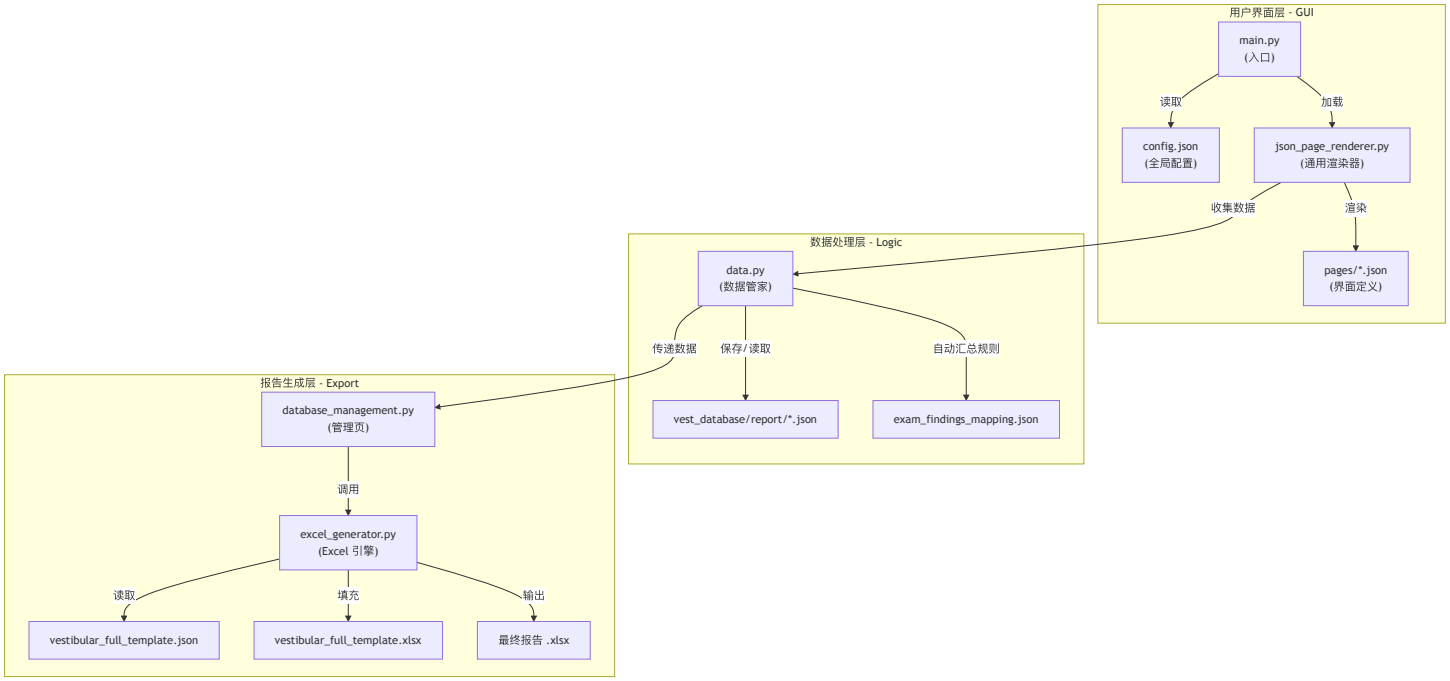
前庭功能检查报告系统 v2.0

1. 系统架构示意图

本系统是一个**完全配置驱动**（Configuration-Driven）的桌面应用。Python 代码仅作为通用引擎，具体的业务界面、逻辑、报告格式全由 JSON 文件定义。



2. 核心设计理念

- **UI 解耦**：界面不是用 Python 代码写死的，而是读取 `pages/` 下的 JSON 动态生成。改字段、改选项、改布局只需改 JSON。
- **报告解耦**：Excel 报告的生成逻辑（填哪个格子、N/A 补齐规则、列表拼接）全部在 `vest_database/templates/*.json` 里配置。
- **规则解耦**：“检查所见”的自动汇总逻辑（哪些字段要汇总、什么算空值）由 `exam_findings_mapping.json` 定义。

3. 项目文件结构解析

```
hospital_report_system/
├── main.py                          # [程序入口] 初始化窗口、菜单、导航栏，调度各模块
├── config.json                      # [全局配置] 系统名称、窗口大小、数据库路径、启用页面列表
├── data.py                          # [数据核心] 负责报告的 CRUD、文件读写、检查所见自动汇总
├── database_management.py           # [数据库页] 报告列表管理、搜索、调用生成器导出 Excel
├── excel_generator.py               # [导出引擎] 通用 Excel 生成器，解析模板配置执行写入
├── json_page_renderer.py            # [UI 引擎]  通用界面渲染器，解析 page JSON 生成控件
├── exam_findings_mapping.json       # [业务规则] 定义“检查所见”自动汇总的字段来源与空值判定
├── hospital_report.spec             # [打包脚本] Windows PyInstaller 打包配置
├── xlsx_reader.py                   # [开发工具] 辅助脚本，用于解析 Excel 结构生成 JSON（开发用
├── requirements.txt                 # [环境依赖] Python 依赖库
│
├── pages/                           # [界面配置] 存放所有检查页面的布局定义
│   ├── index.json                   # 页面导航索引（定义顺序、显示名称、启用状态）
│   ├── basic_info.json              # 基本信息页
│   ├── position_test.json           # 位置试验（整合 DHT/RT/其他位置试验）
│   ├── caloric_test.json            # 温度试验
│   ├── ... (head_impulse, saccade 等其他检查项)
│   └── exam_findings.json           # 检查所见页
│
└── vest_database/                   # [数据存储] 默认数据库目录
    ├── report/                      # 存放生成的患者报告 JSON（按日期文件夹归档）
    └── templates/                   # [报告模板]
        ├── templates_index.json             # 模板索引
        ├── vestibular_full_template.json    # 全套版模板映射配置（核心映射文件）
        └── vestibular_full_template.xlsx    # 全套版 Excel 底表
```

4. 关键配置指南

4.1 修改界面 (pages/*.json)

想在“自发性眼震”页面加一个字段?

1. 打开 `pages/spontaneous_nystagmus.json` 。
2. 在 `fields` 数组里加一项：

```
{
  "key": "新字段名",
  "type": "entry",  // 或 combobox, number, radio...
  "label": "显示名称",
  "order": 99
}
```

3. 重启程序即生效。

4.2 修改报告导出 (templates/*.json)

想把新字段导出到 Excel 的 C50 格子?

1. 打开 `vest_database/templates/vestibular_full_template.json` 。
2. 在 `data_cells` 里加一项:

```
{
  "cell": "C50",
  "data_path": "自发性眼震.新字段名"
}
```

4.3 修改“检查所见”自动汇总

想把新字段加入自动汇总?

1. 打开根目录 `exam_findings_mapping.json` 。
2. 在对应模块的 `result_fields` 列表里加上 `"新字段名"` 。

4.4 配置 N/A 自动补齐

想让某组字段在"部分有值、部分为空"时自动填 N/A?

1. 打开模板 JSON。
2. 在 `na_fill_groups` 里添加一组 `data_paths` 。

4.5 主题配置

系统支持多种预设主题和自定义配色方案。

切换预设主题

打开根目录 `config.json` , 找到 `system.themes.current` 字段:

```
"themes": {
  "current": "blue_light"   // 可选: blue_light, blue_pro, light_clean
}
```

预设主题说明

- **blue_light**: 蓝色亮色风格（白底黑字，适合日间使用）
- **blue_pro**: 专业深蓝风格（深色模式，适合夜间或长时间使用）
- **light_clean**: 清爽亮色风格（极简白色风格）

自定义主题颜色

在 `config.json` 的 `system.themes.presets` 中可以自定义颜色：

```
"presets": {
  "my_theme": {
    "description": "我的自定义主题",
    "appearance_mode": "light",    // light, dark, 或 system
    "color_theme": "blue",         // CustomTkinter 内置主题: blue, green, dark-blue
    "colors": {
      "window_bg": ["#F5F6F8", "#F5F6F8"],        // [浅色模式, 深色模式]
      "text_primary": ["#1A1A1A", "#E0E0E0"],
      "accent_primary": ["#3B8ED0", "#3B8ED0"]
      // ... 更多颜色配置
    }
  }
}
```

可自定义的颜色键包括：

- `window_bg`, `header_bg`, `sidebar_bg`, `content_bg`, `section_bg` - 背景色
- `text_primary`, `text_secondary` - 文字颜色
- `accent_primary`, `accent_hover` - 强调色和悬停色
- `border`, `input_bg`, `input_border` - 边框和输入框
- `success`, `warning`, `error` - 状态提示色

5. 运行与打包

运行

```
pip install -r requirements.txt
python main.py
```

打包 (Windows)

```
pyinstaller hospital_report.spec
```

打包后在 `dist/hospital_report` 目录生成可执行文件（目录形式，非单文件）。