

暨南大学本科实验报告专用纸

课程名称 人工智能与多学科实践创新 成绩评定
实验项目名称 职场助手小程序的开发 指导教师 赵阔
实验项目编号 01 实验项目类型 实验地点
学生姓名 赵俊文 学号 2022104002
学院 智能科学与工程学院 系 专业 人工智能
实验时间 2025 年 4 月 02 日 午 ~ 4 月 23 日 午

一、项目概述

在数字化办公场景下，职场工作者常面临信息检索效率低、工作流程复杂等问题。为解决这些痛点，本团队开发了基于微信生态的“职智通”智能辅助系统。该系统整合了自然语言处理技术与企业知识库，通过智能对话接口为用户提供即时工作支持，并配备完整的交互历史追溯功能。

二、系统功能架构

1. 用户身份认证模块

利用微信官方登录接口获取用户唯一标识（OpenID），确保不同用户数据独立存储。采用 HTTPS 协议保障数据传输安全。

2. 智能问答交互界面

前端界面包含文本输入区域与操作按钮，支持自由输入与预设模板两种提问方式。用户提交问题后，系统自动将内容传输至后端服务，经过格式校验后转发至智能模型处理，最终将响应结果返回至前端界面分层展示。

3. 数据存储与管理模块

采用关系型数据库存储每轮对话的详细记录，包括用户提问内容、系统响应文本、对话发生时间等核心信息。通过建立会话标识字段，实现多轮对话的关联存储，为历史查询提供数据基础。

4. 历史记录检索功能

开发时间轴浏览界面，支持按自然日期筛选历史对话记录。查询结果以列表形式展示，包含提问内容与对应解答的概要信息，用户可通过点击查看完整对话详情。

三、技术架构

1. 前端实现方案

采用微信小程序原生开发框架构建用户界面，使用官方组件库实现标准化的交互元素。通过页面路由管理实现不同功能模块间的跳转，利用数据绑定机制实现界面元素的动态更新。

2. 后端技术架构

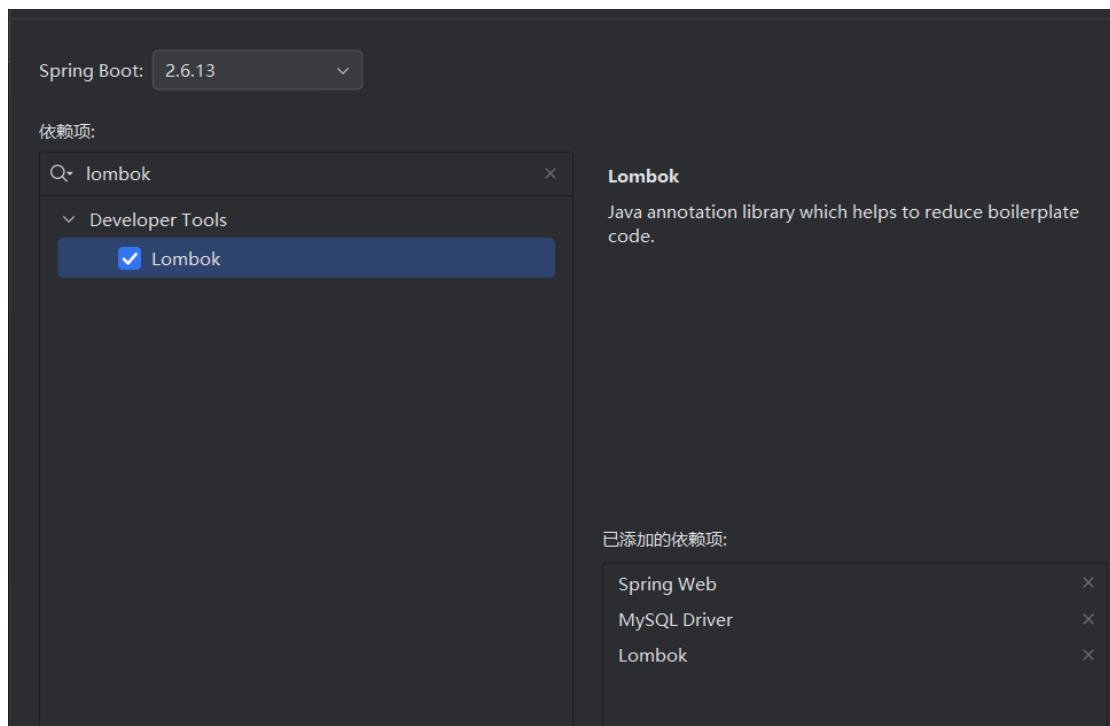
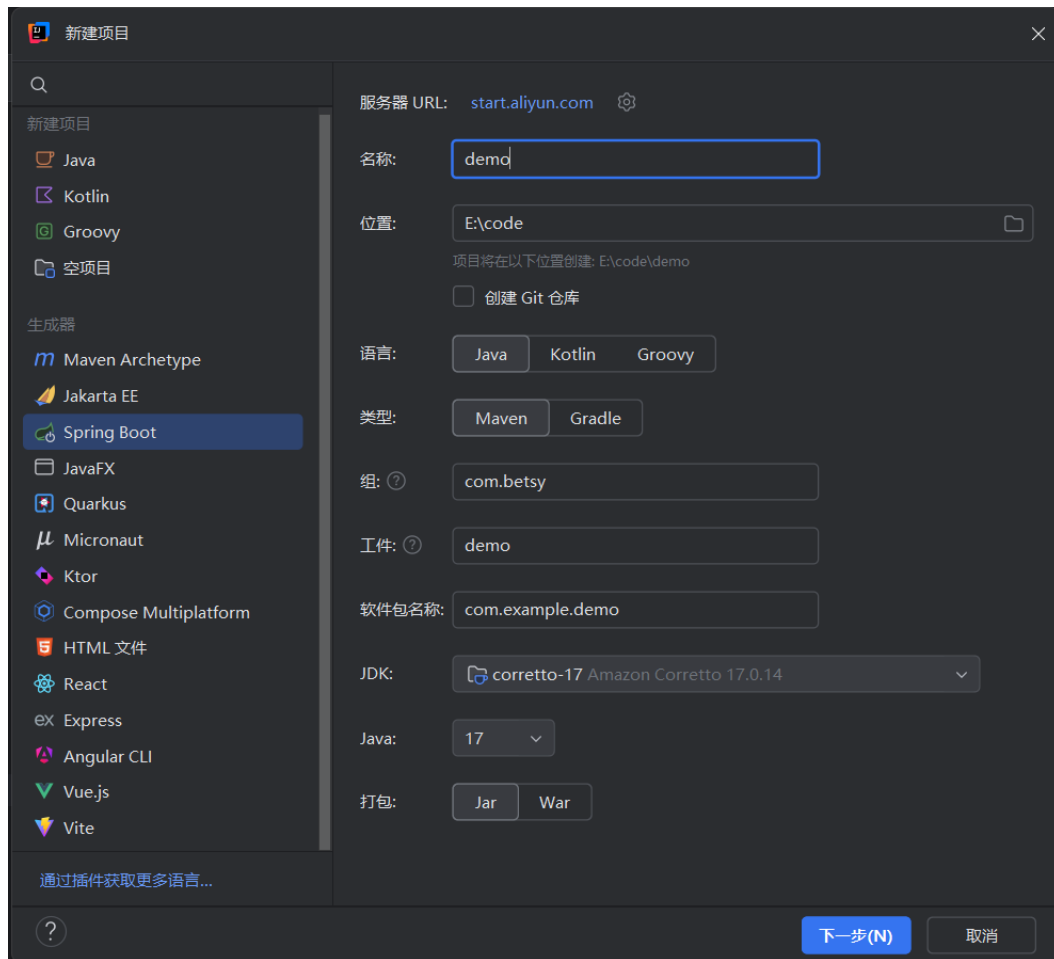
- 服务框架选用 Spring Boot 体系, 基于约定优于配置原则快速搭建服务架构。
通过注解方式定义请求处理接口，简化开发流程
- 数据存储层采用 MySQL 数据库管理系统，通过对象关系映射技术实现数据实体与数据库表的自动转换。配置连接池优化数据库访问性能。
- 辅助开发工具整合代码简化库，自动生成数据实体类的基础方法。集成 JSON 数据处理组件，规范前后端数据交换格式。
- ORM 工具：使用 Spring Data JPA 简化数据库操作，支持快速切换至 MyBatis

四、实验步骤

1. 后端实现

(1) 项目初始化

创建一个 springBoot 工程项目，并引入 Spring Web 框架和数据库依赖已经 lombok 依赖简化实体类的编码



(2) 集成大模型 SDK

添加质谱清言 SDK 依赖，配置 API 密钥与请求参数

```
<!--https://github.com/zhipuai/zhipuai-sdk-java-v4-->
<dependency>
    <groupId>cn.bigmodel.openapi</groupId>
    <artifactId>oapi-java-sdk</artifactId>
    <version>release-V4-2.4.1</version>
</dependency>
```

(3) 核心逻辑封装

在 AIManager 类中实现通用请求方法,通过@Service 注解注入 Spring 容器,便于全局调用

```
15 @Component
16 public class AIManager {
17     @Resource
18     private ClientV4 clientV4;
19
20     public String doRequest(List<ChatMessage> messages, Boolean stream, Float temperature) {
21         // 构造请求
22         ChatCompletionRequest chatCompletionRequest = ChatCompletionRequest.builder()
23             .model(Constants.ModelChatGLM4)
24             .stream(stream)
25             .invokeMethod(Constants.invokeMethod)
26             .temperature(temperature)
27             .messages(messages)
28             .build();
29         ModelApiResponse invokeModelApiResp = clientV4.invokeModelApi(chatCompletionRequest);
30         ChatMessage result = invokeModelApiResp.getData().getChoices().get(0).getMessage();
31         return result.getContent().toString();
32     }
33
34 }
```

(4) 实体类设计

- a) RequestMessage: 包含 sessionId (用户标识) 和 message (问题文本), 用于区分用户会话

```

1 package com.betsy.largemodelinvocationtest.domain;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class RequestMessage {
11     private Long sessionId;
12     private String message;
13 }
14

```

b) Chat: 映射数据库表, 存储问题、回答、时间戳及逻辑删除标记 (@TableLogic)

```



12 @Data
13 @AllArgsConstructor
14 @NoArgsConstructor
15 public class Chat {
16     @TableId(type = IdType.AUTO)
17     private Long id; // id
18
19     private Long sessionId; // 用户id
20
21     private String question; // 用户的问题
22
23     private String answer; // ai的回答
24
25     private Date createTime; // 创建时间
26
27     private Date updateTime; // 更新时间
28
29     @TableLogic
30     private Integer isDelete; // 是否删除, 逻辑删除
31 }

```

(5) 数据库配置

在 application.yml 中设置 MySQL 连接信息, 包括 URL、用户名及密码。

```

7      spring:
8            datasource:
9          driver-class-name: com.mysql.cj.jdbc.Driver
10         url: jdbc:mysql://localhost:3306/ai
11         username: root
12          password: 1234

```

(6) 服务层开发

Service 层处理前端请求，调用大模型 API 并返回结果，同时将对话记录写入数据库

```

1  # 数据库初始化
2  -- 创建库
3  create database if not exists ai;
4
5  -- 切换库
6  use ai;
7
8  -- 用户表
9  create table if not exists chat
10 (
11     id          bigint auto_increment comment 'id' primary key,
12     sessionId   varchar(256)              null comment '用户id',
13     question    varchar(1024)             not null comment '用户问题',
14     answer      varchar(1024)             not null comment 'ai回答',
15     mpOpenId    varchar(256)              null comment '公众号openId',
16     createTime  datetime default CURRENT_TIMESTAMP not null comment '创建时间',
17     updateTime  datetime default CURRENT_TIMESTAMP not null on update CURRENT_TIMESTAMP comment '更新时间',
18     isDelete    tinyint default 0          not null comment '是否删除'
19 ) comment '用户';

```

```

@Override
public String sendMessage(RequestMessage requestMessage) {
    // 调用API
    String userMessage = requestMessage.getMessage();
    String systemMessage = "你是一个只能问答助手，我会向你提问，你的回答要尽量简洁凝练";
    String result = aiManager.doSyncStableRequest(systemMessage, userMessage);

    // 将结果写入数据库
    Chat chat = new Chat();
    chat.setSessionId(requestMessage.getSessionId());
    chat.setQuestion(userMessage);
    chat.setAnswer(result);
    chat.setCreateTime(new Date());
    chat.setUpdateTime(new Date());

    int insert = chatMapper.insert(chat);
    if (insert == 0) {
        throw new RuntimeException("插入失败");
    }
    return result;
}

```

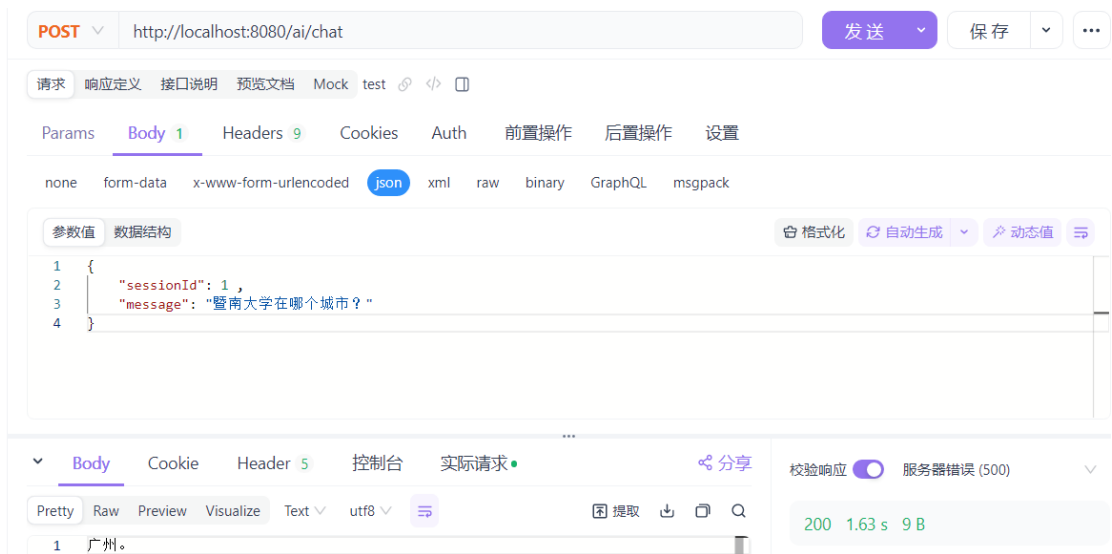
```

11  @RestController
12  public class AIController {
13      @Resource
14      private ChatService chatService;
15      @PostMapping("/ai/chat")
16      public String sendMessage(@RequestBody RequestMessage requestMessage) {
17
18          return chatService.sendMessage(requestMessage);
19      }
20  }

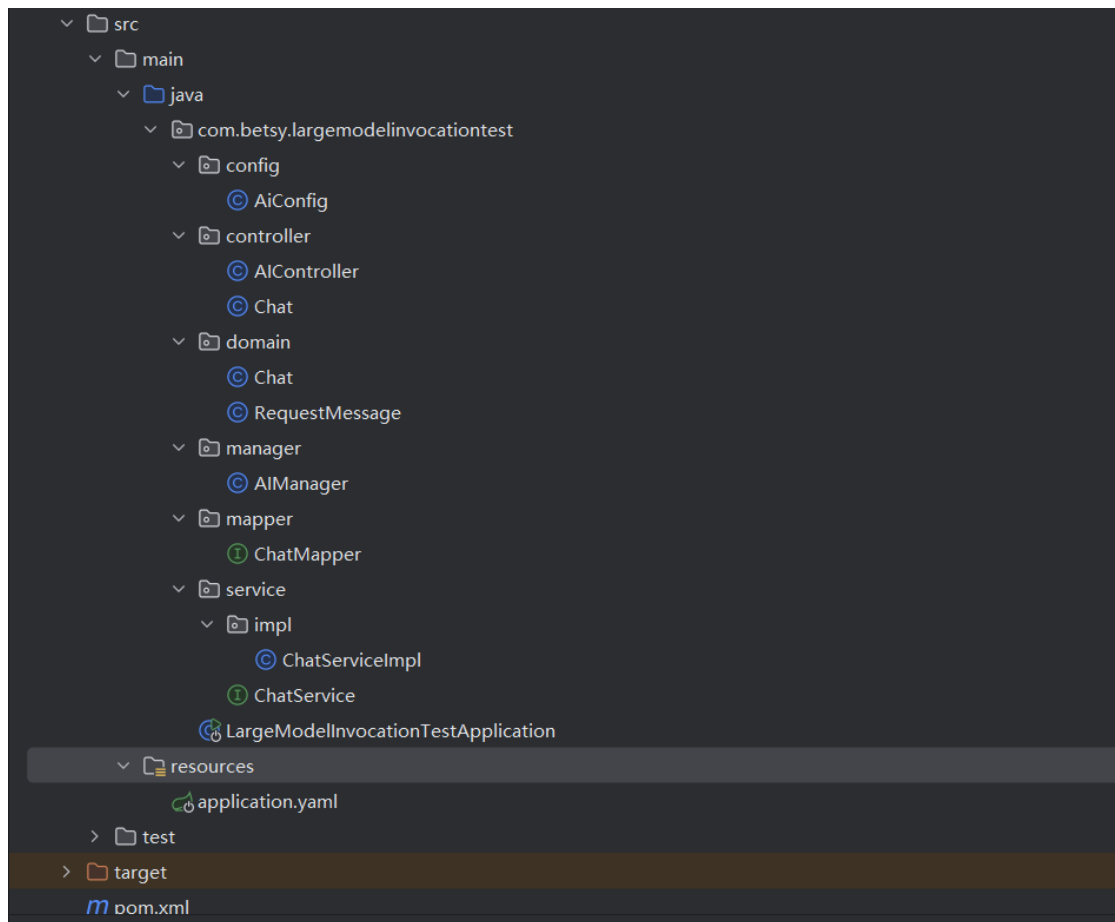
```

(7) 接口测试

使用 ApiFox 测试 /ai/chat 接口，验证问答功能及数据存储是否正常



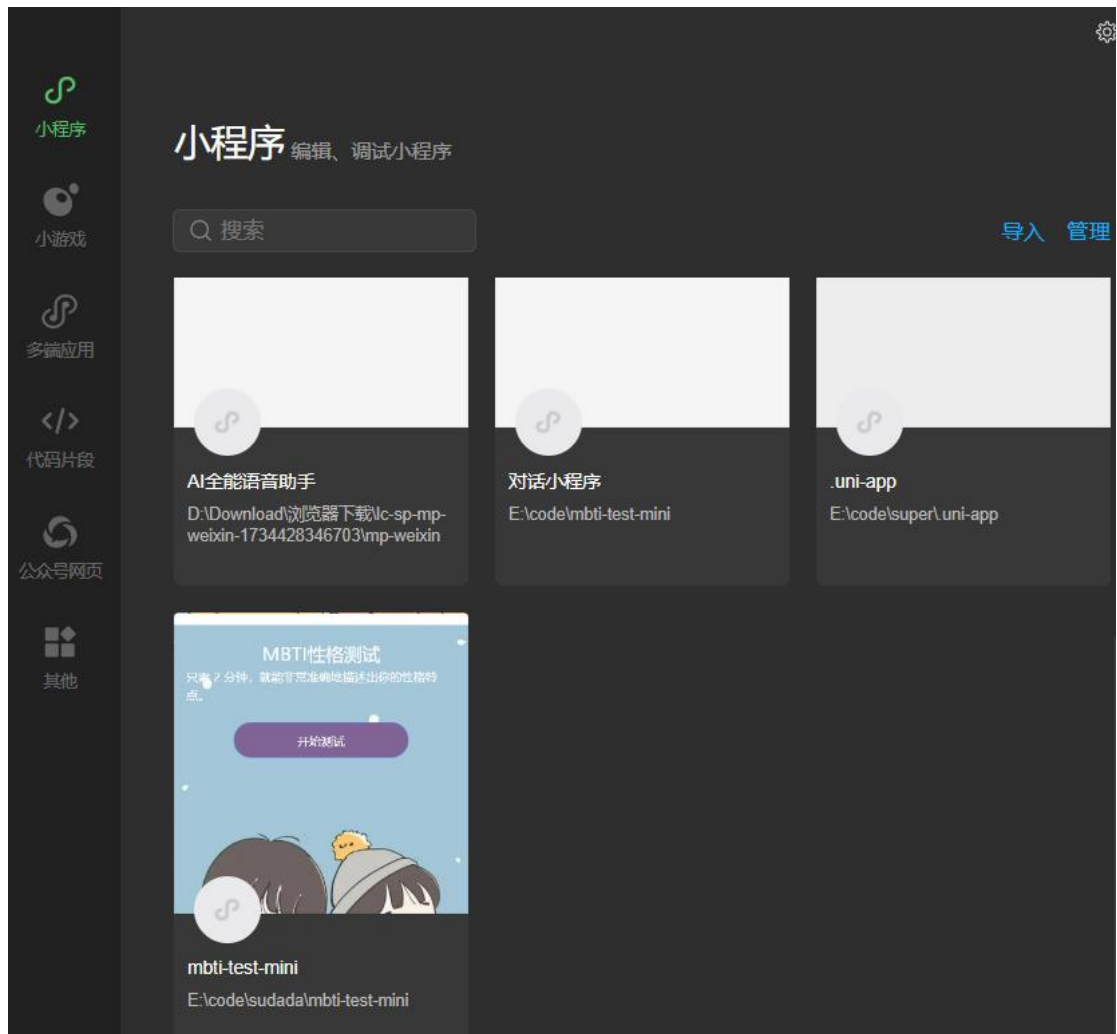
(8) 最终后端架构



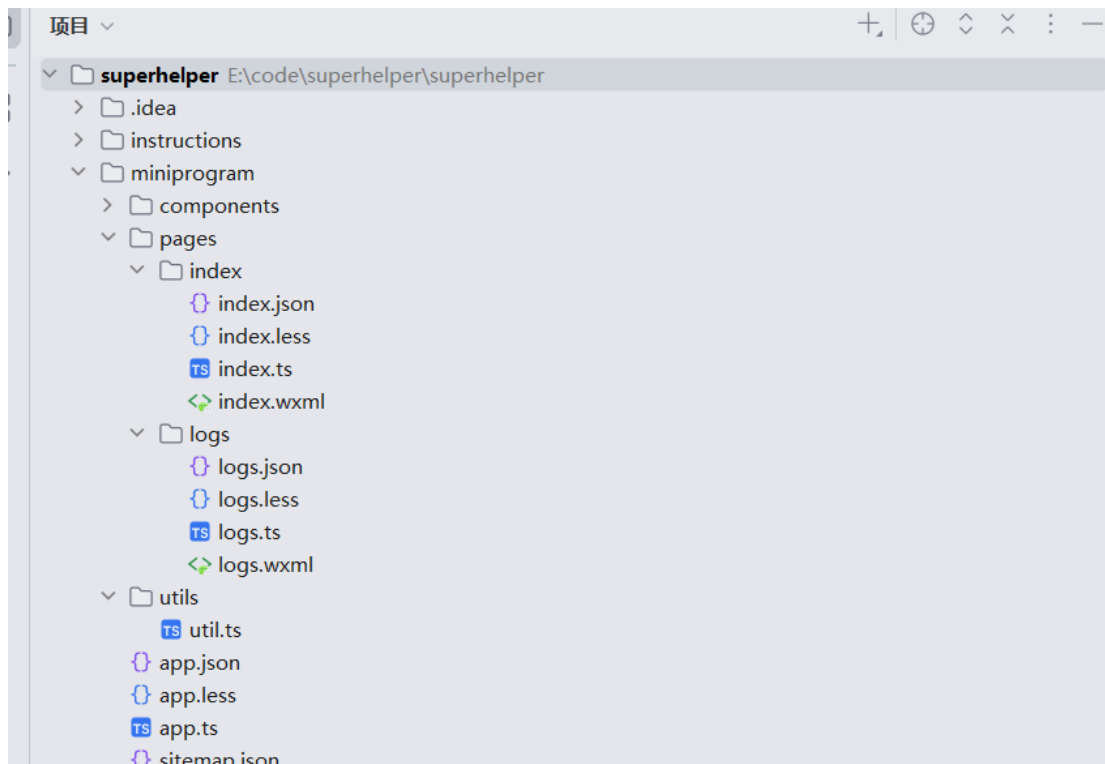
2. 前端开发

(1) 环境搭建

安装微信开发者工具，配置项目目录及 AppID。



(2) 项目架构

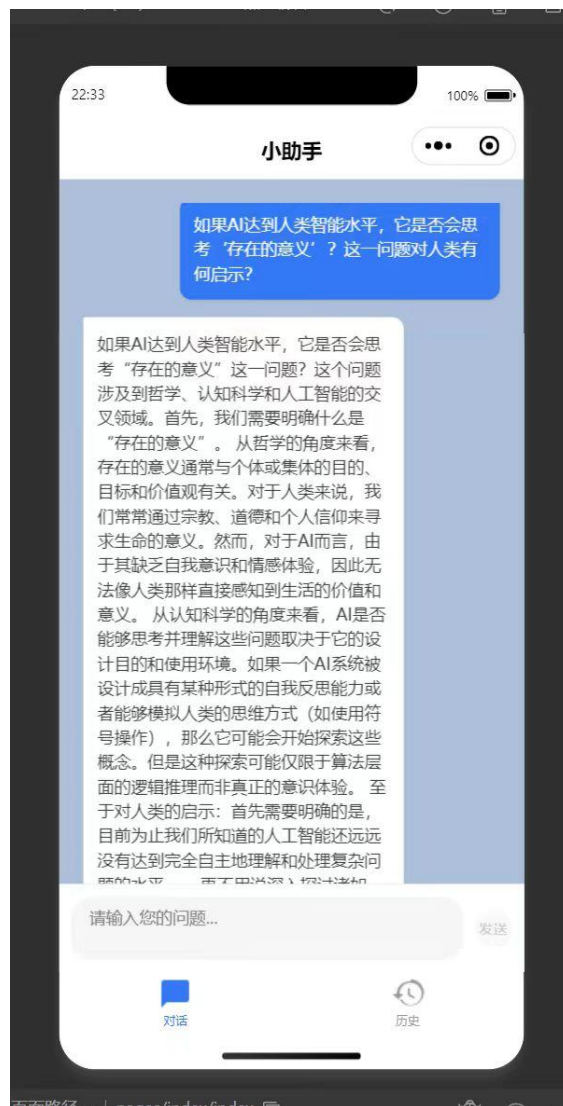


(3) 界面设计

采用经典的上下分栏布局，顶部是消息展示区，底部是输入功能区。消息展示区使用 scroll-view 组件实现可滚动视图，每条消息以气泡形式呈现，用户消息右对齐显示，AI 回复左对齐显示，并附带精确到分钟的时间戳。输入功能区包含一个支持多行输入和高度自适应（1-4 行）的 textarea 组件，以及右侧的发送按钮，该按钮会根据输入内容动态切换可用状态。当用户提交问题时，系统会立即将问题加入本地消息列表以提升响应体验，同时通过 `wx.request()` 方法将问题文本和用户标识以 POST 方式发送至后端服务器。在等待响应期间，界面会显示加载状态，成功接收 AI 回复后自动更新消息列表并滚动至最新内容，若请求失败则会提示用户重试并保留未发送的内容。整个交互流程注重即时反馈和容错处理，确保用户获得流畅自然的对话体验

```
131     async callSparkAPI(message: string): Promise<SparkAPIResponse> {
132         return new Promise((resolve : (value: SparkAPIResponse | PromiseLike<Sp... , reject : (reason?: any) => void ) : void =
133             wx.request({
134                 url: 'https://spark-api-open.xf-yun.com/v1/chat/completions',
135                 method: 'POST',
136                 data: {
137                     model: 'lite',
138                     messages: [
139                         {
140                             role: 'system',
141                             content: '你是知识渊博的助理'
142                         },
143                         {
144                             role: 'user',
145                             content: message
146                         }
147                     ],
148                     temperature: 0.5,
149                     top_k: 4,
150                     stream: false,
151                     max_tokens: 1024,
152                     presence_penalty: 1,
153                     frequency_penalty: 1
154                 },
```

五、结果展示



六、实验总结

本项目的开发实践验证了智能问答系统在职场场景中的应用价值。通过整合 Spring Boot 后端框架与微信小程序前端技术，成功构建了包含用户认证、智能交互、数据存储等核心功能的完整系统。在技术实现层面，我掌握了前后端数据对接、大模型 API 集成、数据库事务管理等关键能力，特别是在解决微信登录状态同步、长文本存储优化等实际问题中积累了宝贵经验。项目采用模块化开发模式，通过清晰的层次划分提升了代码可维护性，为后续功能扩展奠定了基础。

从工程实践角度，本次实验深化了我对企业级应用开发流程的理解，包括需求分析、技术选型、系统调试等环节的衔接配合。尽管在初期遇到接口兼容性和性能瓶颈等问题，但通过查阅文档、调试工具的使用，最终

形成了稳定的解决方案。未来计划引入对话上下文管理模块，优化历史检索效率，进一步提升系统的实用性和智能化水平。