2022 级软件工程实训成绩单

姓名	边博宇	学号	3022244117
课程性质	必修	课程类别	集中实践
课程编号	02484	课程名称	综合实训（软件工程）
课程学分	8	实训单位	麒麟软件
开课日期	2025年4月28日-2025年6月20日		

考核项目	考核方式	A/B/C/D/F不通过
考勤	根据每天出勤情况评定，旷课3次或缺勤1/3以上按不通过计。	
平时表现及项目完成情况	根据团队合作、表达能力、遵守纪律、个人工作量、代码规范程度、项目完成度、系统演示综合评定。	
项目文档	根据项目文档（实训报告）评定	
总评		
备注		

校内指导教师签字：

天津大学智能与计算学部
实训任务书

实训题目：	
姓　　名：	
学　　号：	
实训单位：	
企业指导教师：	
校内指导教师：	
实训时间：	2025年4月28日-2025年6月20日
一、实训项目的目的和意义	
二、实训项目的主要内容	

三、实训时间安排

第1周:

第2周:

第3周:

第4周:

第5周:

第6周:

第7周:

第8周:

企业指导教师签字：　　2023　年　　月　　日

《软件工程综合实训》报告

题目：基于麒麟操作系统的多功能智能助手

实训单位 麒麟软件

姓　　名 边博宇

学　　号 3022244117

专　　业 软件工程

年　　级 2022 级

班　　级 软工 1 班

智能与计算学部

2025 年 6 月

目　录

《软件工程综合实训》报告 4

题目：基于麒麟操作系统的多功能智能助手 4

第一章 团队介绍 1

第二章 系统需求分析 3

2.1 项目背景 3
2.2 功能需求 3
2.2.1 自然语言问答 3
2.2.2 多轮对话上下文管理 3
2.2.3 内置角色 3
2.2.4 自定义角色 3
2.3 非功能需求 3
2.4 系统用例图 4
2.5 详细需求说明 4
2.5.1 用户交互界面设计 4
2.5.2 角色管理系统 4
2.5.3 日志与优化建议 5

第三章 系统概要设计 6

3.1 系统架构 6
3.1.1 表示层（Presentation Layer） 6
3.1.2 业务逻辑层（Business Logic Layer） 6
3.1.3 AI 服务层（AI Service Layer） 6
3.2 功能模块划分 6
3.2.1 聊天核心模块 6
3.2.2 角色管理模块 6
3.2.3 语音模块（预留接口） 7
3.2.4 UI 模块 7

第四章 系统详细设计 8

4.1 核心模块设计 8
4.2 详细设计说明 8
4.2.1 表示层设计 8
4.2.2 业务逻辑层设计 8
4.2.3 AI 服务层设计 9

第五章 系统实现 10

5.1 开发环境 10
5.2 个人负责的核心模块实现（每个成员写自己的分工核心） 10
5.2.1 聊天窗口创建（chatwindow） 10

5.2.2 主窗口设计（mainwindow） *10*
5.2.3 角色管理设计（rolemanager） *11*
5.2.4 新建角色窗口（customroledialog） *12*
5.3 界面实现 12
5.3.1 聊天窗口布局 *12*
5.3.2 消息显示逻辑 *13*
第六章 系统测试 **14**
6.1 项目概述 14
6.2 测试环境 14
6.3 测试内容与方法 14
6.3.1 文本对话功能测试 *14*
6.3.2 内置角色功能测试 *15*
6.3.3 自定义角色功能测试 *15*
6.3.4 角色管理功能测试 *16*
6.4 问题与改进建议 17
6.4.1 发现问题 *17*
6.4.2 改进建议 *17*
6.5 测试结论 17
第七章 系统用户手册 **18**
7.1 安装配置 18
7.1.1 配置依赖环境 *18*
7.1.2 如何执行项目并运用 *18*
7.2 界面导览 18
7.3 使用流程 19
7.3.1 开始对话 *19*
7.3.2 切换角色 *19*
7.3.3 创建自定义角色 *20*
第八章 项目总结报告 **21**
8.1 项目成果总结 21
8.1.1 核心功能实现 *21*
8.1.2 技术突破 *21*
8.2 创新性技术实践 21
8.2.1 角色感知对话系统 *21*
8.2.2 智能上下文管理： *21*
8.3 项目价值分析 21
8.3.1 技术价值 *22*
8.3.2 应用价值 *22*
8.3.3 生态价值 *22*
8.4 项目心得 22
8.5 致谢 22

第一章 团队介绍

1.1 团队成员分工

• 刘文翔

负责整个系统的架构设计以及核心模块的开发与集成工作。任务包括但不限于：

系统架构设计：基于麒麟 AI SDK，设计一个分层架构，确保系统的可扩展性、可维护性和高效性。

聊天核心模块开发：开发并优化智能问答助手的核心功能，包括但不限于用户提问的理解与解析、从文档库中检索相关信息、生成准确的回答等。此外，还需保证该模块能够稳定运行，并具有良好的性能表现。

角色管理模块：虽然这部分工作主要由陈天韵负责，但刘文翔也会参与其中，特别是在涉及到与聊天核心模块交互的部分，例如如何让不同的角色根据用户的输入提供个性化的回答。

系统集成与测试：将各个独立开发的功能模块整合到一起，进行整体调试，确保各部分之间无缝协作。同时，他还需要制定详细的测试计划，涵盖测试等多个层面，确保最终交付的产品质量可靠。

• 陈天韵

负责麒麟 AI SDK 的集成及其相关功能的开发,特别是角色管理模块的设计与实现。具体职责如下：

麒麟 AI SDK 集成：深入了解麒麟 AI SDK 的各项功能，包括但不限于文字识别、语音处理、向量化等功能接口的使用方法。在此基础上，完成 SDK 与项目的对接，确保各项 AI 能力能够顺利应用于本项目中。

角色管理模块设计与实现：开发至少四种内置角色（律师、教师、程序员、作家），每个角色都需具备独特的对话风格和专业知识。此外，还需要实现自定义角色创建功能，允许用户根据个人喜好和特定需求定制专属角色。这项工作的难点在于如何使不同角色之间的切换既流畅又自然，同时还要保证生成的回答符合各自的角色特征。

• 边博宇

承担 UI 界面设计与实现的任务，同时也负责项目的测试与文档编写工作。具体职责：

UI 设计：设计直观易用的用户界面，既要满足美观的要求，又要注重实用性。具体来说，需要为用户提供一个简洁明了的操作界面，方便他们输入问题、查看答案以及切换角色。此外，还应考虑到多模态交互的需求，如支持语音输入输出，增强用户

的互动体验。

前端开发：基于选定的设计方案，使用合适 Qt 开发工具实现上述 UI 界面。在开发过程中，要特别注意响应式设计，确保界面能够有良好的显示效果。

测试与文档编写：制定全面的测试计划，覆盖所有功能模块，确保产品在各种情况下都能正常工作。对于发现的问题，及时反馈给相关人员进行修复。同时，他还需撰写详尽的技术文档，记录项目的开发过程、遇到的问题及解决方案，为后续维护提供参考依据。

第二章 系统需求分析

2.1 项目背景

本项目旨在基于银河麒麟操作系统，利用麒麟 AI SDK 开发一款多功能智能助手。该智能助手不仅支持文本对话和语音对话功能，还具备角色切换与自定义角色创建的能力，以满足不同场景下的多样化需求，并生成高质量的回答，为用户提供高效的服务。

2.2 功能需求

2.2.1 自然语言问答

用户可以通过输入自然语言提问，系统应能准确理解并给出合理的回答。

2.2.2 多轮对话上下文管理

系统需要维护对话的上下文，以便在多轮对话中保持连贯性和逻辑性，确保根据上下文准确理解用户意图。

2.2.3 内置角色

至少提供四种内置角色（律师、教师、程序员、作家），每个角色都有独特的对话风格和专业知识。扩展后的版本将增加护士角色。

律师角色能够解答常见的法律问题，提供法律建议。

教师角色可进行知识讲解、辅导学习。

程序员角色能协助代码问题排查、提供编程建议。

作家角色能进行创意写作、文章润色。

2.2.4 自定义角色

允许用户根据个人喜好和特定需求创建专属角色，包括但不限于输入角色描述、对话风格示例等信息。

2.3 非功能需求

类型	要求
可靠性	尽量降低系统的故障率，确保99%以上的可用性。
性能	响应时间尽量短，目标是在500毫秒内完成用户的请求处理。
易用性	用户界面友好，操作直观，减少用户的学习成本。
可扩展性	支持新增AI模型和功能模块的无缝接入，适应未来的业务发展需求。

图 2-1　非功能需求图

2.4 系统用例图

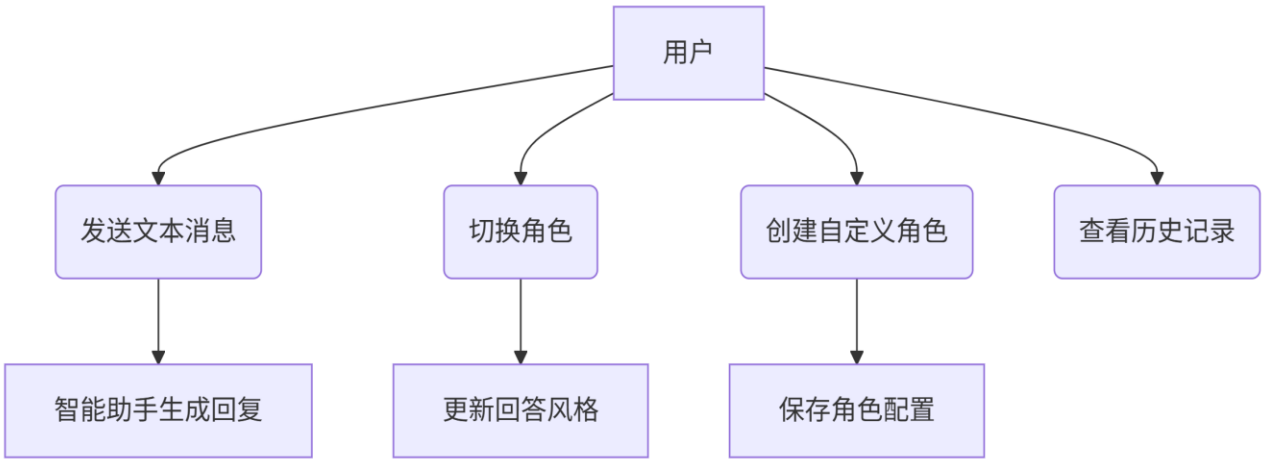


图 2-2　系统用例图

2.5 详细需求说明

2.5.1 用户交互界面设计

设计简洁直观的用户界面，支持用户输入问题，并以清晰易读的方式展示生成的回答及参考文档来源。实现多模态交互，例如支持语音输入问题，语音播报回答结果，提升用户使用的便捷性。

2.5.2 角色管理系统

开发至少 4 种内置角色（律师、教师、程序员、作家），每个角色都有独特的对话

风格和专业知识。提供自定义角色创建功能，用户可通过输入角色描述、对话风格示例等信息创建自定义角色，使助手能够根据用户设定的规则进行对话，展现出独特的风格和特点。

2.5.3 日志与优化建议

根据日志分析结果，为系统管理员提供针对性的优化建议。例如，针对 CPU 使用率过高的情况，建议优化进程调度、关闭不必要的服务；当系统出现潜在风险或异常情况时，及时发出预警信息。预警方式采用弹窗提示，确保管理员能够第一时间知晓并采取应对措施尽量减少误报情况。

第三章 系统概要设计

3.1 系统架构

为了实现多功能智能助手的高效开发与维护，系统采用分层架构。各层之间职责明确，便于模块化开发和扩展。以下是具体的层次划分及其功能描述：

3.1.1 表示层（Presentation Layer）

Qt UI 界面：使用 Qt 框架构建用户友好的图形界面，支持多模态交互（文本输入、语音输入等）。界面设计需简洁直观，确保用户能够轻松上手并高效使用各项功能。

3.1.2 业务逻辑层（Business Logic Layer）

消息收发：负责处理用户的提问及系统的回答，确保信息传递的准

角色管理模块：

角色增删改查：允许用户创建、删除、修改和查询自定义角色，每个角色都有独特的对话风格和专业知识。

提示词管理：为不同角色设置特定的提示词，帮助 AI 更好地理解用户意图，生成符合角色特征的回答。

3.1.3 AI 服务层（AI Service Layer）

麒麟 AI SDK：集成麒麟 AI SDK 提供的各种 AI 能力，如自然语言处理、语音识别与合成等。该层主要负责调用 AI 模型进行信息检索、问题回答以及角色相关的语义理解等任务。

3.2 功能模块划分

3.2.1 聊天核心模块

消息收发：实现用户提问与系统回答之间的通信机制，确保消息的实时传递。包括但不限于文本输入、语音输入等多种交互方式的支持。

历史管理：自动保存所有对话记录，并提供方便的历史记录查看接口，用户可以随时查阅过去的对话内容。此外，还需支持按角色分类的历史记录管理，方便用户根据不同角色回顾互动情况。

3.2.2 角色管理模块

角色增删改查：提供一套完整的角色管理机制，用户可以根据自身需求创建、编辑、删除或查询角色。每个角色都应具备独立的对话风格和专业知识领域。

提示词管理：为每个角色设定特定的提示词，帮助 AI 在处理用户请求时更准确地

理解意图，并生成符合角色特征的回答。这有助于提升用户体验，使对话更加自然流畅。

3.2.3 语音模块（预留接口）

语音识别与合成：虽然目前主要以文本对话为主，但考虑到未来的扩展需求，预留了语音识别与合成的相关接口。未来可以通过集成第三方语音服务（如百度语音、科大讯飞等），实现语音输入输出功能，进一步丰富用户体验。

3.2.4 UI 模块

窗口管理：负责管理应用程序的所有窗口，包括主界面、设置界面、角色管理界面等。确保各个窗口之间的切换平滑且无冲突。

交互控制：设计直观易用的操作控件，如按钮、输入框、下拉菜单等，简化用户操作流程，降低学习成本。同时，还需考虑响应式设计，确保界面在不同设备上的显示效果一致。

第四章 系统详细设计

4.1 核心模块设计

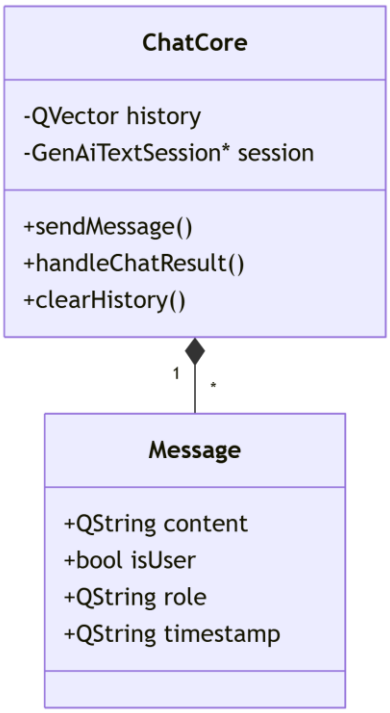


图 4-1　聊天核心类图

4.2 详细设计说明

4.2.1 表示层设计

• Qt UI 界面：采用 Qt 框架进行 UI 设计，充分利用其丰富的组件库和强大的信号槽机制，实现高效稳定的用户界面。界面设计应遵循现代设计理念，注重用户体验，减少不必要的复杂度。

• 多模态交互支持：除了传统的文本输入外，还应支持语音输入和语音播报功能。通过集成麒麟 AI SDK 中的语音识别与合成 API，实现从语音到文本、文本到语音的转换，增强用户交互体验。

4.2.2 业务逻辑层设计

• 聊天核心模块：该模块是整个系统的核心部分，负责处理用户的提问及生成相应的回答。通过调用 AI 服务层的功能，结合上下文信息，生成高质量的回答。

• 角色管理模块：该模块提供了角色的增删改查功能，允许用户根据自己的需求定

制专属角色。每个角色都有独立的对话风格和专业知识，增强了系统的灵活性和个性化服务能力。

4.2.3 AI 服务层设计

• 麒麟 AI SDK：作为系统的重要组成部分，麒麟 AI SDK 提供了强大的 AI 能力支持。通过合理调用其提供的 API 接口，实现对用户提问的理解、文档检索、答案生成等功能，显著提升了系统的智能化水平。

第五章 系统实现

5.1 开发环境

组件	版本
操作系统	银河麒麟 V2503
开发框架	Qt 5.15.2
编译器	GCC 9.4
AI SDK	麒麟AI SDK 1.1

图 4-1　聊天核心类图

5.2 个人负责的核心模块实现（每个成员写自己的分工核心）

5.2.1 聊天窗口创建（chatwindow）

```cpp
#include "chatwindow.h"
#include "ui_chatwindow.h"
#include <QDateTime>
#include <QMessageBox>
#include <QScrollBar>
#include <QIcon>
#include <QDebug>
#include "customroledialog.h"

ChatWindow::ChatWindow(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::ChatWindow)
    , chatCore(new ChatCore(this))
    , voiceHandler(new VoiceHandler(this))
    , currentRole("默认")
    , roleManager(nullptr)
{
    ui->setupUi(this);

    // 设置按钮图标
    ui->sendButton->setIcon(QIcon(":/resources/icons/send.png"));
    ui->voiceButton->setIcon(QIcon(":/resources/icons/voice.png"));

    // 初始化聊天核心
    if (!chatCore->initialize()) {
        QMessageBox::critical(this, "错误", "聊天核心初始化失败");
    }

    // 连接信号和槽
    connect(ui->sendButton, &QPushButton::clicked, this, &ChatWindow::on_sendButton_clicked);
    connect(ui->voiceButton, &QPushButton::clicked, this, &ChatWindow::handleVoiceButtonClicked);
    connect(ui->newRoleButton, &QPushButton::clicked, this, &ChatWindow::on_newRoleButton_clicked);
    connect(chatCore, &ChatCore::messageReceived, this, &ChatWindow::handleMessageReceived);
    connect(chatCore, &ChatCore::errorOccurred, this, &ChatWindow::handleError);
    connect(voiceHandler, &VoiceHandler::textRecognized, this, &ChatWindow::handleTextRecognized);
    connect(voiceHandler, &VoiceHandler::errorOccurred, this, &ChatWindow::handleError);
    connect(ui->roleComboBox, QOverload<int>::of(&QComboBox::currentIndexChanged),
            this, &ChatWindow::on_roleComboBox_currentIndexChanged);

    // 设置占位符文本
    ui->messageEdit->setPlaceholderText("请您输入消息...");
```

```cpp
#ifndef CHATWINDOW_H
#define CHATWINDOW_H

#include <QWidget>
#include <QString>
#include <QComboBox>
#include <QListWidgetItem>
#include "chatcore.h"
#include "voicehandler.h"
#include "customrole.h"
#include "rolemanager.h"
#include "customroledialog.h"
#include "ui_chatwindow.h"  // 确保正确包含自动生成的 UI 类

class RoleManager;
class ChatWindow : public QWidget
{
    Q_OBJECT

public:
    explicit ChatWindow(QWidget *parent = nullptr);
    ~ChatWindow();

    void appendMessage(const QString &sender, const QString &text, const QString &avatarPath = "");
    void clearChatHistory();
    QString getCurrentRole() const;
    void setRoleManager(RoleManager *manager);
    void loadChatHistory(const QString &roleName);

    // 获取当前角色的提示词
    QString getCurrentRolePrompt() const;

public slots:
    void setCurrentRole(const QString &role);
    void updateRoleList(const QStringList &roles);
    void onRoleSelected(const QString &roleName); // Slot to handle role selection from MainWindow

    // Added back from .cpp connections
    void handleMessageReceived(const Message &message);
    void handleError(const QString &error);
    void handleTextRecognized(const QString &text);
```

5.2.2 主窗口设计（mainwindow）

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "chatwindow.h"
#include "rolemanager.h"
#include <QMessageBox>
#include <QIcon>
#include <QDebug>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
    , chatPage(nullptr)
    , rolePage(nullptr)
{
    ui->setupUi(this);
    setupUi();
    setupConnections();

    // 设置菜单图标
    ui->actionSwitchToChat->setIcon(QIcon(":/resources/icons/chat.png"));
    ui->actionSwitchToRole->setIcon(QIcon(":/resources/icons/role.png"));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::setupUi()
{
    stackedWidget = ui->stackedWidget;
    chatPage = ui->chatPage;
    rolePage = ui->rolePage;
    chatWindow = ui->chatWindow; // 确保这是 ChatWindow 类型
    roleManager = ui->roleManager;

    stackedWidget->addWidget(rolePage);
    stackedWidget->addWidget(chatPage);

    switchToChat();
```

```cpp
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QStackedWidget>
#include <QWidget>
#include "ui_chatwindow.h"  // 确保路径正确
#include "rolemanager.h"
#include "customrole.h"
#include "customroledialog.h"
#include "chatwindow.h"   // 包含 ChatWindow 的定义

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void switchToChat();
    void switchToRoleManager();
    void handleRoleSelected(const QString &roleName);
    void handleNewRole(const RoleData &data);
    void updateRoleList(); // 添加声明

private:
    Ui::MainWindow *ui;
    QStackedWidget *stackedWidget;
    QWidget *chatPage;
    QWidget *rolePage;
    ChatWindow *chatWindow;
```

5.2.3 角色管理设计（rolemanager）

```cpp
#include "rolemanager.h"
#include <QFile>
#include <QJsonDocument>
#include <QJsonObject>
#include <QJsonArray>
#include <QDir>
#include <QListWidgetItem>
#include <QMessageBox>
#include <QPushButton>

#define ROLES_FILE "roles.json"

RoleManager::RoleManager(QWidget *parent)
    : QWidget(parent),
      ui(new Ui::RoleManager)
{
    ui->setupUi(this);

    // 连接信号槽（控件名称必须与.ui文件完全一致）
    connect(ui->addButton, &QPushButton::clicked, this, &RoleManager::onAddButtonClicked);
    connect(ui->removeButton, &QPushButton::clicked, this, &RoleManager::onRemoveButtonClicked);
    connect(ui->saveButton, &QPushButton::clicked, this, &RoleManager::onSaveButtonClicked);
    connect(ui->rolelist, &QListWidget::itemClicked, this, &RoleManager::onRoleSelected);

    initializeDefaultRoles();
    loadCustomRoles();
    updateRoleList();
}

RoleManager::~RoleManager()
{
    saveCustomRoles();
    qDeleteAll(roles);
    delete ui;
}

void RoleManager::addCustomRole(const QString &name, const QString &description, const QString
{
    if (roles.contains(name)) {
        emit errorOccurred("角色名称已存在");
        return;
    }
```

```cpp
#ifndef ROLEMANAGER_H
#define ROLEMANAGER_H

#include <QWidget>
#include <QMap>
#include <QString>
#include <QListWidget>
#include "customrole.h"
#include "ui_rolemanager.h"


class RoleManager : public QWidget
{
    Q_OBJECT

public:
    explicit RoleManager(QWidget *parent = nullptr);
    ~RoleManager();

    // 角色管理方法
    void addCustomRole(const QString &name, const QString &description, const QString &prompt);
    bool addRole(const QString &name, const QString &description, const QString &prompt);
    void removeRole(const QString &name);
    void editRole(const QString &name, const QString &description, const QString &prompt);
    QStringList getRoleList() const;
    CustomRole* getRole(const QString &name) const;

signals:
    void roleAdded(const QString &name);
    void roleRemoved(const QString &name);
    void roleModified(const QString &name);
    void roleSelected(const QString &name);
    void errorOccurred(const QString &error);

private slots:
    void onAddButtonClicked();
    void onRemoveButtonClicked();
    void onSaveButtonClicked();
    void onRoleSelected(QListWidgetItem *item);

private:
```

5.2.4 新建角色窗口（customroledialog）

```cpp
边博宇, 前天 | 1 author (边博宇)
#include "customroledialog.h"        边博宇, 前天 • 1 ···
#include "ui_customroledialog.h"
#include <QMessageBox>
#include <QDebug>
#include "customroledialog.h"

CustomRoleDialog::CustomRoleDialog(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::CustomRoleDialog)
    , isEditMode(false)
{
    ui->setupUi(this);

    // 设置窗口属性
    setWindowTitle("新建自定义角色");
    setModal(true);
}

CustomRoleDialog::~CustomRoleDialog()
{
    delete ui;
}

RoleData CustomRoleDialog::getRoleData() const
{
    RoleData data;
    data.name = ui->nameLineEdit->text();
    data.description = ui->descriptionLineEdit->text();
    data.prompt = ui->promptTextEdit->toPlainText();
    return data;
}

void CustomRoleDialog::setRoleData(const RoleData &data)
{
    ui->nameLineEdit->setText(data.name);
    ui->descriptionLineEdit->setText(data.description);
    ui->promptTextEdit->setText(data.prompt);
}
```

```cpp
边博宇, 前天 | 1 author (边博宇)
#ifndef CUSTOMROLEDIALOG_H        边博宇, 前天 • 1 ···
#define CUSTOMROLEDIALOG_H

#include <QDialog>
#include <QString>

边博宇, 前天 | 1 author (边博宇)
namespace Ui {
class CustomRoleDialog;
}

// 使用简单结构体来传递角色数据，避免 QObject 派生类复制问题
边博宇, 前天 | 1 author (边博宇)
struct RoleData {
    QString name;
    QString description;
    QString prompt;
};


边博宇, 前天 | 1 author (边博宇)
class CustomRoleDialog : public QDialog
{
    Q_OBJECT

public:
    explicit CustomRoleDialog(QWidget *parent = nullptr);
    ~CustomRoleDialog();

    // 获取编辑后的角色
    RoleData getRoleData() const;

    // 设置当前编辑的角色
    void setRoleData(const RoleData &data);

    // 设置对话框为编辑模式
    void setEditMode(bool isEdit);

private slots:
```

5.3 界面实现

5.3.1 聊天窗口布局

```
<QWidget>
        <QVBoxLayout>
            <QComboBox name="roleComboBox"/>
            <QTextEdit name="chatDisplay" readonly="true"/>
            <QTextEdit name="messageEdit"/>
            <QPushButton name="sendButton" text="发送"/>
            <QPushButton name="newRoleButton" text="新建角色"/>
        </QVBoxLayout>
</QWidget>
```

5.3.2 消息显示逻辑

```
void ChatWindow::displayMessage(const Message &msg) {
    QString html;
    if (msg.isUser) {
        html = "<div align='right' style='color:blue;'>[我] " + msg.content + "</div>";
    } else {
        html = "<div align='left' style='color:green;'>[" + msg.role + "] " + msg.content +
"</div>";
    }
    ui->chatDisplay->append(html);
}
```

第六章 系统测试

6.1 项目概述

本项目是基于银河麒麟操作系统开发的"多功能 AI 交互助手"，利用麒麟 AI SDK 提供的 AI 能力，实现了一个支持文本对话、语音交互、内置多种角色和自定义角色功能的智能助手系统。

6.2 测试环境

6.2.1 测试环境

开发环境：vscode

开发语言：C/C++

开发工具：CMake Tools

UI 实现：Qt 设计器、UIC

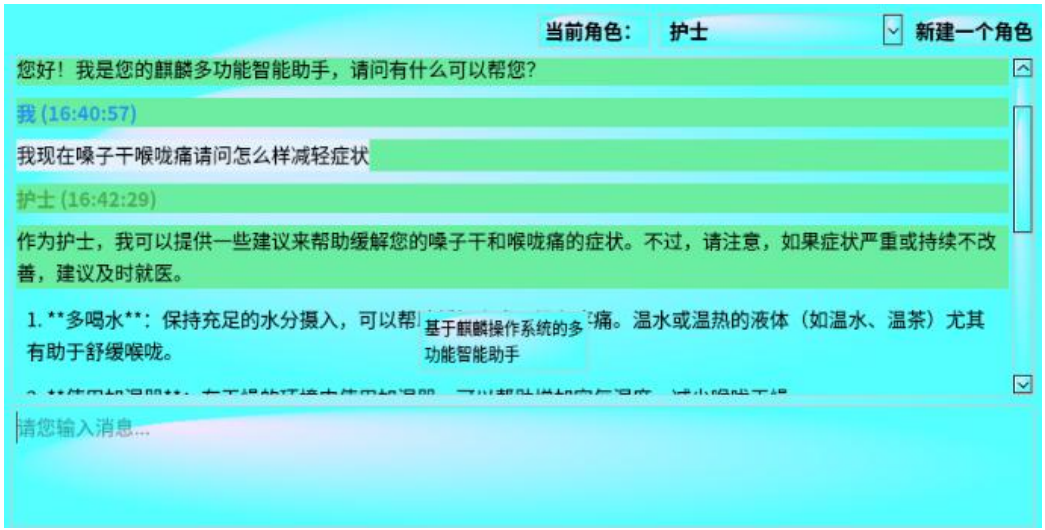
6.3 测试内容与方法

6.3.1 文本对话功能测试

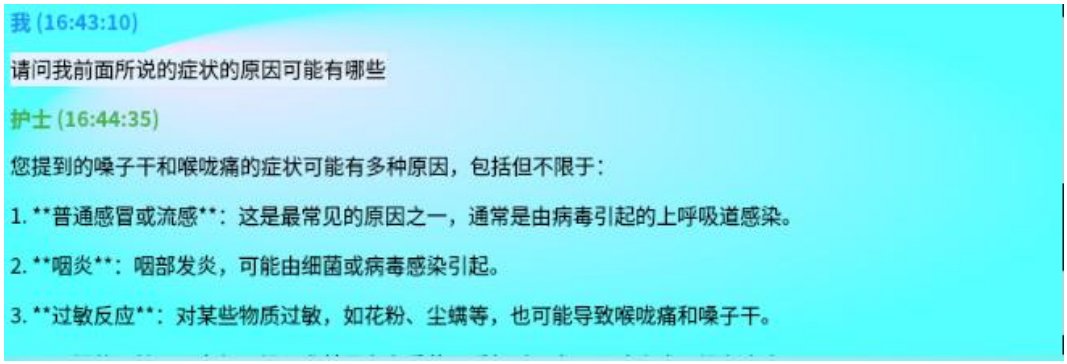
测试用例：

- 多轮对话测试：连续提问相关问题
- 长文本理解测试：输入较长的问题文本

测试结果：支持多轮对话，长文本理解能力较好



(a)



(b)

图 6-1　文本对话功能测试图

6.3.2 内置角色功能测试

测试用例：程序员角色：询问"斐波那契数列的 c++代码"

测试结果：符合程序员角色回答风格

图 6-2　内置角色功能测试图

6.3.3 自定义角色功能测试

测试用例：创建"健身教练"角色并提问

测试结果：角色设定较符合，回答内容相关性较好，角色风格一致性较好

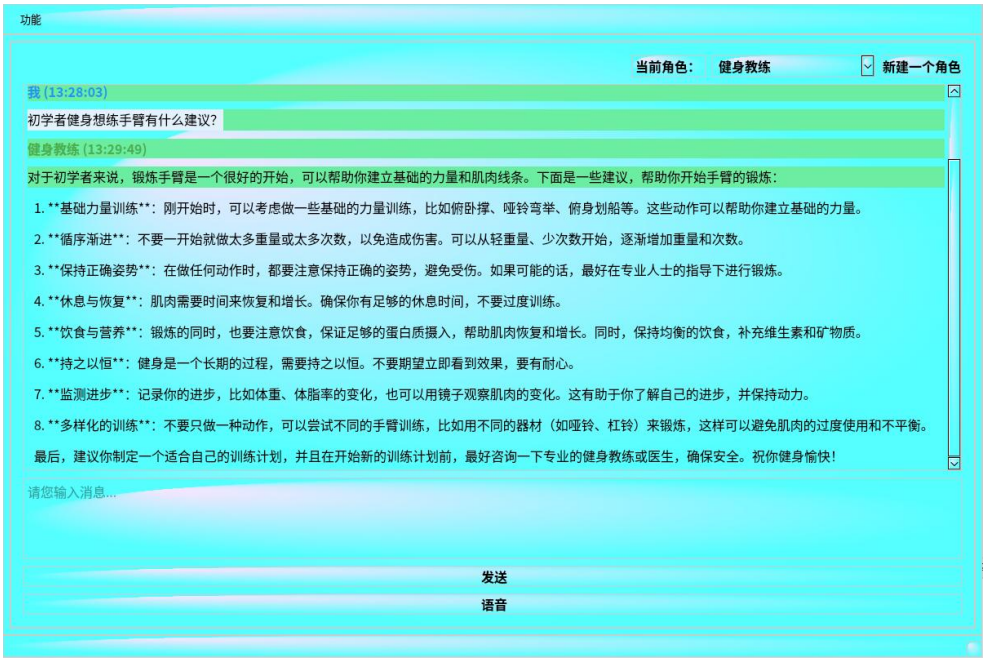
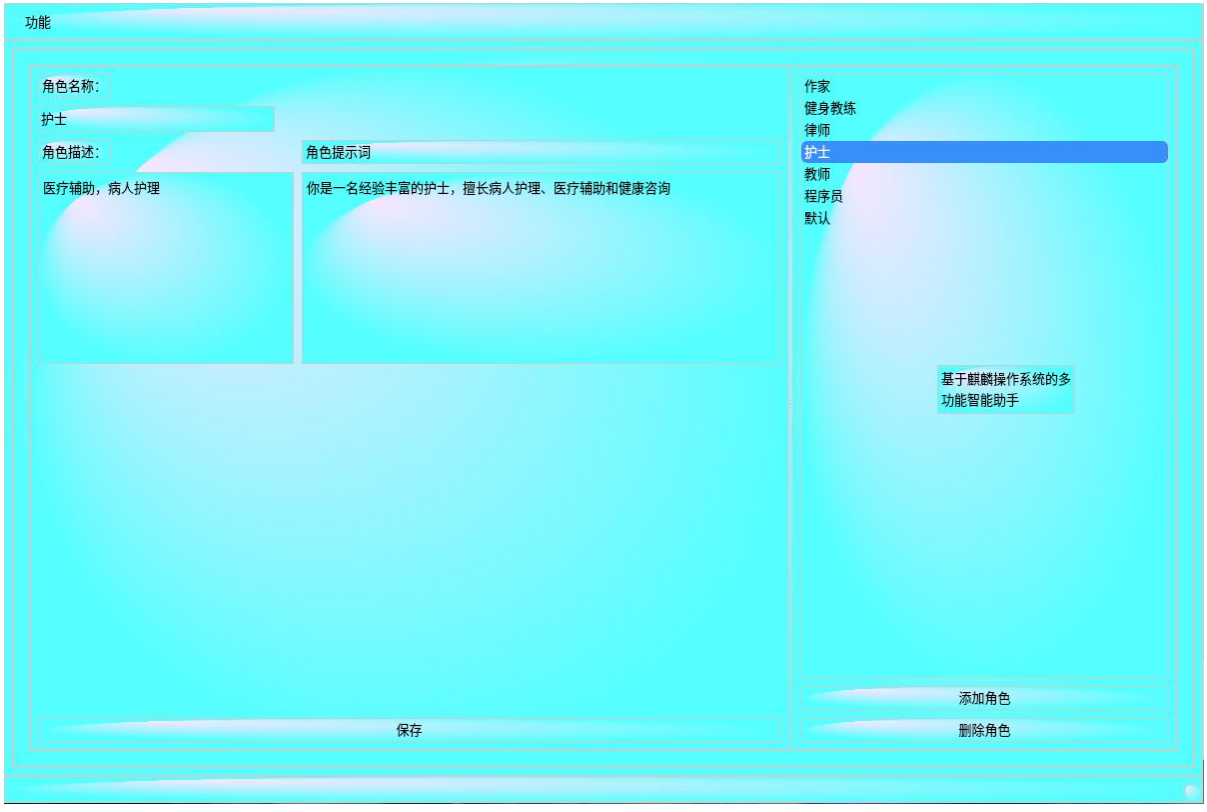


图 6-3　自定义角色功能测试图

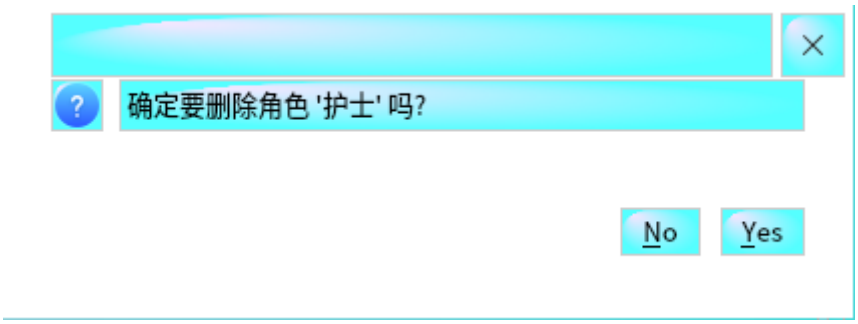
6.3.4 角色管理功能测试

测试用例：创建“护士”角色，删除“护士”角色

测试结果：角色管理示例成功



(a)



(b)

图 6-4　角色管理功能测试图

6.4 问题与改进建议

6.4.1 发现问题

语音功能需要调用云端模型，目前并未对此功能进行进一步开发。

高并发情况下响应时间明显增加

部分专业领域回答不够深入

6.4.2 改进建议

增加本地缓存机制，提高并发性能

接入更专业的领域知识库，提升回答质量

6.5 测试结论

本次测试的多功能 AI 交互助手基本实现了题目要求的除语音处理以外的所有功能，包括文本对话、内置角色和自定义角色功能。系统在银河麒麟操作系统上运行稳定，各功能模块表现良好，达到了预期效果。针对测试中发现的问题，建议进行进一步优化以提高用户体验。

总体评价：系统功能完整，性能达标，具备实用价值。

第七章 系统用户手册

7.1 安装配置

7.1.1 配置依赖环境

- 安装麒麟操作系统，麒麟桌面操作系统 2503 版本下载链接。

ARM 架构：

https://iso.kylinos.cn/web_pungi/download/cdn/i8JVstPLwnSjzaFpgr6f74mdhBlq1N5Y/Kylin-Desktop-V10-SP1-2503-Release-20250430-ARM64.iso

X86 架构：

https://iso.kylinos.cn/web_pungi/download/cdn/oLNq59PmxHAe08TrE6I2bhfl3jRJynBt/Kylin-Desktop-V10-SP1-2503-HWE-PP-Release-20250430-X86_64.iso

- 配置执行环境：配置好相关 c++依赖，qt 依赖
- 配置麒麟 sdk：

```
sudo apt update && sudo apt install -y \
    git cmake build-essential \
    qt6-base-dev libqt6multimedia6 \
    libasound2-dev pulseaudio \
    libkylin-ai-sdk-dev    # 麒麟 AI SDK
```

7.1.2 如何执行项目并运用

- 完成上述依赖配置后，可以直接执行改文件

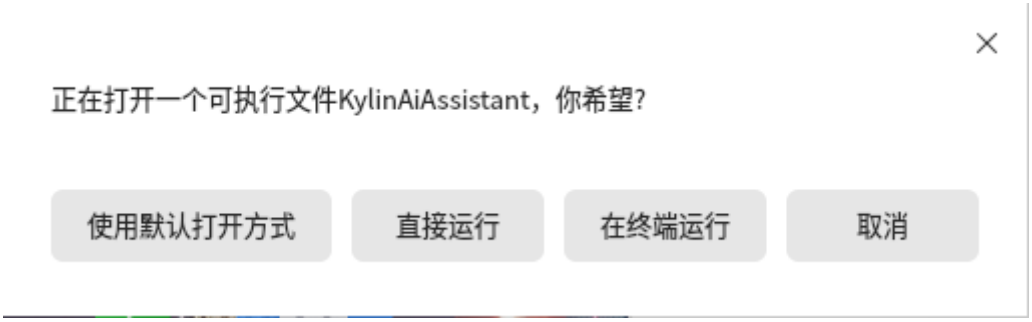


图 7-1 执行 KylinAiAssistant 图

7.2 界面导览

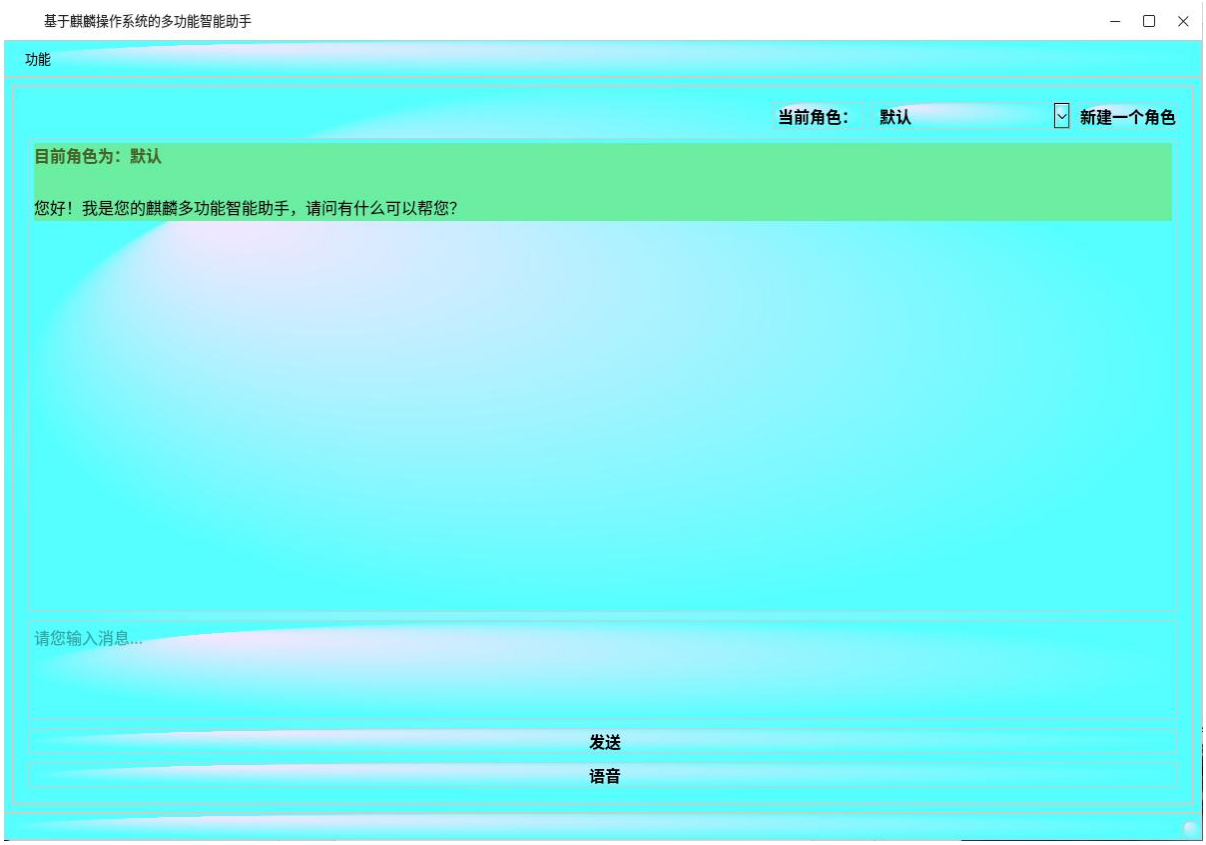


图 7-2　界面导览图

7.3 使用流程

7.3.1 开始对话

- 在消息框输入内容，点击“发送”
- 系统生成回答并显示

7.3.2 切换角色

- 从角色下拉框选择律师/教师等
- 新消息将应用角色特性

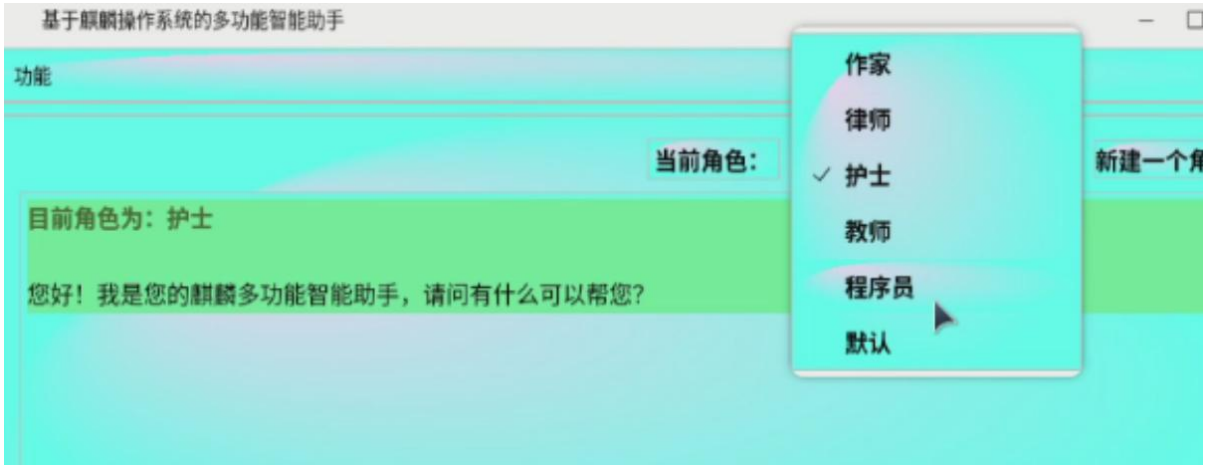


图 7-3　角色切换图

7.3.3 创建自定义角色

- 点击“新建角色”按钮
- 填写名称、描述和提示词

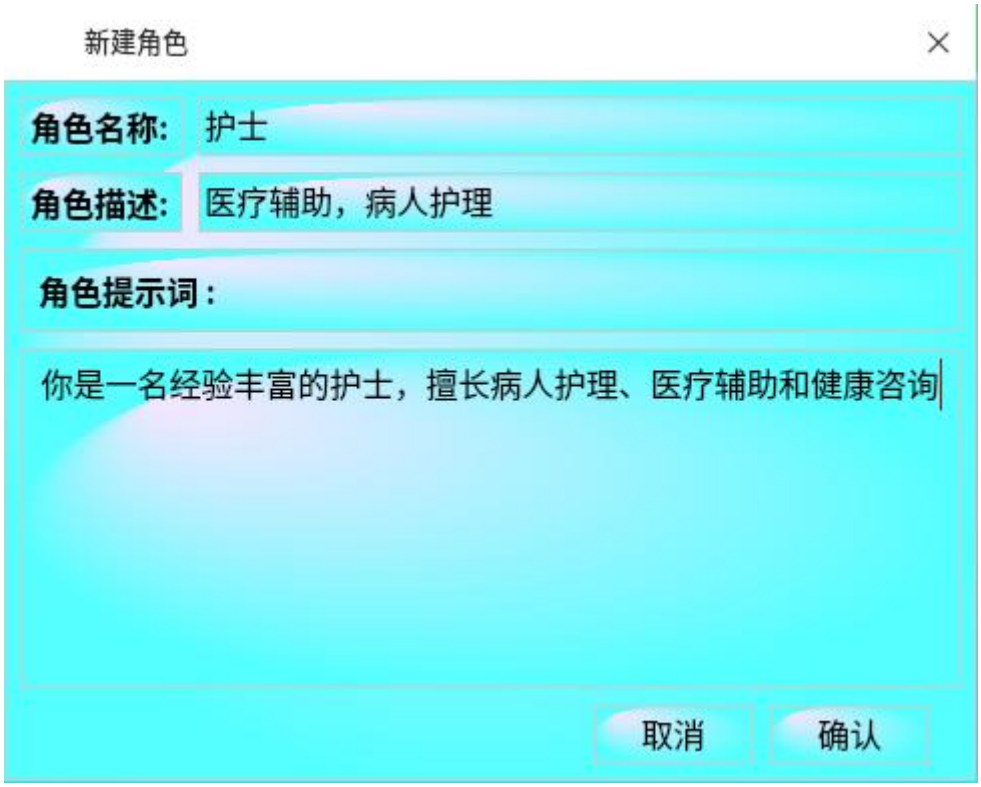


图 7-4　自定义角色创建图

第八章 项目总结报告

8.1 项目成果总结

本项目成功实现了基于银河麒麟操作系统的多功能智能助手系统，主要成果包括：

8.1.1 核心功能实现

• 文本对话系统：集成麒麟 AI SDK 的本地大语言模型（Qwen-2.5-3b），实现高质量对话生成

• 角色管理系统：开发 5 种内置专业角色（律师、教师、程序员、作家、护士）和自定义角色功能

8.1.2 技术突破

- 麒麟 AI SDK 深度集成：完成 C++层到 Qt 应用层的完整封装
- 动态提示词注入：实现角色提示词的实时切换和应用
- 异步消息处理：解决 AI 响应延迟问题，保证 UI 流畅性

8.2 创新性技术实践

8.2.1 角色感知对话系统

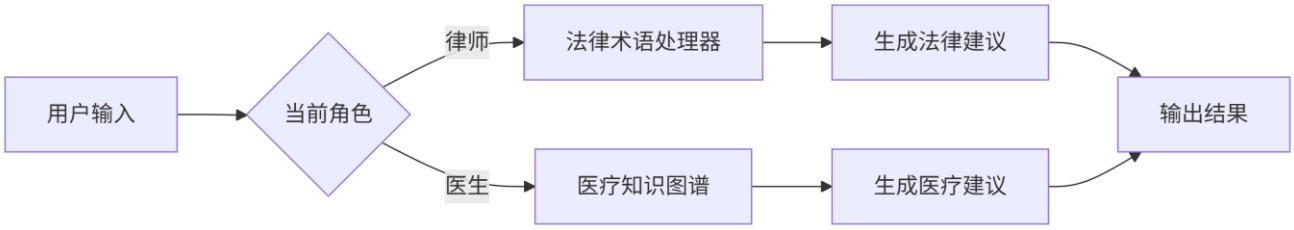


图 8.1　角色感知对话系统图

8.2.2 智能上下文管理：

采用滑动窗口算法保留最近 3 轮对话

关键信息提取与持久化存储

角色专属上下文隔离

8.3 项目价值分析

8.3.1 技术价值

验证了国产操作系统+国产 AI 框架的技术可行性

探索出麒麟 AI SDK 的最佳实践方案

建立了角色化 AI 交互的系统架构

8.3.2 应用价值

表 8-1　应用价值表

应用场景	实现功能	用户价值
法律咨询	法律条文解析	降低法律咨询成本
教育辅导	知识点讲解	个性化学习支持
编程辅助	代码问题排查	提高开发效率
创意写作	内容生成优化	激发创作灵感

8.3.3 生态价值

为银河麒麟应用生态添加 AI 类应用

提供可复用的 AI 集成框架

推动国产 AI 模型的实际应用落地

8.4 项目心得

技术收获：

通过深度集成麒麟 AI SDK，我们不仅掌握了国产 AI 框架的应用方法，更深入理解了大型语言模型的工作原理。

工程实践：

项目初期低估了异步消息处理的复杂性，导致 UI 卡顿问题频发。通过重构为三级消息处理机制（SDK 回调→消息队列→UI 更新），最终实现了流畅的用户体验。"

团队协作感悟：

采用特性分支+每日代码审查的工作模式，使团队在高效协作的同时保证了代码质量。

国产化感悟：

银河麒麟操作系统搭配麒麟 AI SDK 展现出令人惊喜的性能表现。在本地部署 Qwen-2.5-3b 模型的情况下，响应速度超过预期，验证了国产技术栈完全能满足生产级应用需求。

8.5 致谢

本项目获得麒麟软件有限公司的技术支持，感谢所有项目成员，特别感谢各位老师在实训学习中的全程指导，感谢麒麟公司提供的银河麒麟开发环境，使团队能在真实国产化平台上完成开发测试。