急救助手应用程序设计报告

1. 程序功能介绍

本急救助手应用程序是一个功能全面的医疗辅助工具，旨在为用户提供紧急情况下的医疗指导和支持。程序采用 Qt 框架开发，具有跨平台特性，主要功能模块包括:

1. **紧急救援指导模块**:
    - 提供常见伤害类型的急救指导（如休克、骨折、流鼻血等）
    - 从数据库加载伤害类型和症状信息
    - 点击列表项显示详细指导（包括文字说明和图片）
2. **症状自查模块**:
    - 用户输入体温、心率、呼吸频率等生理参数
    - 根据输入评估健康状况并给出反馈
3. **急救设备定位模块**:
    - 通过地址或经纬度定位最近的 AED 设备
    - 使用高德地图 API 进行地理编码（将地址转换为坐标）
    - 从本地文件加载 AED 设备位置数据（北京大学楼宇坐标）
4. **急救包整理模块**:
    - 分类展示急救包物品（文书类、包扎类、消毒类等）
    - 可展开/折叠查看各类物品详情
    - 用户可勾选物品以整理急救包
5. **急救学习模块**:
    - 提供多种急救知识的学习（共 27 类）
    - 通过菜单选择具体急救类型
    - 显示标题、步骤和常见问题（从 Markdown 文件加载）
6. **急救处理方案模块**:
    - 搜索急救知识库（基于 JSON 数据结构）
    - 显示搜索结果的详细急救步骤
    - 支持播放指导视频
7. **紧急呼叫模块**:
    - 点击按钮发送紧急呼叫信息到服务器
    - 获取当前位置（经度和纬度）并发送
    - 使用 TCP 协议与服务器通信

2. 项目各模块与类设计细节

2.1 主窗口模块（FirstAid）

类名：`MainWindow`（位于 FirstAid.cpp）

功能： - 程序的主窗口，包含七个功能按钮 - 建立与服务器的 TCP 连接 - 协调各功能模块的启动

设计细节： - 使用 `QTcpSocket` 进行网络通信 - 按钮点击事件触发相应功能窗口的显示 - `on_CallButton_clicked`：获取当前位置并发送到服务器 - `getLocation`：使用 Qt 的位置服务获取当前位置（经纬度）

关键代码：

```
void MainWindow::on_CallButton_clicked() {
    double lat, lon;
    QString error;
    if (getLocation(lat, lon, error)) {
        // 发送经纬度
    } else {
        // 发送默认位置
    }
    socket->write(ba);
}
```

2.2 紧急救援指导模块（EmergencyGuide）

类名：`EmergencyGuide`（位于 emergencyguide.cpp）

功能： - 显示伤害类型列表 - 从数据库加载伤害类型和症状信息 - 显示详细信息（包括症状和图片）

设计细节： - 使用 SQLite 数据库存储急救指导信息 - 初始化时从文本文件导入数据到数据库 - 使用 `QListWidget` 显示伤害类型列表 - 点击列表项时，从数据库查询详细信息并弹出对话框

关键代码：

```
void EmergencyGuide::setupDatabase() {
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("emergency_guide.db");
    // ... 创建表并导入数据
}
```

2.3 急救设备定位模块（AEDLocator）

类名：`AEDLocator`（位于 AEDLocator.cpp）

功能： - 地址转坐标功能 - 加载本地 AED 设备数据 - 查找最近的 AED 设备

设计细节： - 使用 `QNetworkAccessManager` 进行 HTTP 请求（高德地图 API） - 从文本文件加载 AED 设备数据（格式：ID|经度|纬度） - 查找最近 AED 设备：计算用户坐标与每个 AED 坐标的切比雪夫距离

关键代码：

```
AED AEDLocator::findNearestAED(double userLat, double userLon) {
    AED nearest = aedList.first();
    double minDist = qMax(qAbs(userLat - nearest.latitude),
                          qAbs(userLon - nearest.longitude));
    // ... 遍历比较
    return nearest;
}
```

2.4 症状自查模块（SymptomCheckDialog）

类名：`SymptomCheckDialog`（位于 symptomcheckdialog.cpp）

功能： - 收集用户生理参数 - 评估健康状况 - 显示评估结果

设计细节： - 使用 SQLite 数据库存储健康条件 - 输入验证确保数据有效性 - 多参数综合评估（体温、心率、呼吸频率）

关键代码：

```
void SymptomCheckDialog::evaluateHealth(double temperature, int heartRa
te, int respirationRate) {
    QString condition;
    if (temperature >= 38.5) condition += "高烧; ";
    // ... 其他判断
    QMessageBox::information(this, tr("健康状态"), condition);
}
```

2.5 急救包整理模块（Kits）

类名：`Kits`（位于 kits.cpp）

功能： - 分类展示急救包物品 - 展开/折叠物品列表 - 物品选择管理

设计细节： - 使用 `QToolButton` 显示类别标题 - 使用 `QCheckBox` 显示物品 - 动态切换展开/折叠状态 - 自定义样式美化界面

关键代码:

```
void Kits::on_btns_clicked(){
    QPushButton *button = qobject_cast<QPushButton *>(sender());
    int index = button->property("index").toInt();
    widgets[index]->setVisible(!widgets[index]->isVisible());
    // 切换按钮图标
}
```

2.6 急救学习模块（LearingMenu）

类名: `LearingMenu`（位于 learningmenu.cpp）

功能: - 显示急救知识菜单 - 加载并显示 Markdown 格式内容 - 分类展示急救知识

设计细节: - 使用 `QTextEdit` 显示 Markdown 内容 - 从文件系统加载 Markdown 文件 - 菜单动作绑定特定目录的内容 - 错误处理文件读取问题

关键代码:

```
void LearingMenu::on_action0_triggered() {
    ui->HeadName->setMarkdown(readFileContent("C:/FirstAidapp/knowledge
/0/HN.md"));
    // ... 加载步骤和问答
}

QString LearingMenu::readFileContent(const QString &filePath) {
    QFile file(filePath);
    if (!file.open(QIODevice::ReadOnly)) {
        return QString("无法打开文件: %1\n 错误: %2").arg(filePath, file.
errorString());
    }
    // ... 读取内容
}
```

2.7 急救处理方案模块（Widget）

类名: `Widget`（位于 widget.cpp）

功能: - 提供急救知识搜索 - 展示搜索结果 - 播放急救指导视频

设计细节: - 使用 `EmergencyModel` 类提供 JSON 数据 - 实现 `SearchEngine` 类进行智能搜索 - 动态生成搜索结果界面 - 集成媒体播放功能

关键代码:

```
void Widget::on_searchButton_clicked() {
    QString searchText=ui->lineEdit->text();
    searchWidget(searchText);
```

```
}

void Widget::searchWidget(const QString &query) {
    // 使用SearchEngine 搜索
    SearchEngine s(e->getJson());
    QJsonArray searchFinal = s.search(query);
    // 动态生成显示控件
}
```

2.8 数据模型（EmergencyModel）

类名：`EmergencyModel`（位于 emergencymodel.cpp）

功能： - 构建急救知识库 - 提供结构化急救知识数据 - 支持数据检索

设计细节： - 使用 `QJsonObject` 和 `QJsonArray` 构建数据结构 - 每种急救类型包含多个属性 - 支持视频链接

关键代码：

```
void EmergencyModel::createJson(){
    QJsonObject shock;
    shock["name"]="休克";
    shock["symptom"]=QJsonArray{"皮肤苍白","出冷汗","脉搏加快"};
    // ... 其他属性
    emergencyType.append(shock);
    // ... 添加其他急救类型
}
```

2.9 搜索引擎（SearchEngine）

类名：`SearchEngine`（位于 searchengine.cpp）

功能： - 实现急救知识搜索 - 计算搜索匹配度 - 排序搜索结果

设计细节： - 使用编辑距离（Levenshtein 距离）计算字符串相似度 - 使用 Jaccard 相似度计算关键词匹配度 - 多关键词支持（空格分隔） - 结果按匹配度排序

关键算法：

```
int SearchEngine::calculateScore(const QJsonValue & item, const QString
List &queryTerms) {
    int score=0;
    QJsonObject obj=item.toObject();
    // 检查名称
    QString nameString = obj["name"].toString();
    for (const QString &item: queryTerms) {
```

```
            if (item == nameString) {
                score += 10000;
                break;
            }
            // 计算编辑距离和Jaccard 相似度
            score += levenshteinDistance(item, nameString);
            score += 10 * jaccardSimilarity(item, nameString);
        }
        // 检查关键字
        QJsonArray keywords = obj["keywords"].toArray();
        // ... 类似计算
        // 加上紧急程度
        score += obj["urgency"].toInt();
        return score;
}
```

3. 项目分工方案

为高效推进项目开发，团队成员分工明确如下：

李文静（组长）：

急救知识库建设与管理： 负责急救知识的收集、存储与结构化组织，并设计实施学习效果评估机制（如学习后测试）。

急救包智能指导： 负责制定标准化急救包物品清单，并实现药品及耗材有效期的智能提醒功能。

主界面设计与实现： 负责应用程序主窗口的界面渲染与用户交互逻辑。

闫博睿：

紧急救援操作指导： 负责设计与实现图文并茂的紧急救援操作步骤展示与详细说明系统。

症状自查问卷系统： 负责构建交互式症状自查问卷流程，并实现用户回答与预设解决方案的智能匹配。

急救设备地理定位： 负责集成地图 API，实现自动体外除颤器(AED)等关键急救设备的定位与地图标记功能。

宋铭焕：

急救处理方案呈现： 负责整合急救处理方案，以视频演示结合结构化文字说明的形式进行清晰呈现。

智能症状识别引擎： 负责开发基于关键词匹配与分析的智能症状识别功能，根据用户输入症状提供初步判断。

云智能协同救护平台： 负责构建云端协同救护机制，实现实时消息推送与多方协作功能。

4. 项目总结与反思

4.1 项目亮点

1. **模块化设计**:
    - 将系统划分为功能独立的模块
    - 每个模块有明确的职责和接口
    - 便于团队协作和后续维护
2. **数据驱动**:
    - 使用多种数据存储方式（SQLite 数据库、JSON、文本文件）
    - 数据与代码分离，便于更新和维护
    - Markdown 文件支持富文本内容展示
3. **智能搜索功能**:
    - 实现基于相似度的搜索算法
    - 支持多关键词查询
    - 结果按相关度排序
4. **用户友好界面**:
    - 使用 Qt 框架实现美观的 GUI
    - 清晰的导航结构
    - 响应式设计适应不同屏幕尺寸
5. **多媒体支持**:
    - 集成视频播放功能
    - 支持图片展示
    - 音频设备自动检测

4.2 技术挑战与解决方案

1. **位置服务集成**:
    - 挑战：获取精确位置信息
    - 方案：使用 Qt 的位置服务 API，添加超时处理
2. **跨平台视频播放**:
    - 挑战：不同平台的媒体支持差异
    - 方案：使用 Qt 多媒体框架，自动检测音频设备
3. **搜索算法优化**:
    - 挑战：提高搜索准确性和效率
    - 方案：结合多种相似度算法（编辑距离+Jaccard 系数）

4. **数据库管理**：
    - 挑战：数据库初始化与更新
    - 方案：程序启动时检查并初始化数据库
5. **UI 性能优化**：
    - 挑战：动态生成大量 UI 元素时的性能
    - 方案：使用滚动区域和延迟加载

4.3 项目反思与改进方向

1. **定位精度问题**：
    - 当前使用切比雪夫距离计算最近 AED
    - 改进：使用更精确的球面距离公式（Haversine 公式）
2. **搜索性能优化**：
    - 当前线性搜索在大数据量下可能较慢
    - 改进：引入倒排索引提高搜索效率
3. **多语言支持**：
    - 当前仅支持中文界面
    - 改进：使用 Qt 的多语言翻译机制
4. **视频资源管理**：
    - 当前使用绝对路径，不便移植
    - 改进：使用资源文件或相对路径
5. **异常处理增强**：
    - 增加更多错误处理和用户反馈
    - 网络请求失败时的友好提示
6. **测试覆盖不足**：
    - 增加单元测试和集成测试
    - 使用 Qt Test 框架进行 UI 测试

4.4 项目收获

1. **Qt 框架深入理解**：
    - 掌握了 Qt 的信号槽机制
    - 熟练使用 Qt 的 GUI 组件和布局管理
    - 理解了 Qt 的多线程和网络编程
2. **软件开发流程实践**：
    - 需求分析与功能设计
    - 模块化开发与集成
    - 调试与性能优化

3. **算法应用能力**:
    - 相似度算法在实际应用中的实现
    - 地理空间计算方法的理解
    - 搜索排序算法的应用
4. **团队协作经验**:
    - 使用版本控制（Git）协作开发
    - 代码审查与质量保证
    - 文档编写与知识共享

4.5 未来扩展方向

1. **移动端适配**:
    - 开发 iOS 和 Android 版本
    - 利用手机传感器获取健康数据
2. **云服务集成**:
    - 用户数据云同步
    - 急救知识库在线更新
3. **增强现实(AR)指导**:
    - 使用 AR 技术提供实时急救指导
    - 3D 模型展示急救步骤
4. **人工智能辅助**:
    - 基于症状的智能诊断
    - 个性化急救方案推荐
5. **社区功能**:
    - 急救经验分享平台
    - 附近急救志愿者网络

结论

本急救助手应用程序通过整合多种急救相关功能，为用户提供了全面的急救支持。项目采用了模块化设计思想，实现了数据驱动的知识库管理，并开发了智能搜索算法提升用户体验。在开发过程中，团队克服了位置服务集成、跨平台媒体播放等技术挑战，积累了宝贵的开发经验。

虽然项目已实现基本功能，但仍存在改进空间，如定位精度提升、搜索性能优化和多语言支持等。未来可通过移动端适配、云服务集成和 AI 技术进一步增强应用价值。

本项目的成功开发不仅提供了实用的急救工具，也为类似医疗辅助应用的开发提供了可借鉴的架构和实现方案。