基于 SpringBoot+Vue 的社区居民健康信息管理系统的设计与实现*

刘业奔 林宁 左悦**

南宁学院信息工程学院，南宁，530200 南宁学院土木与建筑工程学院，南宁，530200

摘 要 现随着人工智能技术的不断深入应用，开发智能健康风险评估与个性化健康指导系统对个人健康管理和疾病预防至关重要。本项目基于 Spring Boot 构建后端核心，设计实现了一个社区居民健康信息管理系统。该系统使用 Vue.js 开发前端界面，采用 MyBatis-Plus 作为数据访问层，以 MySQL 实现数据持久化存储。系统包含身体信息管理、健康分析、健康计划、健康知识、健康打卡、健康圈、AI 问诊以及用户管理、健康知识管理、健康圈管理等模块，用户可上传身体信息获取智能健康分析与建议，制定并打卡健康计划，在健康圈分享日常及使用 AI 问诊，管理员可在线管理用户、更新健康知识和管理健康圈。系统功能、性能等多维度测试结果表明，系统核心功能运行正常，可广泛适用于普通民众日常健康管理、社区健康服务推广、企业员工健康干预等场景。

关键字 健康管理；健康计划；Spring Boot

Design and Implementation of Community Residents' Health Information Management System Based on SpringBoot and Vue*

LiuYeBen LinNing ZuoYue**

College of Information Engineering
Nanning University
Nanning 530200 China

College of Architecture and Civil Engineering
Nanning University
Nanning 530200 China
zuoyue@unn.edu.cn

Abstract—With the continuous and in-depth application of artificial intelligence technology, developing an intelligent health risk assessment and personalized health guidance system is crucial for personal health management and disease prevention. This project builds the backend core based on Spring Boot and designs and implements a community residents' health information management system. The system uses Vue.js to develop the front-end interface, adopts MyBatis-Plus as the data access layer, and uses MySQL for data persistence storage. The system includes modules such as physical information management, health analysis, health plan, health knowledge, health check-in, health circle, AI consultation, user management, health knowledge management, and health circle management. Users can upload their physical information to obtain intelligent health analysis and suggestions, formulate and check-in health plans, share daily life in the health circle, and use AI consultation. Administrators can manage users online, update health knowledge, and manage the health circle. Multi-dimensional testing results of system functionality and performance show that the core functions of the system operate normally and can be widely applied to scenarios such as daily health management for ordinary people, promotion of community health services, and health intervention for enterprise employees.

Keywords—Health management；Health plan；Spring Boot

1 引 言

在当前全球发展进程中，人类健康的重要性被提升至前所未有的高度，已成为衡量经济体与社会发展水平、评估民众幸福指数的核心标志之一[1]。但是传统的健康管理模式却依然严重依赖于出现症状再寻求治疗的被动反应机制，这种滞后性的服务模式往往只能在病情出现明显波动或产生并发症时才进行干预，错失了最佳的健康管理时机。而健康管理借助互联网技术的便捷性与智能化优势，能够为用户提供健康指标记录、动态监测、风险预警及个性化指导等一站式服务，恰好契合慢性病预防的核心需求，成为防控慢性病的高效手段[4]。

因此，开发一个能够整合多源健康数据、提供智能风险评估和个性化健康指导的居民健康管理系统，不仅是响应国家健康战略的具体实践，也是满足人民

*基金资助：南宁学院2026年横向课题（2026HX008）资助。
**通讯作者：左悦，副教授，zuoyue@unn.edu.cn

群众日益增长的健康需求的必然选择，具有重要的时代意义和实用价值。

2 相关技术研究及介绍

2.1 Spring MVC 框架介绍

Spring MVC是根据MVC设计模式Spring框架搭建的，是构建Web表现层的经典框架[3] 。它严格遵循MVC架构模式，负责处理HTTP请求并生成响应。框架核心DispatcherServlet作为请求分发枢纽，协调处理器映射、控制器、视图解析器等组件。请求经其路由至对应控制器方法，控制器调用业务服务并封装数据与视图信息，最后由视图解析器渲染视图，实现了Web层、控制层与业务逻辑层的分离，使接口设计更灵活规范。

2.2 Java 语言介绍

Java 作为一种面向对象的高级编程语言，凭借其一次编写，到处运行的跨平台特性而备受青睐，提供统一多线程模型与并发工具库，同时其摒弃指针并采用沙箱安全模型的特性[4]。此外，Java 拥有强大的内存管理机制，通过垃圾自动回收功能有效防止了内存泄漏，提升了程序的健壮性与稳定性。其丰富的类库与成熟的生态体系，为快速构建跨平台的企业级应用提供了全面支持。

2.3 Mysql 数据库介绍

MySQL作为一款广受欢迎的开源关系型数据库管理系统，其具有较高的性能、良好的兼容性和可移植性、较高的安全性和稳定性等优点[5]。它能够快速处理大量数据，并支持标准的结构化查询语言，便于进行复杂的数据操作与管理。MySQL提供了完善的事务支持，确保了数据的一致性和完整性。同时，它具备强大的数据安全机制和灵活的存储引擎架构，能够满足从个人网站到大型企业级系统等不同规模的应用需求。

2.4 通义千问大语言模型介绍

近年来，人工智能领域的最新进展，尤其是生成式人工智能与大型语言模型的突破，推动AI技术逐步融入数字健康应用[6]。通义千问是阿里云自主研发的大语言模型系列，覆盖自然语言、视觉、音频、视频等多模态能力，凭借万亿级数据训练与领先算法框架，在文本处理、多模态交互、行业适配等场景中提供高效智能化服务，是国内技术成熟度高、生态覆盖广的大模型体系之一。

3 需求分析

3.1 系统功能需求

本系统的主要目的是管理居民身体健康，主要功能有用户端功能包括上传身体信息，导出健康数据，制定健康计划，健康打卡，AI问诊，分享健康动态，下载知识视频，管理个人信息。管理端功能包括管理用户，管理健康圈，管理运动知识，管理饮食知识，管理疾病知识。

3.2 系统性能需求

加载性能与响应速度。页面首次加载时间控制在3秒以内，二次加载控制在1秒以内，避免用户因等待时间过长影响界面查看与使用体验。用户操作的反馈延迟控制在300毫秒以内，核心操作如数据提交、状态切换保证即时响应。图表及数据分析页的可视化内容，在数据量不超过3个月历史记录的情况下，渲染完成时间控制在500毫秒以内，且支持流畅的缩放与切换操作。用户进行数据查询时，查询响应时间控制在500毫秒以内，确保快速获取所需信息。

稳定性与并发支持。支持用户连续执行核心业务流程，全程无页面崩溃或数据丢失问题，同时保证数据提交过程的原子性。当用户同时触发多个数据请求系统需支持至少3个并发请求的有序处理，避免因接口阻塞导致页面冻结、操作无响应的情况。

易用性和可维护性。界面设计应直观清晰，用户无需培训或仅需少量引导即可完成核心操作，常用功能的操作步骤简单明了，整个系统的操作逻辑、术语和界面风格保持一致，降低了用户的认知负担。系统采用高内聚、低耦合的架构设计，使得健康分析、知识管理等模块能够独立进行更新、升级和故障排查。

安全性。健康管理系统需具备完善的安全性保障能力，仅允许通过身份验证的合法用户登录并使用系统功能，同时为不同角色用户配置差异化权限，清晰界定各用户的操作边界与数据访问范围。系统还需重点保护后台存储的用户核心数据安全，例如用户的登录账号、加密后的密码信息，以及身体指标、疾病记录等隐私数据，通过多重防护措施确保用户账号不被非法盗用、核心信息不被泄露或篡改，为用户数据安全提供全周期保障。

4 系统概要设计

4.1 系统总体 E-R 图

系统总体E-R图如图1所示，核心围绕用户与管理员两大角色展开：用户可创建打卡类型与打卡记录，运动计划，作息计划，上传身体信息，并能发布健康圈动态、对他人动态发表评论和点赞；管理员则负责管理健康圈动态与健康知识等平台公共内容，整体实现了个人健康数据管理与健康社交互动的业务闭环。

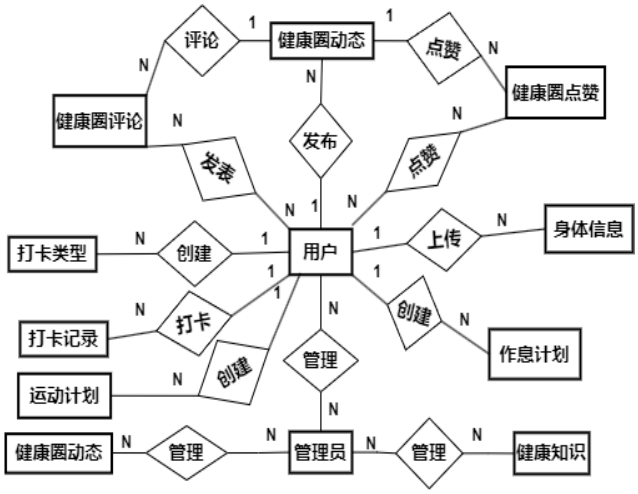

图 1 系统总体 E-R 图

4.2 数据库表设计

（1）用户表：

用户表的说明如表 1 所示。

表 1 用户表

字段名称	类型	长度	空值	说明
id	int	10	否	用户 ID
username	varchar	50	否	用户名
password	varchar	255	否	用户密码
email	varchar	100	否	邮箱
identity	tinyint	255	否	身份标识
created_at	datetime		否	创建时间
updated_at	datetime		否	更新时间
last_login_time	datetime		是	最后登录时间
avatar	varchar	255	是	头像 URL

（2）打卡记录表：

打卡记录表的说明如表 2 所示。

表 2 打卡记录表

字段名称	类型	长度	空值	说明
id	int	10	否	打卡记录 ID
type_id	int	10	否	打卡类型 ID
user_id	int	10	否	用户 ID
check_in_date	date		否	打卡日期
check_in_time	time		否	打卡具体时间
check_in_content	varchar	255	是	打卡内容

表 2 打卡记录表（续）

字段名称	类型	长度	空值	说明
health_snapshot	json		是	健康指标快照

（3）运动计划表：

运动计划表的说明如表 3 所示。

表 3 运动计划表

字段名称	类型	长度	空值	说明
id	int	10	否	计划 ID
user_id	int	10	否	用户 ID
plan_name	varchar	100	否	计划名称
cycle_days	int		否	计划周期
exercise_method	text		否	运动方式
exercise_goal	varchar	200	否	运动目标
start_time	datetime		否	开始时间
end_time	datetime		否	结束时间
create_time	timestamp		是	创建时间
update_time	timestamp		是	更新时间

（4）饮食知识表：

饮食知识表的说明如表 4 所示。

表 4 饮食知识表

字段名称	类型	长度	空值	说明
id	int	10	否	食物 ID
name	varchar	100	否	食物名称
category	varchar	50	否	食物分类
nutrition	text		是	营养成分
edible_value	text		是	食用价值
image_url	varchar	255	是	图片 URL
taboo	text		是	饮食禁忌
calories	decimal	(8,2)	是	食物热量
create_time	datetime		是	创建时间
update_time	datetime		是	更新时间

（5）健康圈动态表：

健康圈动态表的说明如表 5 所示。

表 5　健康圈动态表

字段名称	类型	长度	空值	说明
id	int	10	否	动态 ID
user_id	int	10	否	用户 ID
content	text		是	动态内容
type	tinyint		否	内容类型
related_id	int		是	关联 ID
image_urls	varchar	255	是	图片 URL
like_count	int		是	点赞数
comment_count	int		是	评论数
create_time	datetime		是	创建时间
update_time	datetime		是	更新时间

5　系统核心功能详细设计与实现

5.1 登陆模块实现

登录模块主要实现用户身份认证功能，后端通过login方法接收登录数据，并生成JWT令牌实现状态认证。通过UserService查询数据库验证用户身份，如果验证成功则生成用户令牌并返回用户信息，如果失败则返回相应的错误信息。整个模块实现了从用户输入、前端验证、数据提交、后端处理到权限控制的完整流程。

核心代码：

```
    @PostMapping("/login")
public ResultVO login(@Valid @RequestBody LoginDTO loginDTO) {
    try {
    UserloginUser=userService.login(loginDTO.getUsername(), loginDTO.getPassword());
        if (loginUser != null) {
            loginUser.setLastLoginTime(LocalDateTime.now());
            userService.updateUser(loginUser);
            String token = jwtUtils.generateToken(
                    loginUser.getId(),
                    loginUser.getUsername(),
                    loginUser.getIdentity()
            );
            // 返回用户信息和令牌
            Map<String, Object> data = new HashMap<>();
            data.put("token", token);
            data.put("user", loginUser);
            return ResultVO.success(data);
        } else {
            return ResultVO.fail("用户名或密码错误");
        }
    } catch (Exception e) {
        return ResultVO.fail("登录失败: " + e.getMessage());
    }
}
```

5.2 注册模块实现

注册模块主要实现用户账号创建功能，后端调用checkUsernameExists/checkEmailExists接口校验注册数据的唯一性，校验失败则即时显示用户名已存在并阻断提交，校验通过则将加密后的密码及完整注册信息提交至后端完成账号创建。后端采用MD5加密技术保护密码安全，并支持头像文件上传处理，最后通过UserMapper将用户数据插入数据库。整个模块实现了从用户信息输入、异步验证、数据提交、后端处理的完整流程。

核心代码：

```
    @PostMapping("/register")
public ResultVO register(@RequestBody User user) {
    try {
        // 检查用户名是否存在
        if (userService.checkUsernameExists(user.getUsername())) {
            return ResultVO.fail("用户名已存在");
        }
        // 检查邮箱是否已存在
        if (userService.checkEmailExists(user.getEmail())) {
            return ResultVO.fail("邮箱已被注册");
        }
        // 对密码进行MD5加密
        String encryptedPassword = Md5Util.md5(user.getPassword());
        user.setPassword(encryptedPassword);
        // 设置创建时间
        user.setCreatedAt(LocalDateTime.now());
        user.setUpdatedAt(LocalDateTime.now())
        // 默认普通用户身份
        user.setIdentity(0);
        // 设置默认头像（如果前端没有传头像URL）
        if (user.getAvatar() == null || user.getAvatar().isEmpty()) {
            user.setAvatar("/default-avatar.png");
        }
        boolean success = userService.register(user);
        return success ? ResultVO.success("注册成功") : ResultVO.fail("注册失败");
    } catch (org.springframework.dao.DuplicateKeyException e) {
        // 捕获数据库唯一约束异常
        if (e.getMessage().contains("uk_email")) {
            return ResultVO.fail("邮箱已被注册");
        } else if (e.getMessage().contains("uk_username")) {
            return ResultVO.fail("用户名已存在");
        }
        return ResultVO.fail("注册失败：数据已存在");
    } catch (Exception e) {
        return ResultVO.fail("注册失败：" + e.getMessage());
    }
}
```

5.3 作息计划制定模块实现

健康计划模块主要实现用户健康作息计划的创建、管理、查询和分享功能，后端通过RoutinePlan实体类接收数据，RoutinePlanService调用MyBatis Plus的save方法将数据持久化到数据库，同时通过getRoutinePlans方法支持计划查询和分页展示，updateRoutinePlan方法实现健康计划的更新，deleteRoutinePlan方法实现健康计划的删除，整个模块实现了从计划创建、验证、提交到管理的完整作息规划流程。

核心代码：

```
    @PostMapping //新增作息计划
    public ResultVO addRoutinePlan(@RequestBody RoutinePlan plan, HttpServletRequest request) {
        String token = jwtUtils.getTokenFromRequest(request);
        Long userId = jwtUtils.getUserIdFromToken(token).longValue();
        plan.setUserId(userId);
        // 再次校验日期唯一性（防止并发冲突）
        if (routinePlanService.existsByDateAndUserId(String.valueOf(plan.getScheduleDate()), userId)) {
            return ResultVO.fail("同一天只能添加一个作息计划");
        }
        plan.setCreateTime(LocalDateTime.now());
        plan.setUpdateTime(LocalDateTime.now());
        boolean success = routinePlanService.save(plan);
        return success ? ResultVO.success("新增计划成功") : ResultVO.fail("新增计划失败");
    }
//更新作息计划

        @PutMapping("/{id}")
    public ResultVO updateRoutinePlan(
            @PathVariable Long id,
            @RequestBody RoutinePlan plan,
            HttpServletRequest request) {
        String token = jwtUtils.getTokenFromRequest(request);
        Long userId = jwtUtils.getUserIdFromToken(token).longValue();
        RoutinePlan existingPlan = routinePlanService.getById(id);
        if (existingPlan == null) {
            return ResultVO.fail("计划不存在");
        if (!existingPlan.getUserId().equals(userId)) {
            return ResultVO.fail("无权修改该计划");
        }
        plan.setId(id);
        plan.setUserId(userId);
        plan.setUpdateTime(LocalDateTime.now());
        boolean success = routinePlanService.updateById(plan);
        return success ? ResultVO.success("更新计划成功") : ResultVO.fail("更新计划失败");
    }
//删除作息计划
    @DeleteMapping("/{id}")
    public ResultVO deleteRoutinePlan(@PathVariable Long id, HttpServletRequest request) {
        String token = jwtUtils.getTokenFromRequest(request);
        Long userId = jwtUtils.getUserIdFromToken(token).longValue();
        RoutinePlan plan = routinePlanService.getById(id);
        if (plan == null) {
            return ResultVO.fail("计划不存在");
        }
        if (!plan.getUserId().equals(userId)) {
            return ResultVO.fail("无权删除该计划");
        }
        boolean success = routinePlanService.removeById(id);
        return success ? ResultVO.success("删除计划成功") : ResultVO.fail("删除计划失败");
    }
}
```

5.4 健康圈模块实现

健康圈模块是一个社交化的健康分享平台，主要功能有动态发布，点赞，评论，以及管理员可以对所有动态进行删除。后端调用addCircleWithImages方法处理图片上传和数据验证，使用insert方法将数据持久化到数据库；动态列表查询getCircleList方法实现分页展示和关联数据加载；publishCircle方法实现发布动态，deleteCircle方法实现删除动态，点赞功能通过handleLike方法，评论功能通过showCommentDialog和addComment接口实现，整个模块实现从动态发布、展示到社交互动的完整健康分享流程。

核心代码:

```
    @PostMapping("/publish")//发布动态-支持文件上传
public ResultVO publishCircle(
        @RequestParam("content") String content,
        @RequestParam(value = "images", required = false) MultipartFile[] images,
        HttpServletRequest request)
    Integer userId = getUserIdFromRequest(request);
    if (userId == null) {
        return ResultVO.fail("请先登录");
    }
    boolean success = healthCircleService.addCircleWithImages(userId, content, images);
    return success ? ResultVO.success("发布成功") : ResultVO.fail("发布失败");
}
// 点赞动态
@PostMapping("/like/{circleId}")
public ResultVO likeCircle(@PathVariable Integer circleId, HttpServletRequest request) {
    Integer userId = getUserIdFromRequest(request);
    if (userId == null) {
        return ResultVO.fail("请先登录");
    }
    boolean success = healthCircleService.likeCircle(circleId, userId);
    return success ? ResultVO.success("点赞成功") : ResultVO.fail("点赞失败");
}
//添加评论
@PostMapping("/comment")
public ResultVO addComment(@RequestBody HealthCircleComment comment, HttpServletRequest request) {
    Integer userId = getUserIdFromRequest(request);
```

```
    if (userId == null) {
        return ResultVO.fail("请先登录");
    }
    comment.setUserId(userId);
    boolean success = healthCircleService.addComment(comm
    ent);
    return success ? ResultVO.success("评论成功") : ResultV
O.fail("评论失败");
}
//删除动态
@DeleteMapping("/{circleId}")
public ResultVO deleteCircle(@PathVariable Integer circleId,
HttpServletRequest request) {
    Integer userId = getUserIdFromRequest(request);
    if (userId == null) {
        return ResultVO.fail("请先登录");
    }
    boolean success = healthCircleService.deleteCircle(circleId,
 userId);
    return success ? ResultVO.success("删除成功") : ResultV
O.fail("删除失败");
}
```

5.5 AI 问诊模块实现

AI问诊模块是一个基于人工智能的医疗咨询平台，主要功能通过用户给出的症状返回诊断建议。后端通过AiMedicalServiceImpl的getDiagnosisSuggestion方法构建专业医疗提示词，调用外部AI模型的API获取诊断建议，通过parseResponse方法解析AI响应结果，最终将格式化后的诊断建议返回给前端；前端通过formatResult方法处理显示格式，整个模块实现从症状描述输入到AI智能诊断建议输出的完整问诊流程。

核心代码：

```
public class AiMedicalServiceImpl implements AiMedical
Service {
    @Value("${ai.medical.api-key}")
    private String apiKey;
    @Value("${ai.medical.api-url}")
    private String apiUrl;
    private final OkHttpClient client = new OkHttpClient();
    private final ObjectMapper objectMapper = new ObjectM
    apper();
    @Override
    public String getDiagnosisSuggestion(String symptoms) {
        try {
            // 1. 构建符合API要求的请求体
            String requestBody = buildRequestContent(sympt
            oms);
            // 2. 创建HTTP请求
            Request request = new Request.Builder()
                    .url(apiUrl)
                    .addHeader("Content-Type", "application
                    /json")
                    .addHeader("Authorization", "Bearer "
            + apiKey)
post(RequestBody.create(requestBody, MediaType.parse("applica
tion/json")))
            .build();
            // 3. 发送请求并处理响应
            Response response = client.newCall(request).exec
            ute();
            if (response.isSuccessful() && response.body() !
            = null) {
                return parseResponse(response.body().string
            ());
            } else {
                return "获取诊断建议失败：" + (response.b
ody() != null ? response.body().string() : response.message());
            }
        } catch (Exception e) {
            e.printStackTrace();
            return "系统错误：" + e.getMessage();
        }
    }
//构建AI模型请求内容
    private String buildRequestContent(String symptoms) thro
ws IOException {
        String medicalPrompt = "你是一名专业的医疗咨询
助手，请根据用户描述的症状或问题，直接提供专业、全面且
有针对性的回答。\n" +
        "用户问题：" + symptoms;
        ObjectNode rootNode = objectMapper.createObjectNo
        de();
        rootNode.put("model", "qwen-turbo");
        rootNode.put("temperature", 0.7);
        rootNode.put("max_tokens", 1024);
        // 创建 messages 数组
        ArrayNode messagesArray = objectMapper.createArra
        yNode();
        ObjectNode messageNode = objectMapper.createObje
        ctNode();
        messageNode.put("role", "user");
        messageNode.put("content", medicalPrompt);
        messagesArray.add(messageNode);
        rootNode.set("messages", messagesArray);
        return objectMapper.writeValueAsString(rootNode);
    }
    private String parseResponse(String responseBody) throws
    IOException {
        JsonNode responseNode = objectMapper.readTree(resp
    onseBody);
        // 添加空值检查，避免 NullPointerException
        if (responseNode.has("choices") &&
                responseNode.get("choices").isArray() &&
                responseNode.get("choices").size() > 0) {
            JsonNode choice = responseNode.get("choices").
        get(0);
            if (choice.has("message") && choice.get("messag
e").has("content")) {
                return choice.get("message").get("content").
asText();
            }
        }
        return "未能获取有效的AI响应";
    }
}
```

6 系统测试

该系统测试主要包括功能测试、性能测试和安全测试三方面。功能测试验证用户端健康信息上传、健康计划制定、AI问诊、健康打卡及管理端用户、健康

知识、健康圈管理等模块的正常运行；性能测试评估页面加载、接口响应、数据渲染与并发处理能力，保障系统高效稳定；安全测试完善身份认证、权限管控等，防范信息泄露与非法访问。多维度测试全面保障系统规范、流畅、安全运行。

7 结束语

本系统基于Spring Boot、Vue.js为核心技术框架，搭配MyBatis-Plus与MySQL实现数据交互，集成AI大模型支撑的智能问诊系统，最终实现用户端身体信息管理、健康计划制定与打卡、健康圈社交互动，以及管理端用户管控、健康知识更新等功能，形成完整的健康管理闭环。

参 考 文 献

[1] 王洁琳.基于“健康中国 2030”战略的健康数据重用行为影响因素研究[D].哈尔滨:黑龙江大学,2023.

[2] 樊换换.个人健康管理系统的设计与实现[D].北京：北京邮电大学，2021.

[3] 黄建勋,王建钧,高卉,等.基于 SpringMVC 的农机具服务系统设计与实现[J].农业工程,2024,14(08):39-44.

[4] 孙子涵.基于 Java 的智慧羊场育种信息系统的设计与应用[D].新疆农业大学,2024.

[5] 刘沛.基于 MySQL 数据库的在线林业有害生物数字化展现路径研究[D].山西农业大学,2024.

[6] SCIBETTA L, PELLEGRINO M, MONGE ROFFARELLO A, et al. Intelligent support for digital wellbeing: A design framework through a systematic literature review[J]. International Journal of Human-Computer Studies,2025, 205: 103653.