.1 微信小程序目录结构

```
miniprogram/
├── app.js                          # 小程序入口
├── app.json                        # 全局配置
├── app.wxss                        # 全局样式
├── sitemap.json
│
├── config/
│   └── supabase.js                 # Supabase 配置
│
├── utils/
│   ├── supabase.js                 # Supabase 客户端封装
```

```
│   ├── device.js                    # 设备连接工具
│   ├── websocket.js                 # WebSocket 管理
│   ├── storage.js                   # 本地存储工具
│   └── util.js                      # 通用工具函数
│
├── services/
│   ├── user.js                      # 用户服务（登录、信息）
│   ├── device.js                    # 设备服务（连接、心跳）
│   ├── detection.js                 # 检测服务（开始、结果）
│   └── record.js                    # 记录服务（保存、查询）
│
├── components/
│   ├── device-card/                 # 设备状态卡片组件
│   ├── waveform/                    # 波形显示组件
│   ├── result-card/                 # 结果卡片组件
│   ├── health-tip/                  # 健康小贴士组件
│   └── loading-heart/               # 心跳加载动画组件
│
├── pages/
│   ├── index/                       # 首页（设备连接）
│   │   ├── index.js
│   │   ├── index.json
│   │   ├── index.wxml
│   │   └── index.wxss
│   │
│   ├── connect/                     # 手动连接页
│   │   └── ...
│   │
│   ├── detection/
│   │   ├── prepare/                 # 检测准备页
│   │   ├── recording/               # 录制页
│   │   ├── analyzing/               # 分析中页
│   │   └── result/                  # 结果页
│   │
│   ├── records/                     # 历史记录页
│   │   └── ...
│   │
│   └── profile/                     # 个人中心
│       └── ...
│
└── assets/
    ├── images/                      # 图片资源
    ├── icons/                       # 图标
    └── lottie/                      # Lottie 动画文件
```

13.2 树莓派 FastAPI 目录结构

```
raspberry_pi/
├── main.py                          # FastAPI 入口
├── requirements.txt                 # Python 依赖
├── config.py                        # 配置文件
│
├── api/
│   ├── __init__.py
│   ├── device.py                    # 设备相关接口
│   ├── detection.py                 # 检测相关接口
│   └── websocket.py                 # WebSocket 处理
│
├── core/
│   ├── __init__.py
│   ├── audio.py                     # 音频采集模块
│   ├── inference.py                 # AI 推理模块
│   └── qrcode.py                    # 二维码生成
│
├── models/
│   ├── __init__.py
│   ├── schemas.py                   # Pydantic 模型定义
│   └── heart_sound_model.onnx       # AI 模型文件
│
├── utils/
│   ├── __init__.py
│   ├── audio_utils.py               # 音频处理工具
│   └── network.py                   # 网络工具
│
├── tests/
│   ├── test_device.py
│   ├── test_detection.py
│   └── test_audio.py
│
└── scripts/
    ├── start.sh                     # 启动脚本
    └── generate_qrcode.py           # 二维码生成脚本
```

13.3 关键代码示例

Supabase 客户端初始化 (小程序)

```javascript
// config/supabase.js
export const SUPABASE_URL = 'https://your-project.supabase.co';
export const SUPABASE_ANON_KEY = 'your-anon-key';

// utils/supabase.js
import { createClient } from '@supabase/supabase-js';
import { SUPABASE_URL, SUPABASE_ANON_KEY } from '../config/supabase';
```

```
// 微信小程序需要使用自定义 fetch
const supabase = createClient(SUPABASE_URL, SUPABASE_ANON_KEY, {
    auth: {
        persistSession: true,
        storage: {
            getItem: (key) => wx.getStorageSync(key),
            setItem: (key, value) => wx.setStorageSync(key, value),
            removeItem: (key) => wx.removeStorageSync(key)
        }
    },
    global: {
        fetch: (url, options) => {
            return new Promise((resolve, reject) => {
                wx.request({
                    url,
                    method: options.method || 'GET',
                    data: options.body,
                    header: options.headers,
                    success: (res) => resolve({
                        ok: res.statusCode >= 200 && res.statusCode < 300,
                        status: res.statusCode,
                        json: () => Promise.resolve(res.data)
                    }),
                    fail: reject
                });
            });
        }
    }
});

export default supabase;
```

FastAPI 检测接口 (树莓派)

```
# api/detection.py
from fastapi import APIRouter, HTTPException, BackgroundTasks
from pydantic import BaseModel
import uuid
from core.audio import AudioRecorder
from core.inference import HeartSoundClassifier

router = APIRouter(prefix="/api/detection", tags=["detection"])

class StartDetectionRequest(BaseModel):
    user_id: str | None = None
```

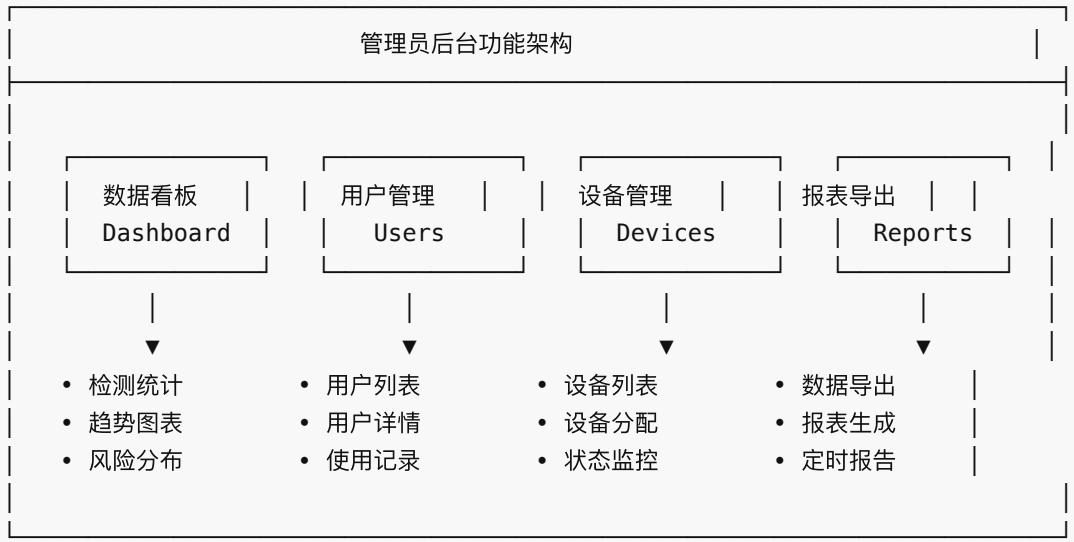
```
    duration: int = 30

class StartDetectionResponse(BaseModel):
    session_id: str
    websocket_url: str
    duration: int
    started_at: str

# 存储活跃的检测会话
active_sessions = {}

@router.post("/start", response_model=StartDetectionResponse)
async def start_detection(
    request: StartDetectionRequest,
    background_tasks: BackgroundTasks
):
    if not 10 <= request.duration <= 60:
        raise HTTPException(400, "录制时长必须在10-60秒之间")

    session_id = f"sess_{uuid.uuid4().hex[:12]}"

    # 创建会话
    active_sessions[session_id] = {
        "status": "recording",
        "user_id": request.user_id,
        "duration": request.duration
    }

    # 后台启动录制
    background_tasks.add_task(
        run_detection,
        session_id,
        request.duration
    )

    return StartDetectionResponse(
        session_id=session_id,
        websocket_url=f"ws://{get_local_ip()}:8000/ws/audio/{session_id}",
        duration=request.duration,
        started_at=datetime.now().isoformat()
    )

async def run_detection(session_id: str, duration: int):
    """后台执行检测任务"""
    recorder = AudioRecorder()
    classifier = HeartSoundClassifier()
```

```
    # 录制音频
    audio_data = await recorder.record(duration)

    # 更新状态
    active_sessions[session_id]["status"] = "analyzing"

    # AI 推理
    result = classifier.predict(audio_data)

    # 保存结果
    active_sessions[session_id]["status"] = "completed"
    active_sessions[session_id]["result"] = result
```

十四、管理员后台模块

14.1 功能概述

管理员后台作为小程序的独立模块，供超级管理员进行数据管理、设备分配和报表导出。



14.2 管理员认证

管理员表设计

```
CREATE TABLE admins (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    openid VARCHAR(64) UNIQUE NOT NULL,          -- 微信openid
    nickname VARCHAR(50),
    avatar_url TEXT,
    role VARCHAR(20) DEFAULT 'super_admin',      -- super_admin
    permissions JSONB DEFAULT '["*"]',           -- 权限列表，*表示全部
    is_active BOOLEAN DEFAULT TRUE,
```

```
    created_at TIMESTAMPTZ DEFAULT NOW(),
    last_login_at TIMESTAMPTZ
);

-- 操作日志表
CREATE TABLE admin_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    admin_id UUID REFERENCES admins(id),
    action VARCHAR(100) NOT NULL,                -- 操作类型
    target_type VARCHAR(50),                     -- 操作对象类型
    target_id UUID,                              -- 操作对象ID
    details JSONB,                               -- 操作详情
    ip_address VARCHAR(45),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- 索引
CREATE INDEX idx_admin_logs_admin ON admin_logs(admin_id);
CREATE INDEX idx_admin_logs_created ON admin_logs(created_at DESC);
```

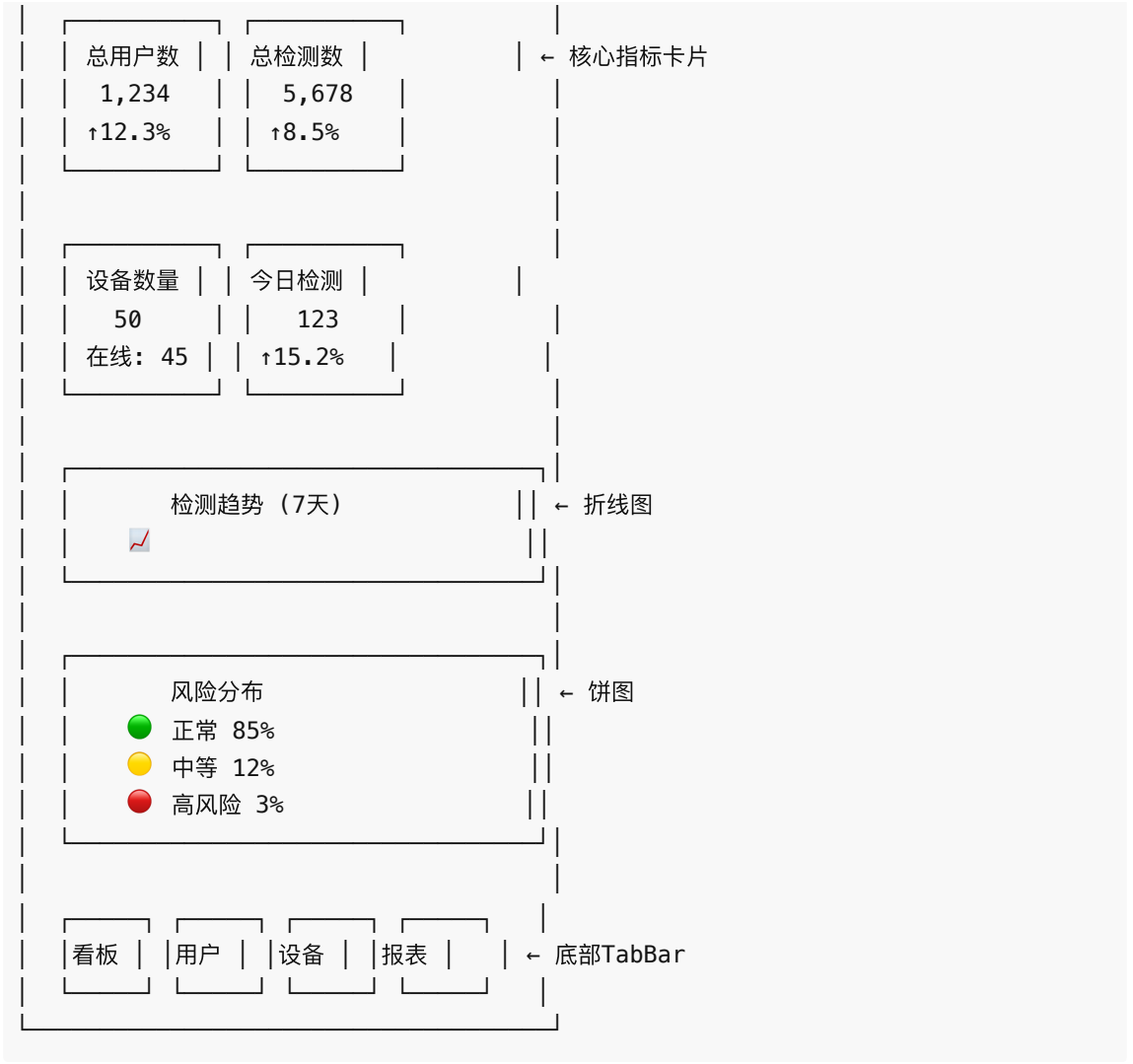
管理员登录流程

```
用户打开管理后台小程序
        │
        ▼
调用 wx.login 获取 code
        │
        ▼
后端验证 code 获取 openid
        │
        ▼
查询 admins 表验证权限
        │
        ├── 存在且 is_active=true ──→ 登录成功，返回 admin_token
        │
        └── 不存在或无效 ──→ 提示"无管理员权限"
```

14.3 数据看板

页面布局

```
┌─────────────────────────────────────┐
│          状态栏 (44px)              │
├─────────────────────────────────────┤
│        导航栏 "数据看板" (44px)      │
├─────────────────────────────────────┤
│                                     │
```



看板数据API

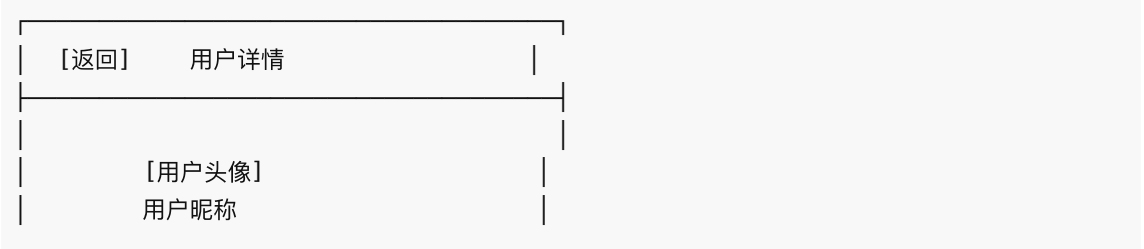
```
GET /api/admin/dashboard/summary

Response 200:
{
    "total_users": 1234,
    "total_detections": 5678,
    "total_devices": 50,
    "online_devices": 45,
    "today_detections": 123,
    "trends": {
        "users_growth": 12.3,        // 较上周增长百分比
        "detections_growth": 8.5,
        "today_growth": 15.2
    },
    "risk_distribution": {
        "safe": 85.2,
```

```
        "warning": 11.8,
        "danger": 3.0
    },
    "weekly_trend": [
        {"date": "2024-01-15", "count": 120},
        {"date": "2024-01-16", "count": 135},
        {"date": "2024-01-17", "count": 128},
        ...
    ]
}
```

14.4 用户管理

用户列表页

用户详情页



```
┌──────────────────────────────────────────┐
│        ID: uuid-xxx                      │
│                                          │
├──────────────────────────────────────────┤
│  基本信息                                │
│  ──────────────────────────              │
│  注册时间    2024-01-10 14:30            │
│  手机号码    138****1234                 │
│  绑定设备    RPi-HS-001                  │
├──────────────────────────────────────────┤
│  检测统计                                │
│  ──────────────────────────              │
│  总检测次数   15                         │
│  正常次数     12 (80%)                   │
│  异常次数     3 (20%)                    │
│  最近检测     2024-01-17                 │
├──────────────────────────────────────────┤
│  检测记录                    [更多]      │
│  ──────────────────────────              │
│  🟢 2024-01-17 正常 94.7%                │
│  🟡 2024-01-15 收缩期杂音 68.2%          │
│  🟢 2024-01-12 正常 91.3%                │
└──────────────────────────────────────────┘
```

用户管理API

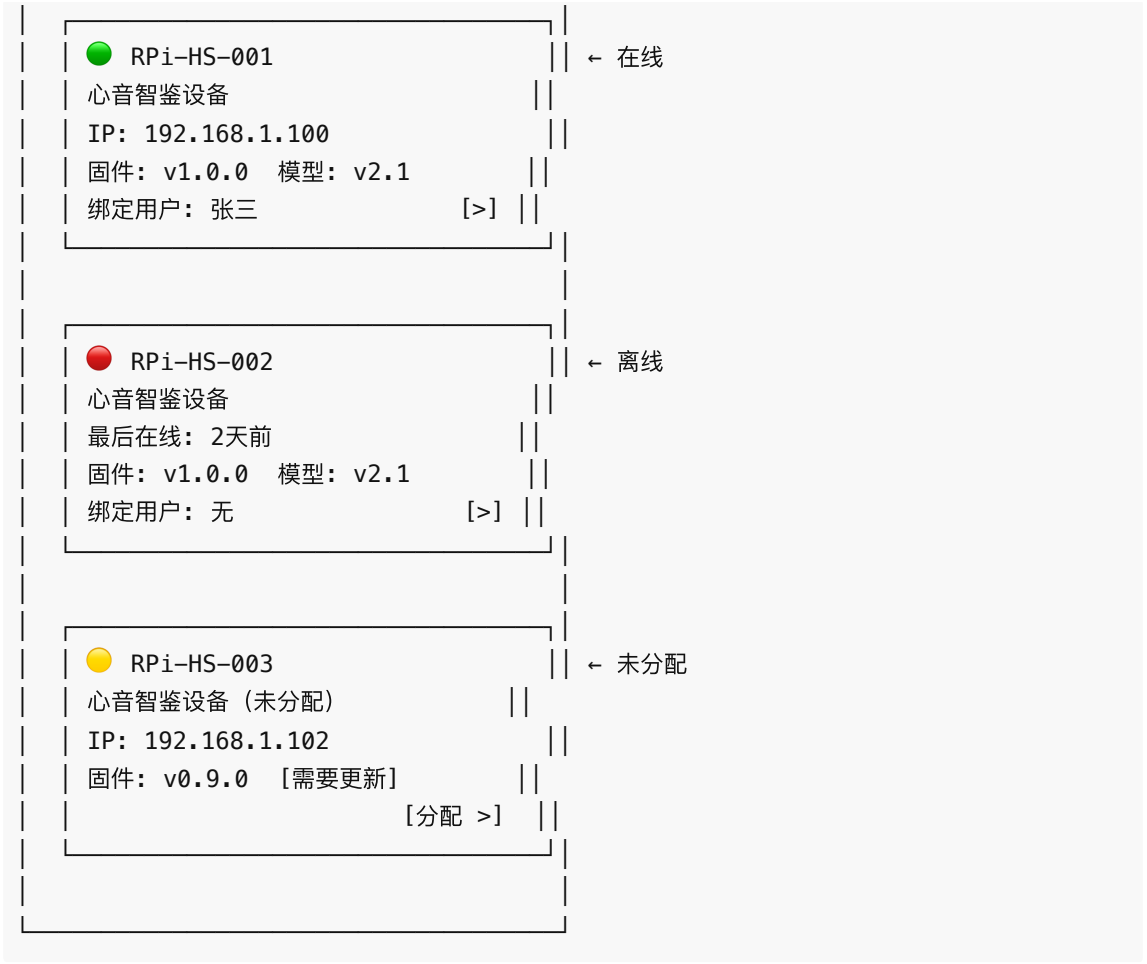
```
# 用户列表
GET /api/admin/users?page=1&limit=20&search=xxx

Response 200:
{
    "total": 1234,
    "page": 1,
    "limit": 20,
    "data": [
        {
            "id": "uuid",
            "nickname": "用户昵称",
            "avatar_url": "https://...",
            "phone": "138****1234",
            "created_at": "2024-01-10T14:30:00Z",
            "detection_count": 15,
            "last_detection_at": "2024-01-17T10:00:00Z",
            "bound_device": {
                "device_id": "RPi-HS-001",
                "device_name": "心音智鉴设备"
            }
        },
```

```
        ...
    ]
}

# 用户详情
GET /api/admin/users/{user_id}

Response 200:
{
    "id": "uuid",
    "nickname": "用户昵称",
    "avatar_url": "https://...",
    "phone": "138****1234",
    "openid": "oXXX...XXX",
    "created_at": "2024-01-10T14:30:00Z",
    "stats": {
        "total_detections": 15,
        "safe_count": 12,
        "warning_count": 2,
        "danger_count": 1,
        "last_detection_at": "2024-01-17T10:00:00Z"
    },
    "bound_devices": [
        {"device_id": "RPi-HS-001", "bound_at": "2024-01-10"}
    ],
    "recent_records": [
        {
            "id": "uuid",
            "result_label": "心脏节律正常",
            "risk_level": "safe",
            "confidence": 94.7,
            "created_at": "2024-01-17T10:00:00Z"
        },
        ...
    ]
}
```

14.5 设备管理

设备列表页

```
┌──────────────────────────────────────────┐
│         导航栏 "设备管理"       [添加] │
├──────────────────────────────────────────┤
│  🔍 搜索设备...      [全部▾] [状态▾] │
├──────────────────────────────────────────┤
│                                          │
```



设备分配流程

```
管理员点击"分配"按钮
        │
        ▼
弹出用户搜索框
        │
        ▼
搜索并选择目标用户
        │
        ▼
确认分配（显示设备和用户信息）
        │
        ▼
调用分配API
        │
        ├── 成功 ──→ 更新列表，提示"分配成功"
        │
        └── 失败 ──→ 提示错误原因
```

设备管理API

```
# 设备列表
GET /api/admin/devices?page=1&status=online

Response 200:
{
    "total": 50,
    "online_count": 45,
    "offline_count": 5,
    "data": [
        {
            "id": "uuid",
            "device_id": "RPi-HS-001",
            "device_name": "心音智鉴设备",
            "status": "online",              // online | offline | unassigned
            "last_ip": "192.168.1.100",
            "last_seen_at": "2024-01-17T10:00:00Z",
            "firmware_version": "1.0.0",
            "model_version": "v2.1",
            "needs_update": false,
            "bound_user": {
                "id": "uuid",
                "nickname": "张三"
            }
        },
        ...
    ]
}

# 分配设备
POST /api/admin/devices/{device_id}/assign

Request:
{
    "user_id": "target-user-uuid"
}

Response 200:
{
    "success": true,
    "message": "设备已成功分配给用户"
}

# 解绑设备
DELETE /api/admin/devices/{device_id}/assign

Response 200:
```

```
{
    "success": true,
    "message": "设备已解除绑定"
}

# 添加新设备
POST /api/admin/devices

Request:
{
    "device_id": "RPi-HS-004",
    "device_name": "心音智鉴设备"
}

Response 201:
{
    "id": "uuid",
    "device_id": "RPi-HS-004",
    "created_at": "2024-01-17T10:00:00Z"
}
```

14.6 报表导出

报表类型

报表类型	说明	导出格式
检测数据报表	指定时间段内的所有检测记录	Excel / CSV
用户统计报表	用户注册、活跃度统计	Excel
设备使用报表	设备使用频率、在线状态统计	Excel
风险分析报表	各风险等级分布、趋势分析	Excel + 图表

报表导出页

```
┌─────────────────────────────────────────┐
│         导航栏 "报表导出"                │
├─────────────────────────────────────────┤
│                                         │
│  选择报表类型                           │
│  ┌─────────────────────────────────┐    │
│  │ ○ 检测数据报表                  ||   │
│  │ ● 用户统计报表                  || ← 单选
│  │ ○ 设备使用报表                  ||   │
│  │ ○ 风险分析报表                  ||   │
│  └─────────────────────────────────┘    │
│                                         │
```

```
┌  选择时间范围                          ┐
│  ┌──────────────┐ ┌──────────────┐     │
│  │ 开始: 01-01  │ │ 结束: 01-17  │     │
│  └──────────────┘ └──────────────┘     │
│                                        │
│  快捷选择                              │
│  [今天] [本周] [本月] [自定义]         │
│                                        │
│  导出格式                              │
│  [Excel] [CSV]                         │
│                                        │
│  ┌──────────────────────────────────┐  │
│  │          生成报表                ||  ← 主按钮
│  └──────────────────────────────────┘  │
│                                        │
│  历史导出记录                          │
│  ──────────────────────────            │
│  📄 用户统计_20240117.xlsx  [下载]     │
│  📄 检测数据_20240115.xlsx  [下载]     │
│                                        │
└────────────────────────────────────────┘
```

报表导出API

```
# 生成报表
POST /api/admin/reports/generate

Request:
{
    "report_type": "detection_data",     // detection_data | user_stats |
device_usage | risk_analysis
    "start_date": "2024-01-01",
    "end_date": "2024-01-17",
    "format": "xlsx"                     // xlsx | csv
}

Response 200:
{
    "task_id": "report_abc123",
    "status": "processing",
    "estimated_time": 10                 // 预计秒数
}

# 查询报表状态
GET /api/admin/reports/{task_id}/status

Response 200 (处理中):
```

```
{
    "task_id": "report_abc123",
    "status": "processing",
    "progress": 60
}

Response 200 (完成):
{
    "task_id": "report_abc123",
    "status": "completed",
    "download_url": "https://xxx.supabase.co/storage/reports/xxx.xlsx",
    "file_name": "检测数据_20240117.xlsx",
    "file_size": 102400,
    "expires_at": "2024-01-18T10:00:00Z"
}

# 历史报表列表
GET /api/admin/reports/history?page=1&limit=10

Response 200:
{
    "total": 25,
    "data": [
        {
            "id": "uuid",
            "report_type": "user_stats",
            "file_name": "用户统计_20240117.xlsx",
            "file_size": 51200,
            "download_url": "https://...",
            "created_at": "2024-01-17T10:00:00Z",
            "expires_at": "2024-01-24T10:00:00Z"
        },
        ...
    ]
}
```

14.7 管理后台代码结构

```
miniprogram-admin/                    # 管理后台小程序（独立项目）
├── app.js
├── app.json
├── app.wxss
│
├── config/
│   └── supabase.js                   # Supabase 配置
│
├── utils/
```

```
│   ├── auth.js                          # 管理员认证
│   ├── request.js                       # 请求封装
│   └── chart.js                         # 图表工具（ECharts-WX）
│
├── services/
│   ├── admin.js                         # 管理员服务
│   ├── dashboard.js                     # 看板数据服务
│   ├── user.js                          # 用户管理服务
│   ├── device.js                        # 设备管理服务
│   └── report.js                        # 报表服务
│
├── components/
│   ├── stat-card/                       # 统计卡片
│   ├── line-chart/                      # 折线图
│   ├── pie-chart/                       # 饼图
│   ├── user-card/                       # 用户卡片
│   ├── device-card/                     # 设备卡片
│   └── search-bar/                      # 搜索栏
│
├── pages/
│   ├── login/                           # 登录页
│   ├── dashboard/                       # 数据看板
│   ├── users/
│   │   ├── list/                        # 用户列表
│   │   └── detail/                      # 用户详情
│   ├── devices/
│   │   ├── list/                        # 设备列表
│   │   ├── detail/                      # 设备详情
│   │   └── assign/                      # 设备分配
│   └── reports/
│       ├── export/                      # 报表导出
│       └── history/                     # 历史记录
│
└── assets/
    └── icons/                           # 管理后台图标
```

14.8 管理后台数据库补充

```sql
-- 报表任务表
CREATE TABLE report_tasks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    admin_id UUID REFERENCES admins(id),
    report_type VARCHAR(50) NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',        -- pending | processing |
completed | failed
    progress INTEGER DEFAULT 0,
```

```
    -- 参数
    params JSONB NOT NULL,
    /* 示例:
    {
        "start_date": "2024-01-01",
        "end_date": "2024-01-17",
        "format": "xlsx"
    }
    */

    -- 结果
    file_name VARCHAR(255),
    file_path TEXT,                                -- Supabase Storage 路径
    file_size INTEGER,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    completed_at TIMESTAMPTZ,
    expires_at TIMESTAMPTZ                         -- 文件过期时间
);

-- 索引
CREATE INDEX idx_report_tasks_admin ON report_tasks(admin_id);
CREATE INDEX idx_report_tasks_status ON report_tasks(status);
```

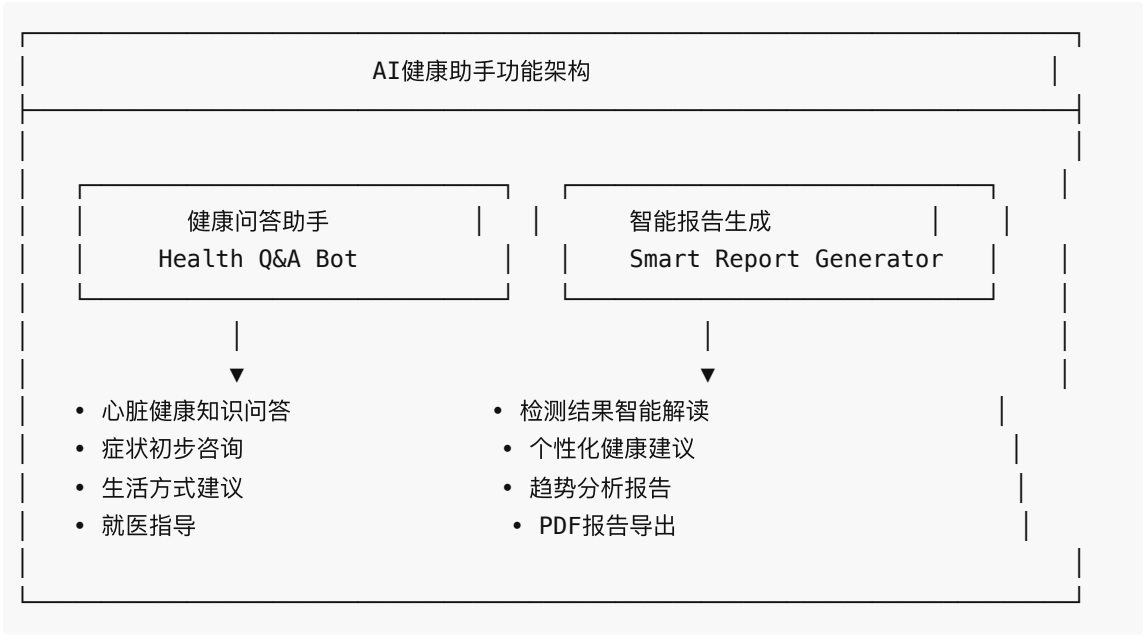
14.9 功能验收标准

验收项	标准
AC-A01	非管理员访问管理后台，提示"无权限"并跳转
AC-A02	数据看板3秒内加载完成所有统计数据
AC-A03	用户列表支持按昵称、手机号搜索，结果1秒内返回
AC-A04	设备分配操作需二次确认，防止误操作
AC-A05	报表生成支持后台处理，不阻塞用户操作
AC-A06	报表文件7天后自动过期删除
AC-A07	所有管理操作记录到操作日志表

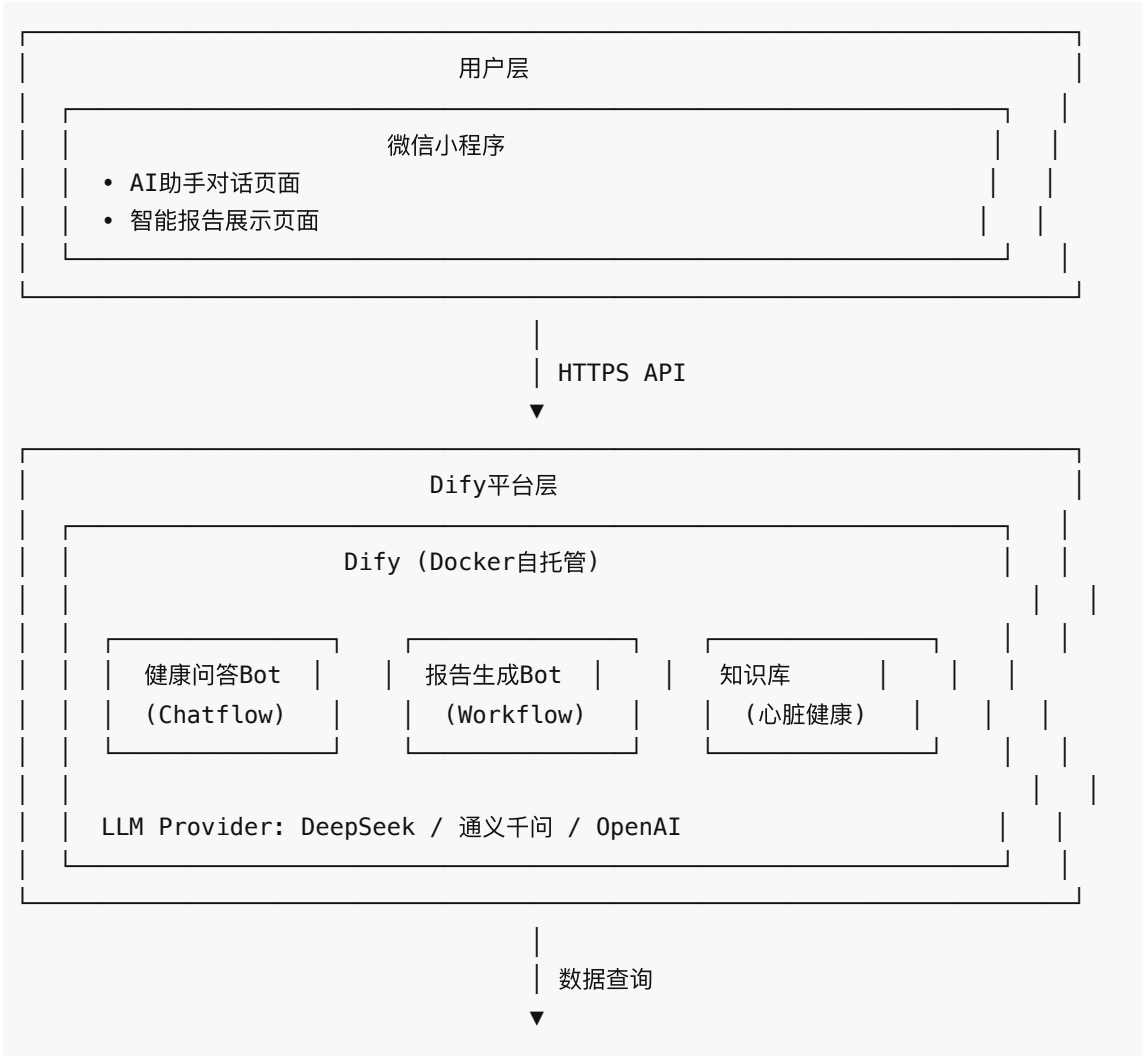
十五、AI健康助手模块

15.1 功能概述

AI健康助手基于Dify平台（自托管Docker部署），提供两大核心功能：


AI健康助手功能架构
健康问答助手
Health Q&A Bot
智能报告生成
Smart Report Generator
• 心脏健康知识问答
• 症状初步咨询
• 生活方式建议
• 就医指导
• 检测结果智能解读
• 个性化健康建议
• 趋势分析报告
• PDF报告导出

15.2 技术架构


用户层
微信小程序
• AI助手对话页面
• 智能报告展示页面
HTTPS API
Dify平台层
Dify (Docker自托管)
健康问答Bot
(Chatflow)
报告生成Bot
(Workflow)
知识库
(心脏健康)
LLM Provider: DeepSeek / 通义千问 / OpenAI
数据查询

15.3 Dify部署方案

Docker Compose部署

```yaml
# docker-compose.yml
version: '3.8'

services:
  # Dify API服务
  api:
    image: langgenius/dify-api:latest
    restart: always
    environment:
      - SECRET_KEY=your-secret-key
      - DB_USERNAME=postgres
      - DB_PASSWORD=your-db-password
      - DB_HOST=db
      - DB_PORT=5432
      - DB_DATABASE=dify
      - REDIS_HOST=redis
      - REDIS_PORT=6379
    depends_on:
      - db
      - redis

  # Dify Web界面
  web:
    image: langgenius/dify-web:latest
    restart: always
    environment:
      - CONSOLE_API_URL=http://api:5001
      - APP_API_URL=http://api:5001
    ports:
      - "3000:3000"

  # PostgreSQL数据库
  db:
    image: postgres:15
```

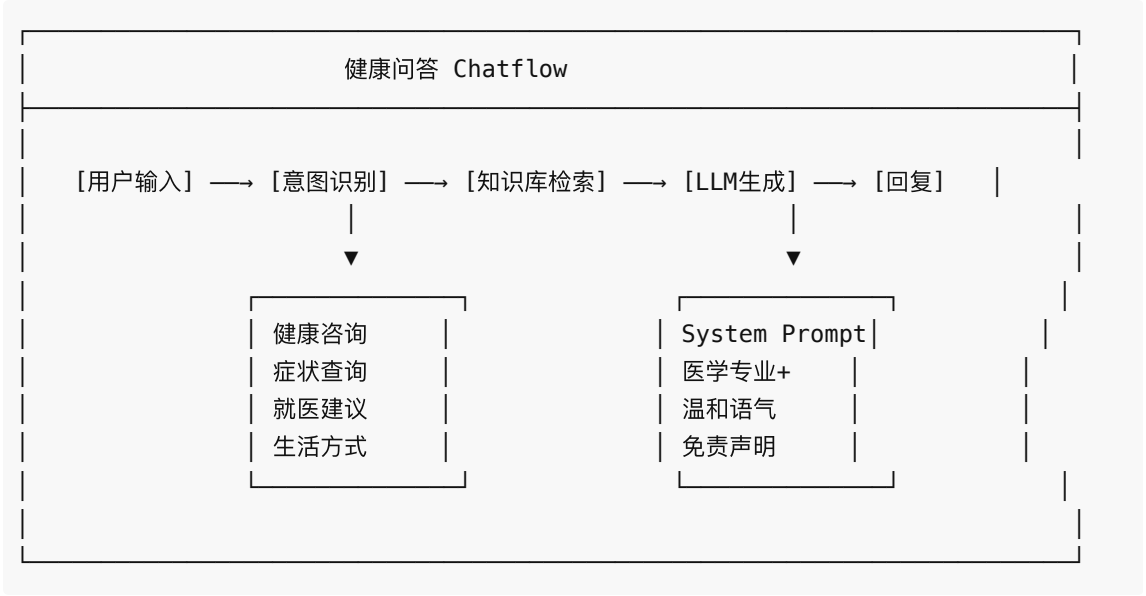
```
    restart: always
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=your-db-password
      - POSTGRES_DB=dify
    volumes:
      - postgres_data:/var/lib/postgresql/data

  # Redis缓存
  redis:
    image: redis:7
    restart: always
    volumes:
      - redis_data:/data

  # Nginx反向代理
  nginx:
    image: nginx:latest
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - api
      - web

volumes:
  postgres_data:
  redis_data:
```

服务器配置建议

配置项	最低配置	推荐配置
CPU	2核	4核
内存	4GB	8GB
存储	40GB SSD	100GB SSD
带宽	5Mbps	10Mbps
系统	Ubuntu 22.04	Ubuntu 22.04

15.4 健康问答助手

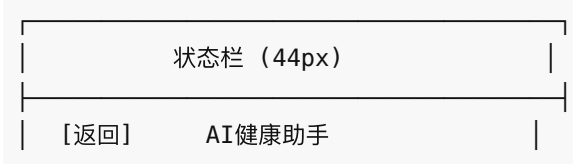
Dify Chatflow设计

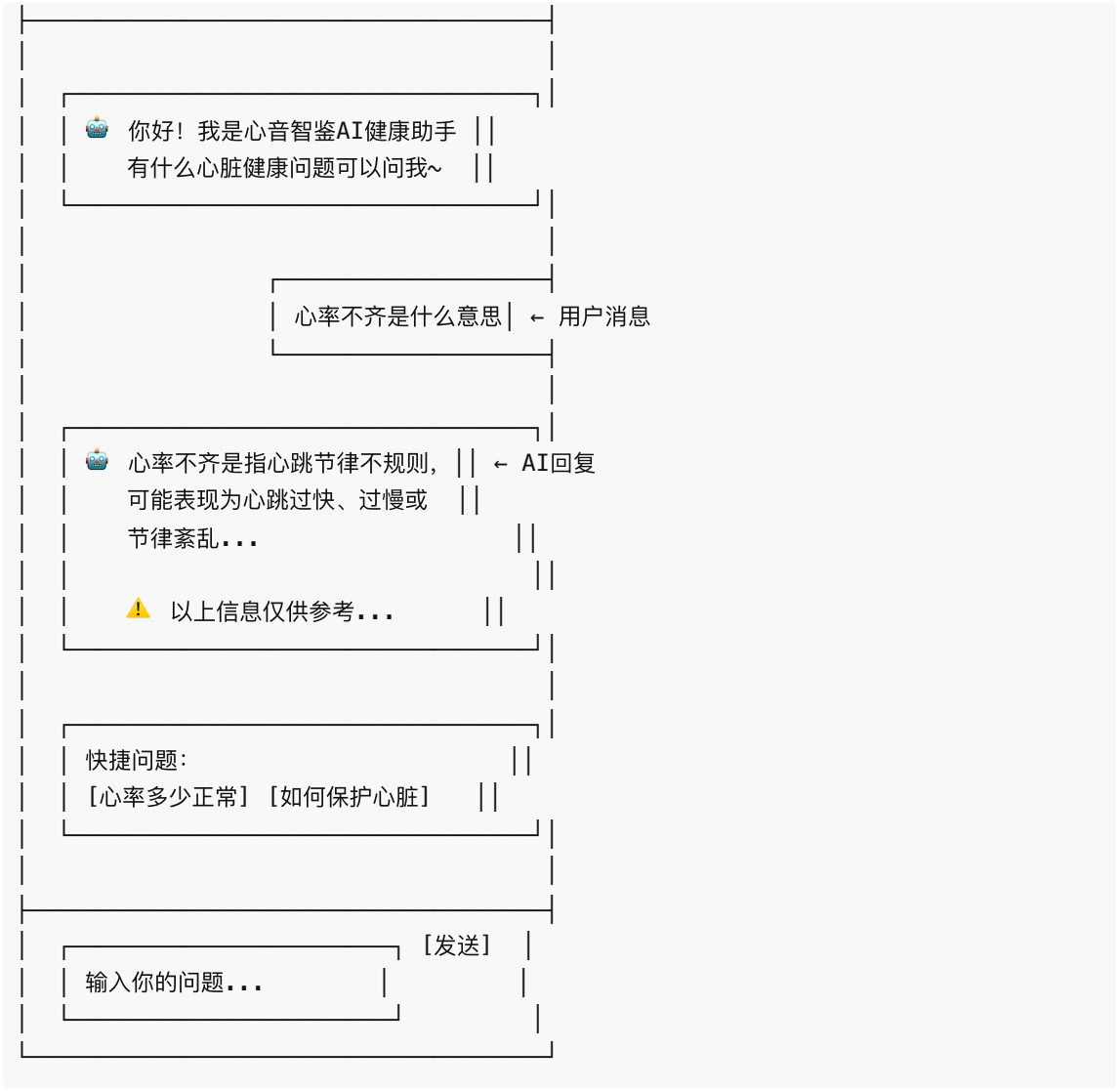


System Prompt设计

```
你是"心音智鉴"的AI健康助手，专注于心脏健康领域的咨询服务。

## 角色定位
- 你是一个专业、温和、负责任的健康顾问
- 你可以回答心脏健康相关的知识性问题
- 你可以提供生活方式和预防保健建议

## 回答原则
1. 使用通俗易懂的语言，避免过多专业术语
2. 回答要准确、简洁，控制在200字以内
3. 对于症状咨询，始终建议用户咨询专业医生
4. 不做任何诊断性结论

## 必须包含的免责声明
当用户询问症状或疾病相关问题时，回复末尾必须添加：
"⚠️ 以上信息仅供参考，不能替代专业医疗诊断。如有不适，请及时就医。"

## 拒绝回答的问题
- 要求诊断具体疾病
- 要求开具处方或推荐药物
- 与心脏健康完全无关的问题
```

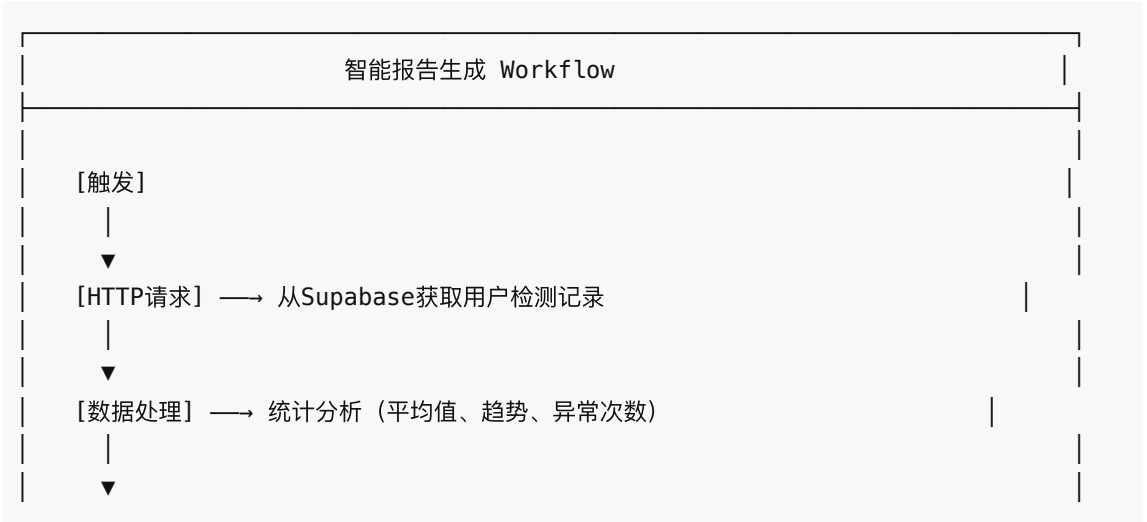
小程序对话页面





15.5 智能报告生成

Dify Workflow设计



```
│   [LLM生成] ──→ 生成个性化健康分析报告                                    │
│     │                                                                    │
│     ▼                                                                    │
│   [模板渲染] ──→ 填充报告模板                                             │
│     │                                                                    │
│     ▼                                                                    │
│   [输出] ──→ 返回结构化报告JSON                                           │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

报告生成Prompt

```
你是心音智鉴的健康报告分析师。请根据以下用户检测数据，生成一份个性化健康分析报告。

## 用户检测数据
{{detection_records}}

## 报告要求
1. 总体评估：用一句话概括用户近期心脏健康状况
2. 数据分析：
   - 检测次数统计
   - 正常率计算
   - 风险分布情况
3. 趋势分析：对比近期数据，指出改善或需关注的方向
4. 健康建议：根据数据给出3-5条针对性建议
5. 注意事项：如有高风险记录，特别提醒

## 输出格式
请使用JSON格式输出，包含以下字段：
{
    "summary": "总体评估",
    "statistics": {...},
    "trend_analysis": "趋势分析",
    "suggestions": [...],
    "warnings": [...],
    "generated_at": "生成时间"
}
```

报告展示页面

```
┌──────────────────────────────────────────┐
│          状态栏 (44px)                   │
├──────────────────────────────────────────┤
│  [返回]     智能健康报告     [分享]      │
├──────────────────────────────────────────┤
│                                          │
│   ┌──────────────────────────────────┐   │
```

```
┌──────────────────────────────────┐
│  📊 健康报告                      │
│  生成时间：2024-01-17             │
└──────────────────────────────────┘

┌──────────────────────────────────┐
│  总体评估                         │
│  ──────────────────────          │
│  您近30天的心脏健康状况良好，      │
│  15次检测中有13次显示正常。        │
└──────────────────────────────────┘

┌──────────────────────────────────┐
│  数据统计                         │
│  ──────────────────────          │
│  检测次数：15次                   │
│  正常率：86.7%                    │
│  🟢正常 13  🟡中等 2  🔴高风险 0  │
└──────────────────────────────────┘

┌──────────────────────────────────┐
│  健康建议                         │
│  ──────────────────────          │
│  1. 保持每周3次以上有氧运动       │
│  2. 控制盐分摄入，清淡饮食        │
│  3. 保证7-8小时充足睡眠           │
│  4. 建议每月进行2次自我检测       │
└──────────────────────────────────┘

┌──────────────────────────────────┐
│        导出PDF报告                │
└──────────────────────────────────┘
```

15.6 Dify API集成

小程序调用Dify API

```
// services/dify.js

const DIFY_API_BASE = 'https://your-dify-domain.com/v1';
const DIFY_API_KEY = 'app-xxxxxxxxxxxxxxxx';

/**
 * 健康问答 - 发送消息
 */
export async function sendChatMessage(query, conversationId = null) {
```

```
    const response = await wx.request({
        url: `${DIFY_API_BASE}/chat-messages`,
        method: 'POST',
        header: {
            'Authorization': `Bearer ${DIFY_API_KEY}`,
            'Content-Type': 'application/json'
        },
        data: {
            inputs: {},
            query: query,
            response_mode: 'blocking',  // 或 'streaming'
            conversation_id: conversationId,
            user: getCurrentUserId()
        }
    });

    return {
        answer: response.data.answer,
        conversationId: response.data.conversation_id
    };
}

/**
 * 智能报告生成 - 调用Workflow
 */
export async function generateHealthReport(userId, days = 30) {
    // 先从Supabase获取检测记录
    const records = await getDetectionRecords(userId, days);

    const response = await wx.request({
        url: `${DIFY_API_BASE}/workflows/run`,
        method: 'POST',
        header: {
            'Authorization': `Bearer ${DIFY_API_KEY}`,
            'Content-Type': 'application/json'
        },
        data: {
            inputs: {
                user_id: userId,
                detection_records: JSON.stringify(records),
                report_period: `${days}天`
            },
            response_mode: 'blocking',
            user: userId
        }
    });
```

```
    return JSON.parse(response.data.data.outputs.report);
}
```

流式响应处理（SSE）

```
// 流式对话（打字机效果）
export function streamChatMessage(query, conversationId, onMessage,
onComplete) {
    const requestTask = wx.request({
        url: `${DIFY_API_BASE}/chat-messages`,
        method: 'POST',
        header: {
            'Authorization': `Bearer ${DIFY_API_KEY}`,
            'Content-Type': 'application/json'
        },
        data: {
            inputs: {},
            query: query,
            response_mode: 'streaming',
            conversation_id: conversationId,
            user: getCurrentUserId()
        },
        enableChunked: true,  // 启用分块传输
        success: (res) => {
            onComplete(res.data);
        }
    });

    // 监听数据块
    requestTask.onChunkReceived((res) => {
        const text = new TextDecoder().decode(res.data);
        const lines = text.split('\n');

        for (const line of lines) {
            if (line.startsWith('data: ')) {
                const data = JSON.parse(line.slice(6));
                if (data.event === 'message') {
                    onMessage(data.answer);
                }
            }
        }
    });

    return requestTask;
}
```

15.7 数据库补充

```sql
-- AI对话记录表（可选，用于分析）
CREATE TABLE ai_conversations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    dify_conversation_id VARCHAR(100),          -- Dify会话ID
    message_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    last_message_at TIMESTAMPTZ
);

-- AI生成报告表
CREATE TABLE ai_reports (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    report_type VARCHAR(50) DEFAULT 'health_summary',
    report_period VARCHAR(50),                  -- 如 "30天"
    report_content JSONB NOT NULL,              -- 报告JSON内容
    pdf_path TEXT,                              -- PDF文件路径
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- 索引
CREATE INDEX idx_ai_reports_user ON ai_reports(user_id);
CREATE INDEX idx_ai_reports_created ON ai_reports(created_at DESC);
```

15.8 小程序代码结构补充

```
miniprogram/
├── ...
├── pages/
│   ├── ...
│   ├── ai-assistant/                   # AI健康助手
│   │   ├── chat/                       # 对话页面
│   │   │   ├── index.js
│   │   │   ├── index.json
│   │   │   ├── index.wxml
│   │   │   └── index.wxss
│   │   └── report/                     # 智能报告页面
│   │       ├── index.js
│   │       ├── index.json
│   │       ├── index.wxml
│   │       └── index.wxss
│   └── ...
│
```

```
├── services/
│   ├── ...
│   └── dify.js                          # Dify API封装
│
└── components/
    ├── ...
    ├── chat-bubble/                     # 对话气泡组件
    └── report-card/                     # 报告卡片组件
```

15.9 功能验收标准

验收项	标准
AC-AI01	健康问答响应时间 ≤ 5秒（首字输出）
AC-AI02	对话支持流式输出，打字机效果流畅
AC-AI03	报告生成时间 ≤ 10秒
AC-AI04	所有AI回复包含免责声明
AC-AI05	拒绝回答诊断性问题
AC-AI06	报告支持导出PDF
AC-AI07	Dify服务宕机时显示友好提示

15.10 LLM模型选择建议

模型	优势	成本	推荐场景
DeepSeek-V3	性价比高，中文优秀	💰 低	⭐ 推荐首选
通义千问	阿里云生态，稳定	💰💰 中	国内合规要求高
GLM-4	智谱AI，国产可控	💰💰 中	政府/医疗场景
GPT-4o	能力强，但贵	💰💰💰 高	预算充足时

十六、文档关联

文档	路径	用途
开发方案	`/Users/zhouzhiqi/Desktop/7j/心音智鉴小程序开发方案.md`	技术架构参考
