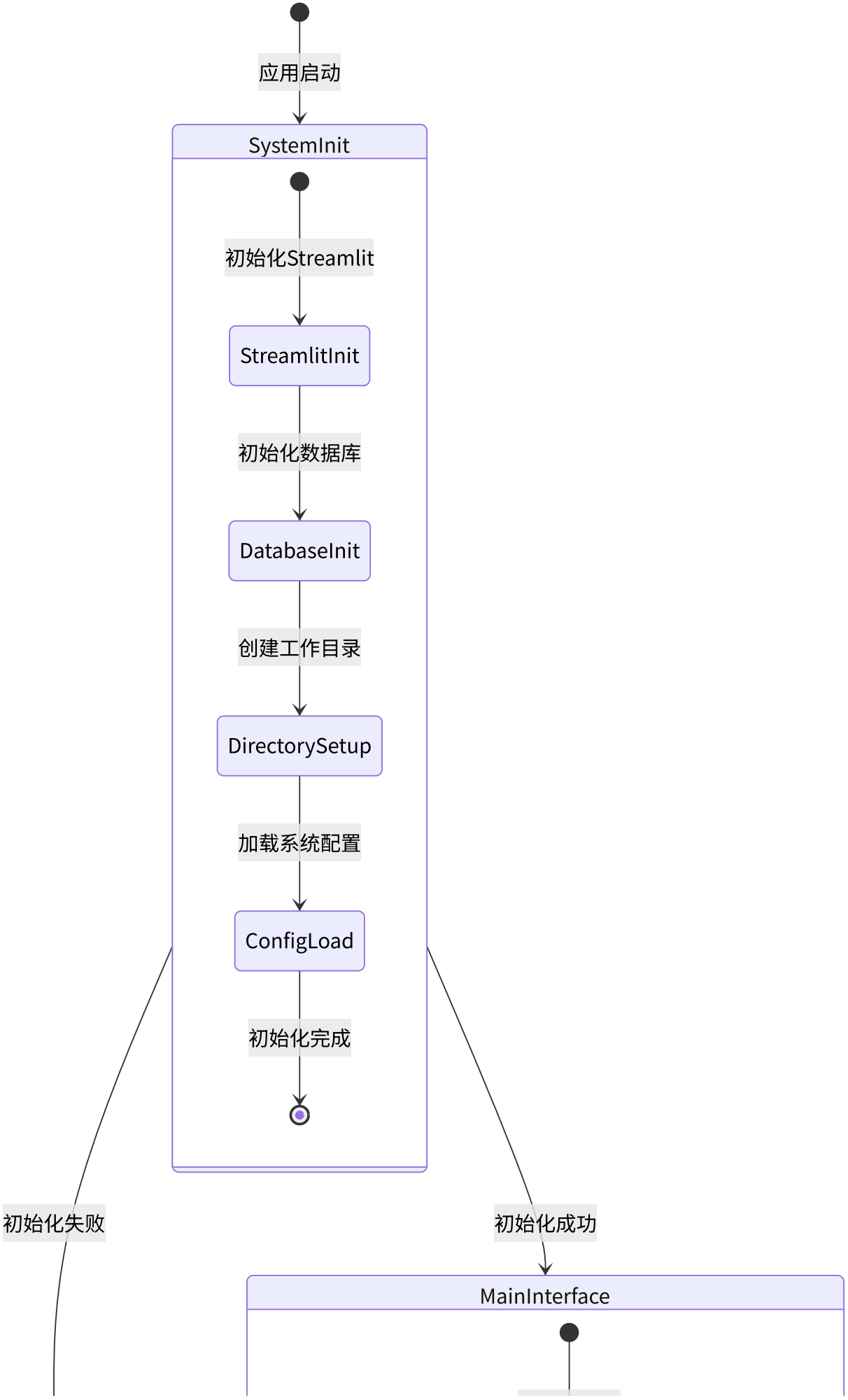
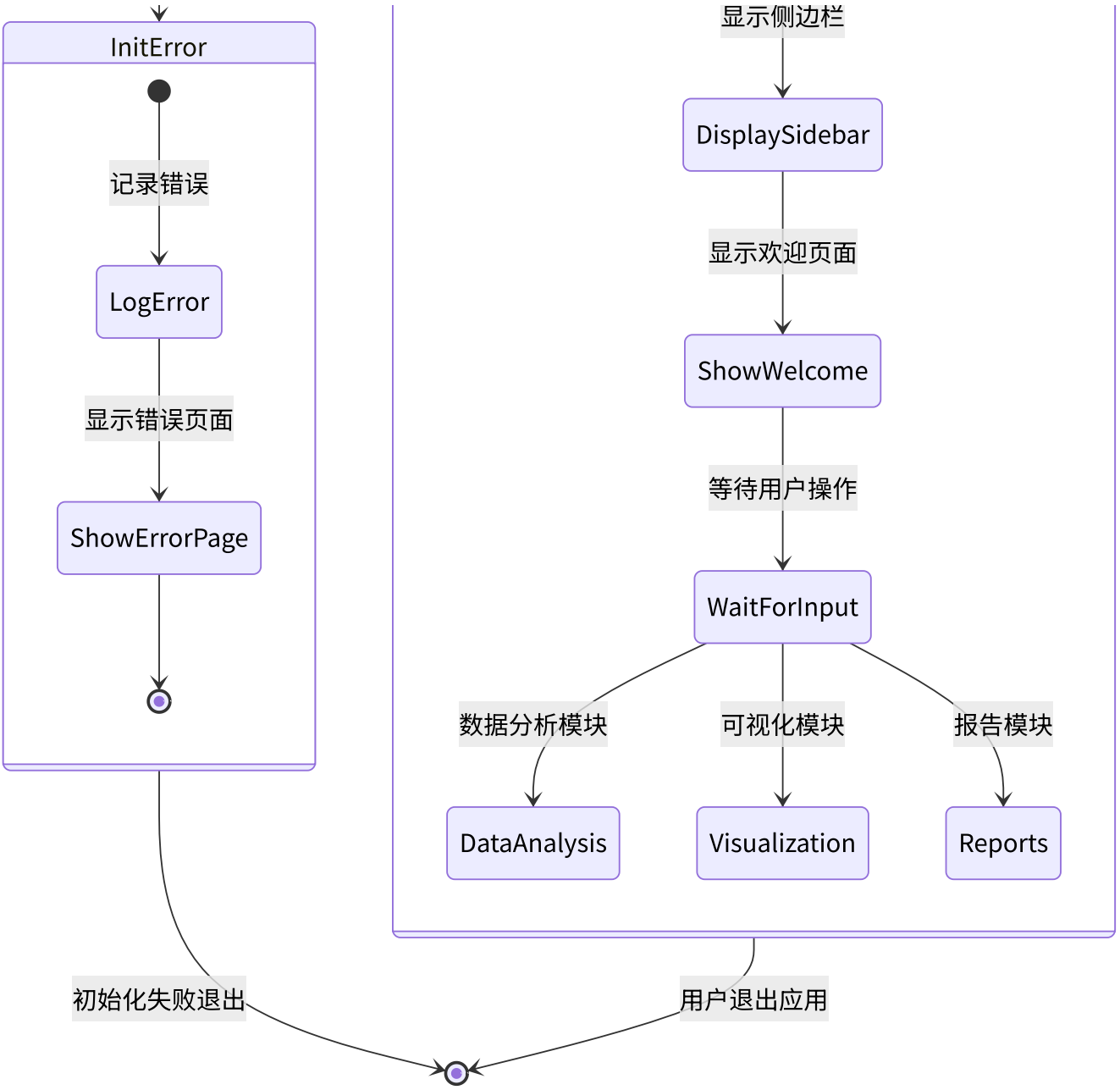
状态图

图1：系统初始化与核心流程


应用启动
SystemInit
初始化Streamlit
StreamlitInit
初始化数据库
DatabaseInit
创建工作目录
DirectorySetup
加载系统配置
ConfigLoad
初始化完成
初始化失败
初始化成功
MainInterface



详细介绍

基于代码分析的系统初始化流程：

根据项目中的 `streamlit_app.py` 和相关配置文件，系统初始化包含以下关键步骤：

1. **Streamlit初始化**
   - 设置页面配置：`st.set_page_config(page_title="中山大学医疗数据分析系统")`
   - 配置主题和布局参数
   - 初始化会话状态管理
2. 数据库初始化

- 检查SQLite数据库文件是否存在
- 创建必要的数据表结构（患者信息表、分析结果表等）
- 建立数据库连接池

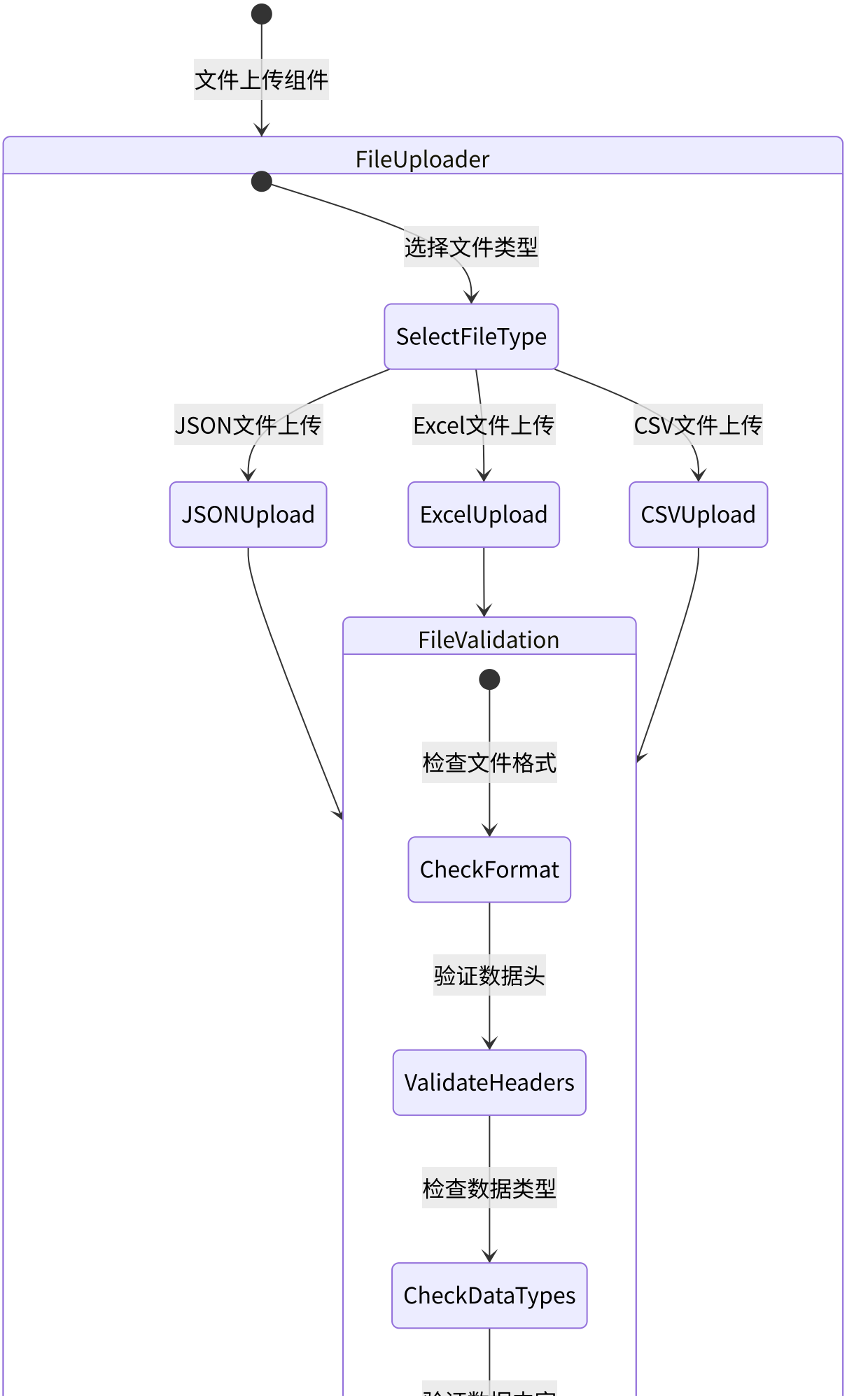
3. 目录结构设置

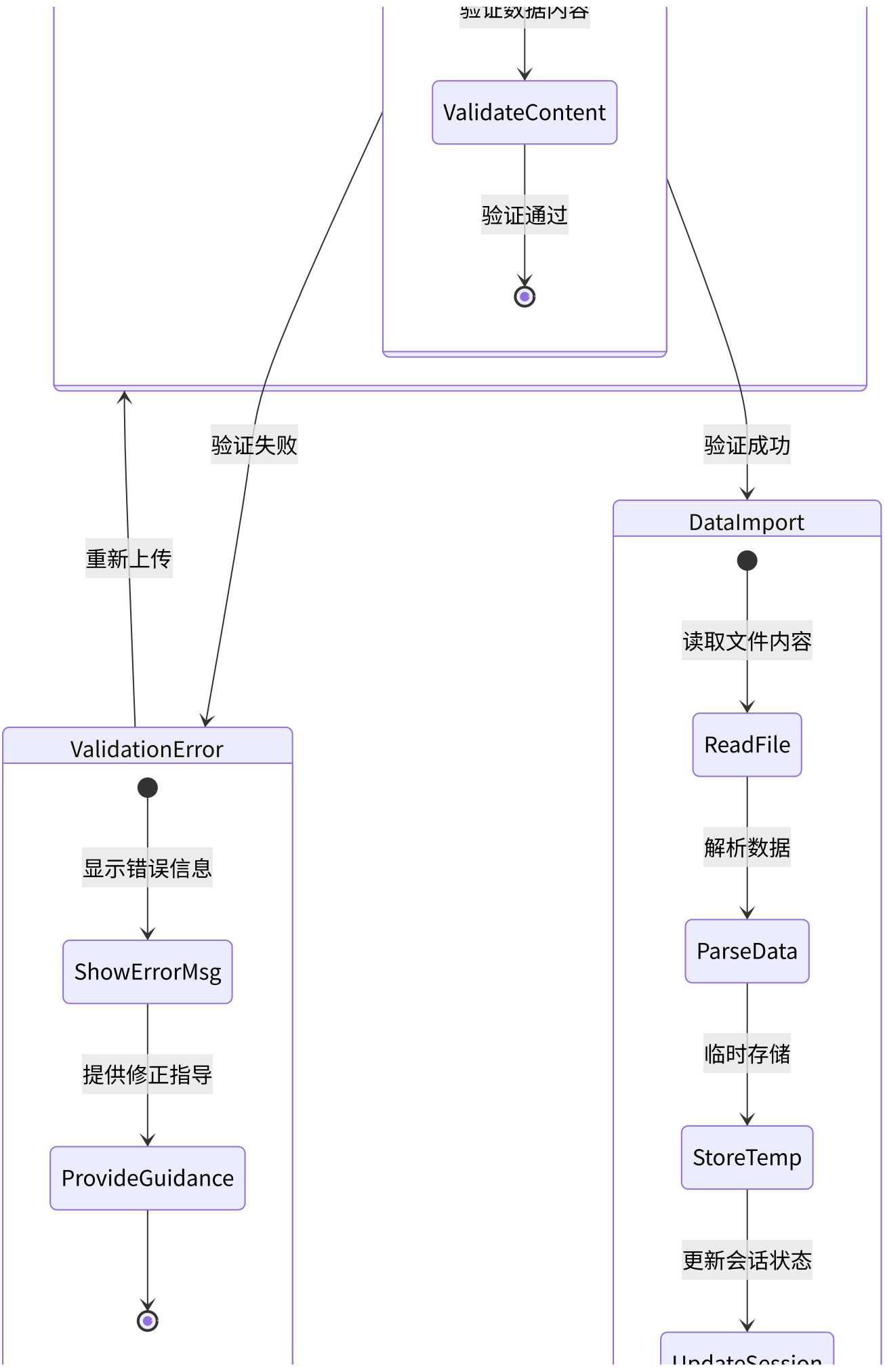
```
# 基于项目结构创建的目录
- data/            # 数据文件存储
- uploads/         # 用户上传文件
- reports/         # 生成的报告
- charts/          # 图表文件
- logs/            # 系统日志
```

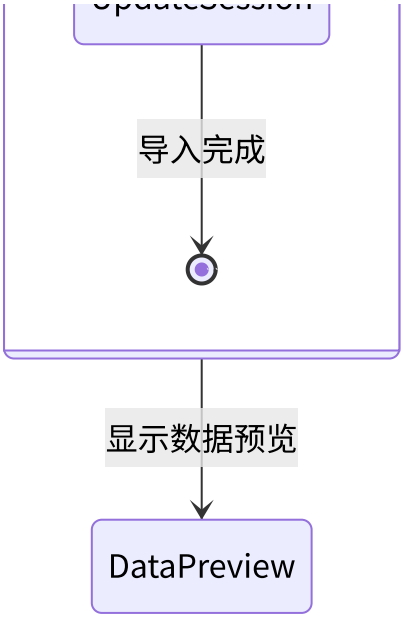
4. 配置加载

- 读取系统配置文件
- 初始化日志系统
- 设置数据处理参数

图2：文件上传与数据导入流程


文件上传组件
FileUploader
选择文件类型
SelectFileType
JSON文件上传
Excel文件上传
CSV文件上传
JSONUpload
ExcelUpload
CSVUpload
FileValidation
检查文件格式
CheckFormat
验证数据头
ValidateHeaders
检查数据类型
CheckDataTypes


验证数据内容
ValidateContent
验证通过
验证失败
验证成功
重新上传
DataImport
读取文件内容
ReadFile
解析数据
ParseData
临时存储
StoreTemp
更新会话状态
ValidationError
显示错误信息
ShowErrorMsg
提供修正指导
ProvideGuidance



详细介绍

基于 **Streamlit** 文件上传组件的实现：

1. 文件上传界面

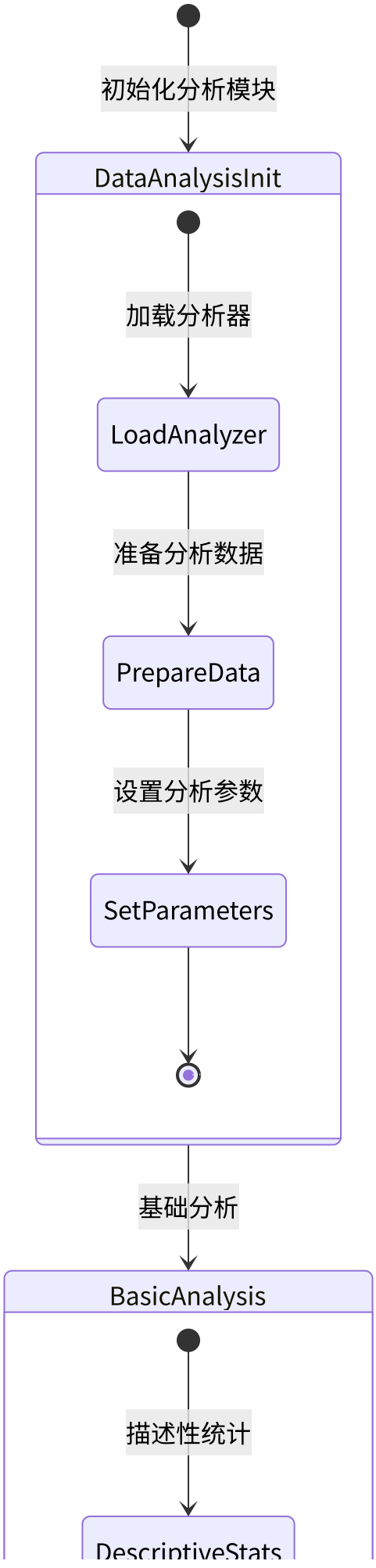
```python
uploaded_file = st.file_uploader(
    "选择医疗数据文件",
    type=['csv', 'xlsx', 'json'],
    accept_multiple_files=False
)
```

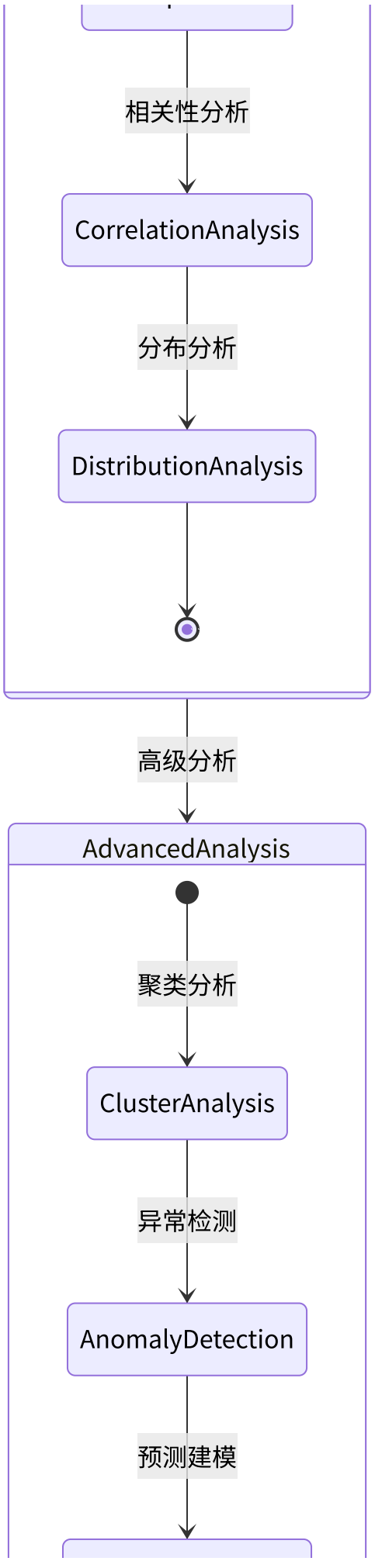
2. 文件格式验证
   - **CSV文件**: 检查分隔符、编码格式、列名规范
   - **Excel文件**: 验证工作表结构、数据范围
   - **JSON文件**: 验证JSON格式、必需字段
3. 数据头验证

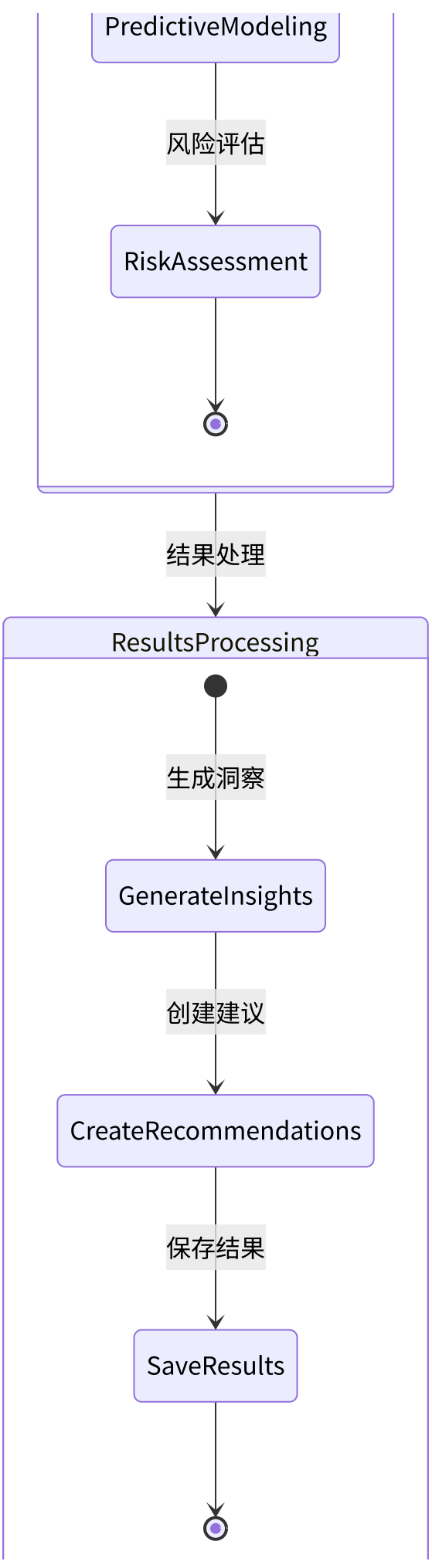
```python
required_columns = ['患者ID', '年龄', '性别', '症状', '检查结果']
missing_columns = set(required_columns) - set(df.columns)
if missing_columns:
    st.error(f"缺少必需列: {missing_columns}")
```

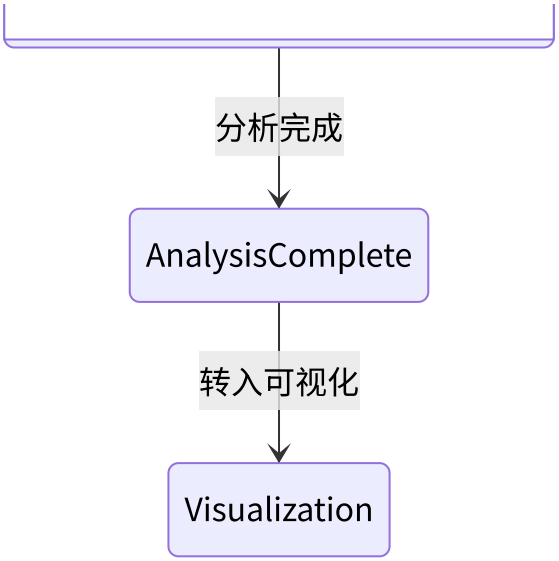
4. 数据预览功能
   - 使用 `st.dataframe()` 展示前几行数据
   - 显示数据统计信息：行数、列数、缺失值统计
   - 提供数据类型自动识别和手动调整选项

图3：数据分析与处理流程


初始化分析模块
DataAnalysisInit
加载分析器
LoadAnalyzer
准备分析数据
PrepareData
设置分析参数
SetParameters
基础分析
BasicAnalysis
描述性统计
DescriptiveStats


相关性分析
CorrelationAnalysis
分布分析
DistributionAnalysis
高级分析
AdvancedAnalysis
聚类分析
ClusterAnalysis
异常检测
AnomalyDetection
预测建模


PredictiveModeling
风险评估
RiskAssessment
结果处理
ResultsProcessing
生成洞察
GenerateInsights
创建建议
CreateRecommendations
保存结果
SaveResults



详细介绍

基于项目 `analyzer.py` 模块的分析流程：

1. 基础统计分析

```
# 描述性统计
df.describe()  # 均值、标准差、分位数
df.info()      # 数据类型、缺失值
df.value_counts()  # 频次统计
```

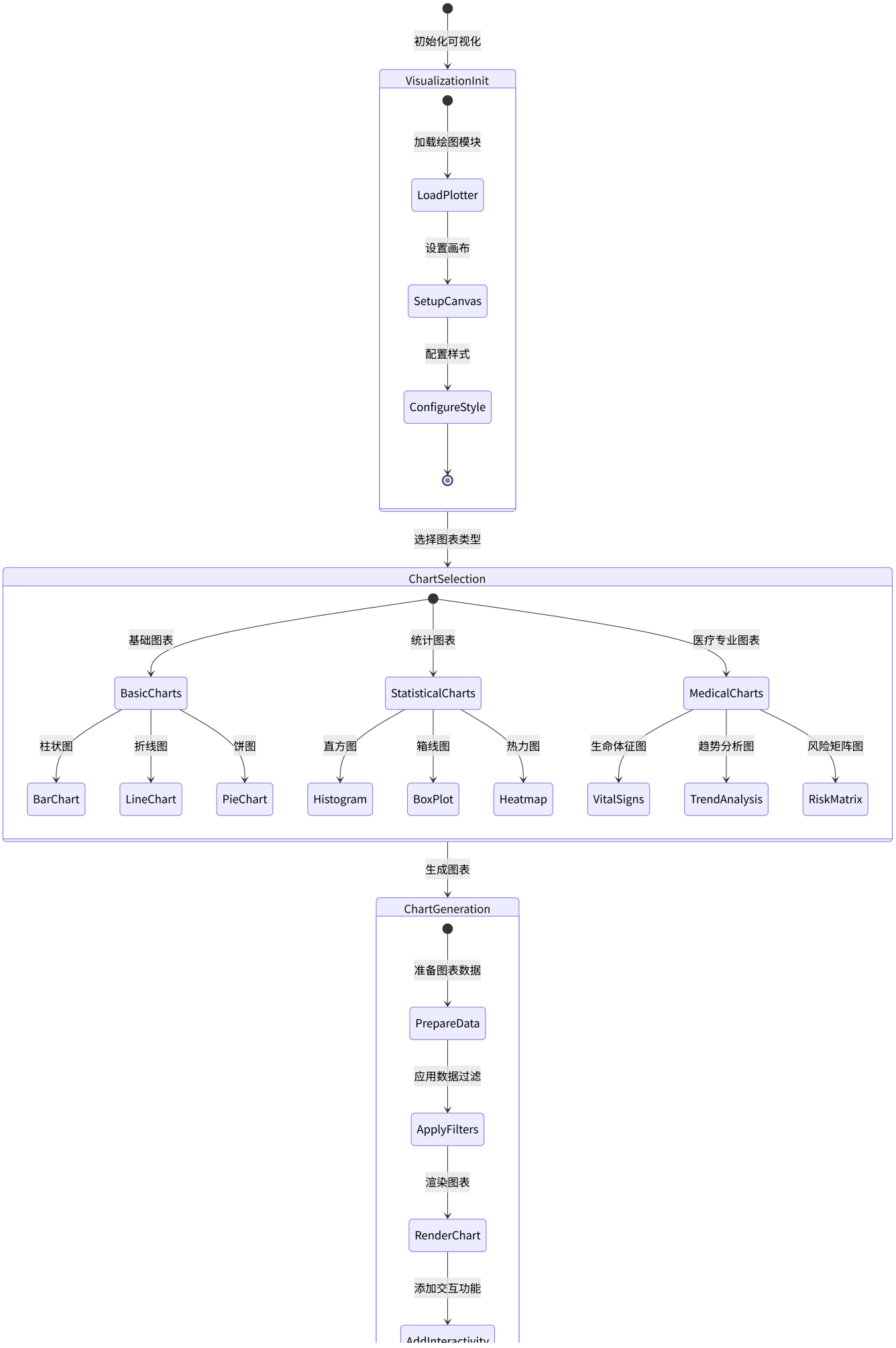
2. 相关性分析
   - 使用 Pearson 相关系数分析数值变量
   - 使用 Chi-square 检验分析分类变量
   - 生成相关性矩阵热力图
3. 聚类分析

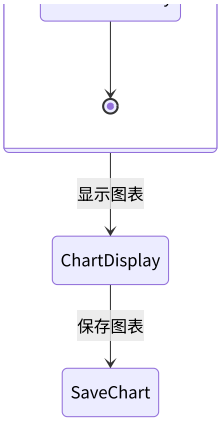
```
from sklearn.cluster import KMeans
# 基于症状和检查结果进行患者聚类
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(features)
```

4. 异常检测
   - 使用 Isolation Forest 检测异常值
   - 基于统计方法识别离群点
   - 医疗指标异常范围检测
5. 风险评估

- 根据年龄、症状、检查结果计算风险评分
- 分层风险分级（低、中、高风险）
- 生成个性化健康建议

图4：数据可视化生成流程


初始化可视化
VisualizationInit
加载绘图模块
LoadPlotter
设置画布
SetupCanvas
配置样式
ConfigureStyle
选择图表类型
ChartSelection
基础图表
统计图表
医疗专业图表
BasicCharts
StatisticalCharts
MedicalCharts
柱状图
折线图
饼图
直方图
箱线图
热力图
生命体征图
趋势分析图
风险矩阵图
BarChart
LineChart
PieChart
Histogram
BoxPlot
Heatmap
VitalSigns
TrendAnalysis
RiskMatrix
生成图表
ChartGeneration
准备图表数据
PrepareData
应用数据过滤
ApplyFilters
渲染图表
RenderChart
添加交互功能
AddInteractivity



详细介绍

基于项目 `plotter.py` 模块的可视化实现：

1. 图表库选择

```python
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
```

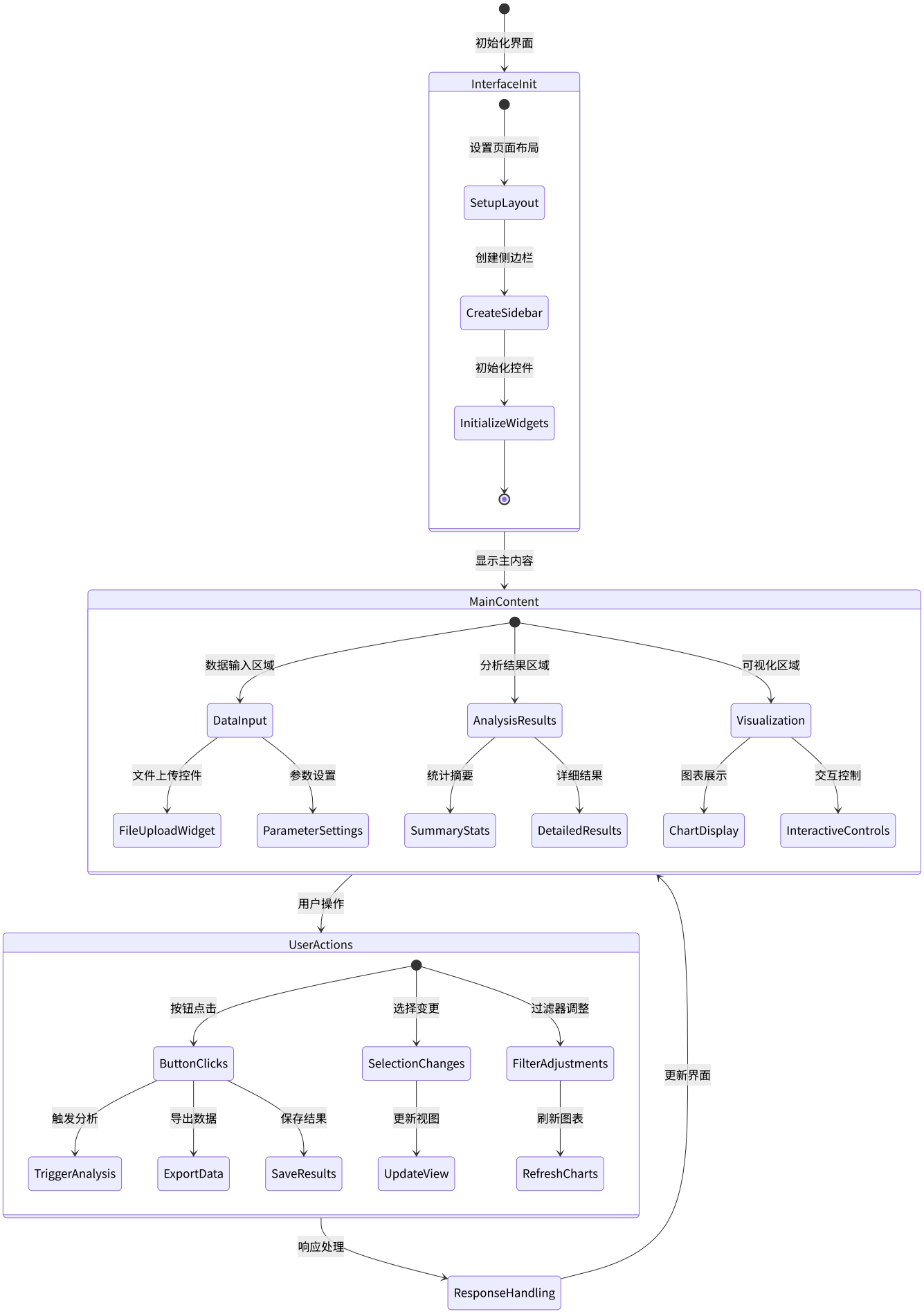
2. 基础图表类型
   - **柱状图**: 展示不同疾病类型的患者分布
   - **折线图**: 显示患者指标随时间的变化趋势
   - **饼图**: 显示性别、年龄组等分类数据占比
3. 统计可视化

```python
# 相关性热力图
fig = px.imshow(correlation_matrix,
                title="医疗指标相关性分析")

# 箱线图显示指标分布
fig = px.box(df, x="风险等级", y="血压值")
```

4. 医疗专业图表
   - **生命体征监控图**: 实时显示心率、血压、体温变化
   - **风险评估雷达图**: 多维度健康风险可视化
   - **患者群体分析散点图**: 基于聚类结果的可视化
5. 交互功能
   - 图表缩放、平移

- 数据点悬停显示详细信息
- 图例点击隐藏/显示数据系列

图5：用户界面交互流程


初始化界面
InterfaceInit
设置页面布局
SetupLayout
创建侧边栏
CreateSidebar
初始化控件
InitializeWidgets
显示主内容
MainContent
数据输入区域
分析结果区域
可视化区域
DataInput
AnalysisResults
Visualization
文件上传控件
参数设置
统计摘要
详细结果
图表展示
交互控制
FileUploadWidget
ParameterSettings
SummaryStats
DetailedResults
ChartDisplay
InteractiveControls
用户操作
UserActions
按钮点击
选择变更
过滤器调整
ButtonClicks
SelectionChanges
FilterAdjustments
触发分析
导出数据
保存结果
更新视图
刷新图表
TriggerAnalysis
ExportData
SaveResults
UpdateView
RefreshCharts
响应处理
ResponseHandling
更新界面

详细介绍

基于 **Streamlit** 框架的用户界面实现：

1. 页面布局设计

```python
# 侧边栏配置
with st.sidebar:
    st.title("功能导航")
    page = st.selectbox("选择功能",
                        ["数据上传", "数据分析", "可视化", "报告生成"])

# 主内容区域
col1, col2, col3 = st.columns([2, 3, 2])
```

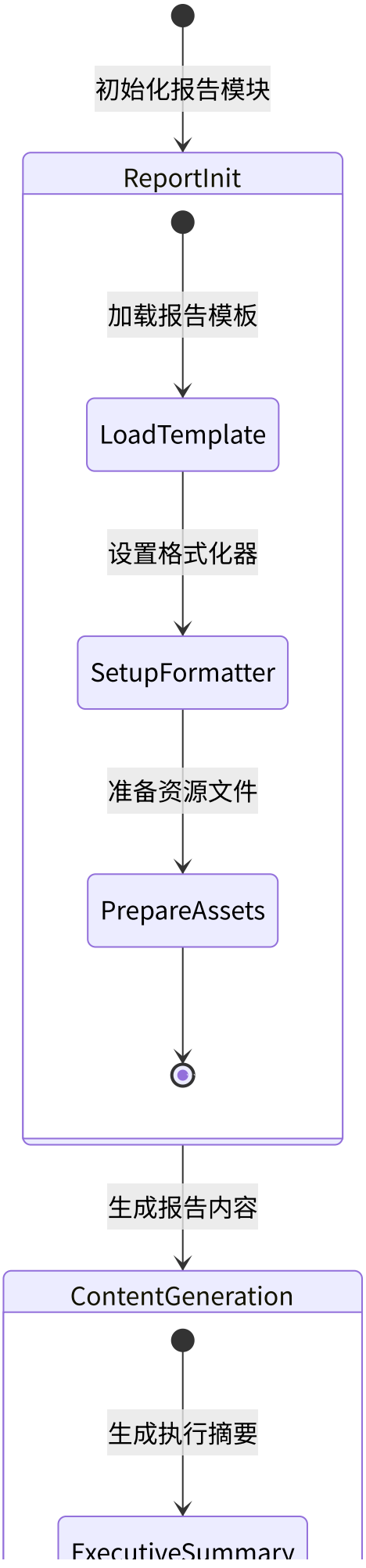
2. 交互控件
   - 文件上传器: `st.file_uploader()`
   - 参数滑块: `st.slider()` 设置分析参数
   - 选择框: `st.selectbox()` 选择图表类型
   - 按钮: `st.button()` 触发分析操作

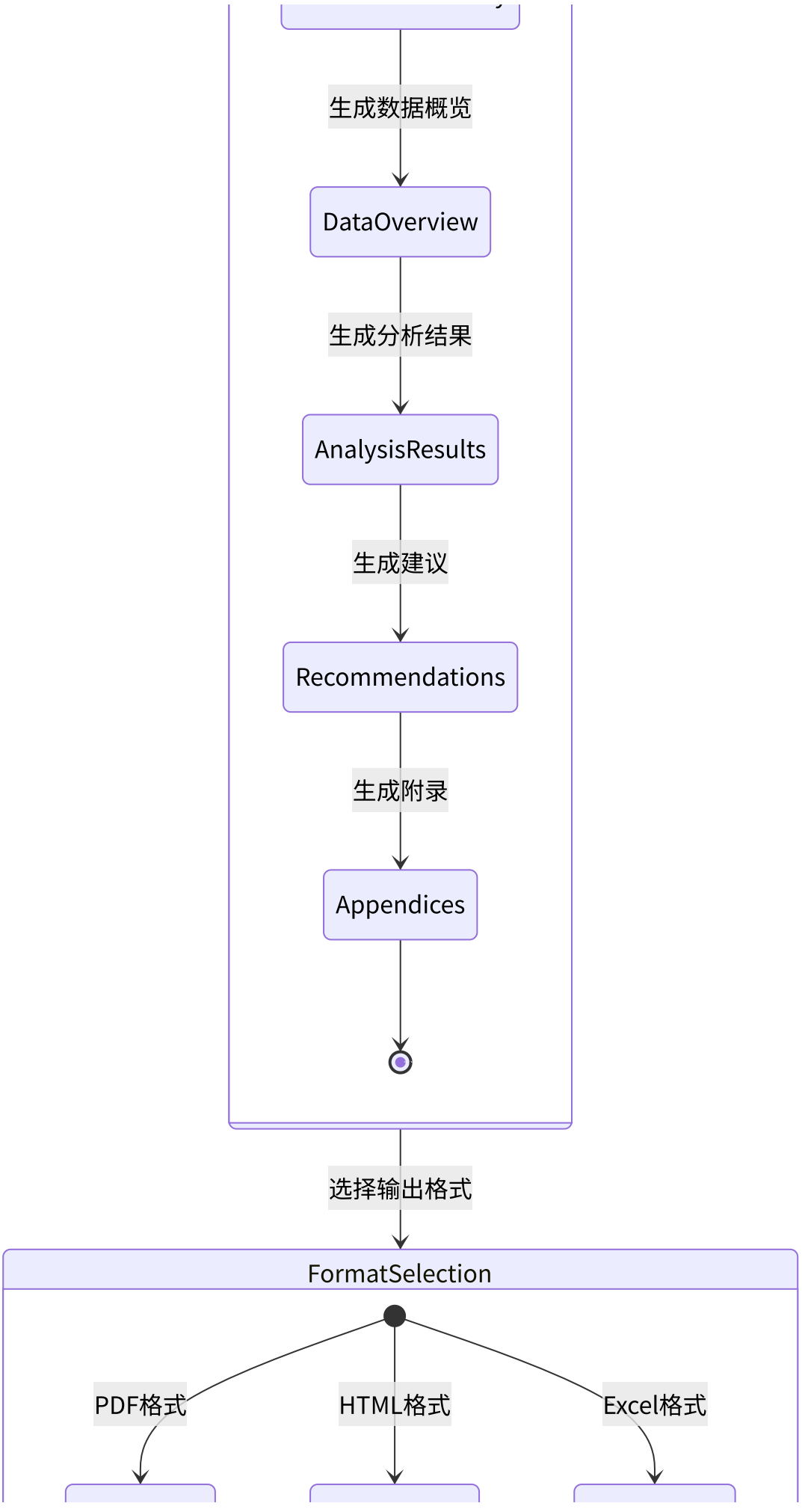
3. 实时响应机制

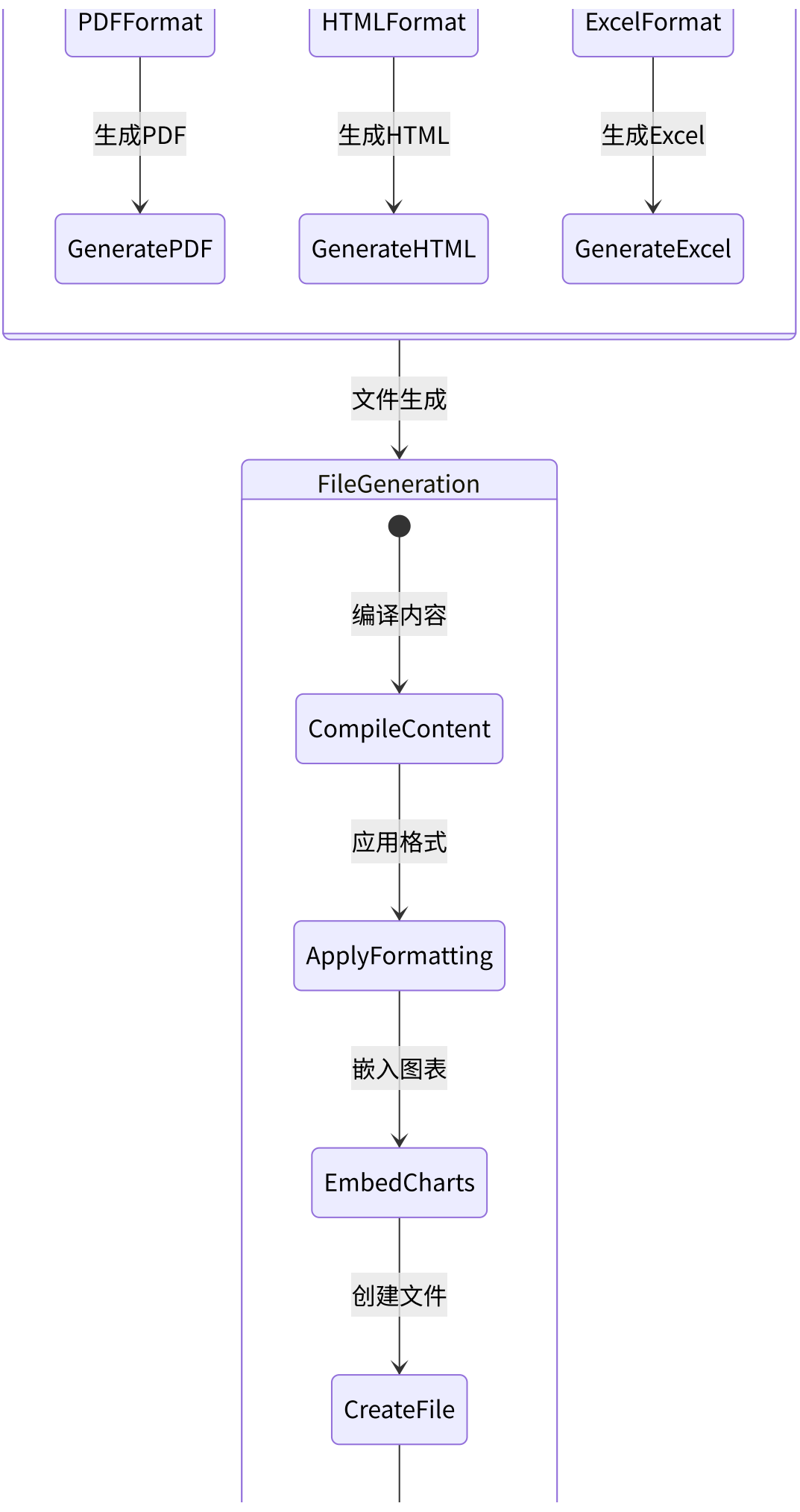
```python
# 会话状态管理
if 'analysis_results' not in st.session_state:
    st.session_state.analysis_results = None

# 响应用户操作
if st.button("开始分析"):
    with st.spinner("分析中..."):
        results = perform_analysis(data)
        st.session_state.analysis_results = results
```

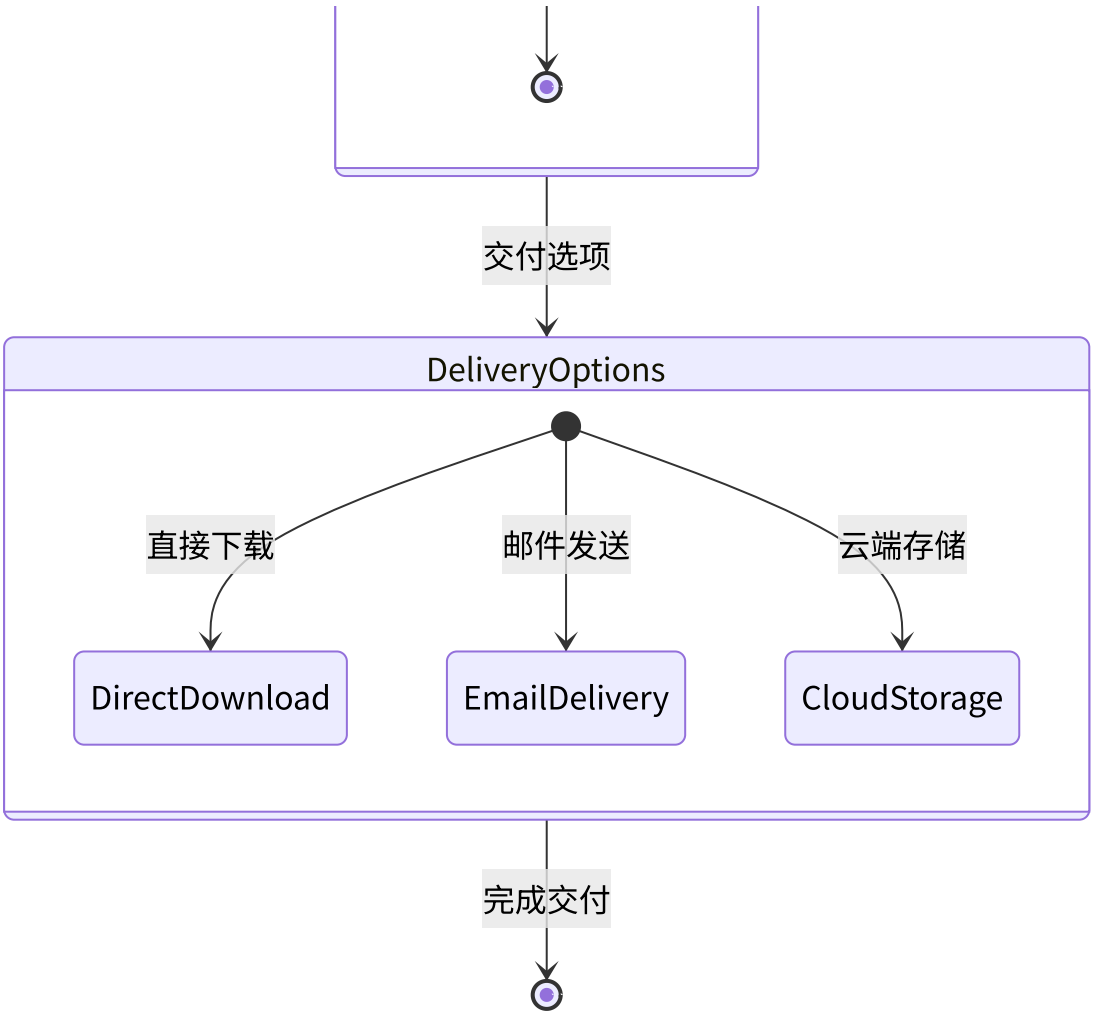
4. 进度指示与反馈
   - 使用 `st.progress()` 显示分析进度
   - 使用 `st.success()`, `st.error()`, `st.warning()` 提供操作反馈
   - 使用 `st.spinner()` 显示加载状态

图6：报告生成与导出流程


初始化报告模块
ReportInit
加载报告模板
LoadTemplate
设置格式化器
SetupFormatter
准备资源文件
PrepareAssets
生成报告内容
ContentGeneration
生成执行摘要
ExecutiveSummary


生成数据概览
DataOverview
生成分析结果
AnalysisResults
生成建议
Recommendations
生成附录
Appendices
选择输出格式
FormatSelection
PDF格式
HTML格式
Excel格式


PDFFormat
HTMLFormat
ExcelFormat
生成PDF
生成HTML
生成Excel
GeneratePDF
GenerateHTML
GenerateExcel
文件生成
FileGeneration
编译内容
CompileContent
应用格式
ApplyFormatting
嵌入图表
EmbedCharts
创建文件
CreateFile



详细介绍

基于项目报告生成功能的实现：

1. 报告模板系统

```
# HTML模板结构
template = """
<!DOCTYPE html>
<html>
<head>
    <title>医疗数据分析报告</title>
    <style>{css_styles}</style>
</head>
<body>
    <h1>分析报告</h1>
    <div class="summary">{executive_summary}</div>
    <div class="charts">{embedded_charts}</div>
</body>
</html>
```

```
15  """
```

2. 内容生成模块
   - **执行摘要**: 自动生成关键发现和结论
   - **数据概览**: 统计数据表格和基本信息
   - **分析结果**: 详细的分析过程和结果解释
   - **健康建议**: 基于分析结果的个性化建议
3. 多格式导出

```python
# PDF生成
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter

# Excel生成
import pandas as pd
with pd.ExcelWriter('report.xlsx') as writer:
    summary_df.to_excel(writer, sheet_name='摘要')
    results_df.to_excel(writer, sheet_name='详细结果')
```

4. 图表嵌入技术
   - 将 Plotly 图表转换为静态图片
   - 在 PDF 中嵌入高质量图表
   - 保持图表的交互性（HTML格式）
5. 下载与分享功能

```python
# Streamlit 下载按钮
st.download_button(
    label="下载PDF报告",
    data=pdf_buffer,
    file_name=f"医疗分析报告_{datetime.now().strftime('%Y%m%d')}.pdf",
    mime="application/pdf"
)
```