# 智能HR系统需求分析报告

一、系统概述  
本系统致力于构建智能化、安全可靠的人力资源管理平台，通过技术创新实现人事管理全流程数字化。系统采用微服务架构，覆盖员工全生命周期管理，重点突破传统HR系统在实时性、智能化方面的短板。

## **二、**用户需求

**2.1 目标受众群体**

HR部门：需要提升事务性工作效率的专业人员

部门经理：需实时掌握团队人力数据的业务管理者

企业员工：期望便捷获取HR服务的终端用户

高层管理者：关注组织效能与人力资本的决策者

外部合作伙伴：猎头公司、社保代缴机构等第三方服务商

**2.2 用户共同需求特点**

全流程线上化操作需求

数据准确性要求（薪酬计算误差率<0.1%）

多终端无缝协同体验

敏感数据保密性保障

系统响应速度要求（关键操作响应时间<2秒）

三、功能需求分析  
**3.1 档案管理模块**

数据交互：支持Excel/CSV格式的千人级数据批量处理，导入容错率<0.1%

权限体系：实现字段级RBAC控制，薪酬等敏感字段采用动态水印技术

审计能力：记录字段级修改日志，支持版本回溯

**3.2 入职自动化模块**

系统集成：通过REST API对接AD域控系统，账号创建延迟≤3秒

流程引擎：可视化配置入职任务链，支持跨系统状态同步

通知机制：企业微信消息送达率≥99.9%，邮件递送延迟≤2分钟

**3.3 薪酬计算模块**

多账套管理：支持10+子公司独立核算，自动生成合并报表

数据安全：采用SM4算法加密PDF文件，邮件附件自动过期

合规审计：完整记录薪酬调整操作轨迹，支持穿透式查询

**3.4 任务分发模块**

任务分级体系：建立三维优先级矩阵（紧急程度/业务类型/处理难度）

可视化监控系统：融合甘特图的部门任务负载热力图

时效追踪预警：A级任务剩余时间不足预估处理时长1.5倍时预警

**四、非功能需求**  
**4.1 性能指标**

并发处理：支持3000人同时打卡

计算效率：百万级薪资数据批处理时间≤30分钟

**4.2 扩展能力**

插件框架：支持第三方考勤设备接入

服务治理：基于Kubernetes实现自动弹性伸缩

**五、创新功能实施路径**  
**5.1 AI简历匹配系统**

阶段演进：

规则引擎：构建50+岗位特征模板库

语义分析：基于BERT模型实现JD-CV相似度计算

多模态评估：整合语音面试视频分析

**5.2 情绪感知引擎**

迭代计划：  
V1.0：基础情感词典

**5.3 低代码平台**

组件库：预制20+HR业务组件（假勤审批等）

设计器：拖拽式界面，支持条件分支配置

运行时：基于BPMN 3.0标准的流程引擎

**六、技术规范**

6.1 技术栈

前端：React 18+Ant Design 5.x+WebSocket

后端：Python 3.10+Django 4.2+Celery

数据库：MySQL 8.0（主业务库）+InfluxDB 2.6（时序数据）

中间件：Redis 7.0（缓存）+Camunda 7.17（工作流）

安全：Keycloak 21（IAM）+Vault 1.13（密钥管理）

6.2接口标准

{

"REST API": "遵循OpenAPI 3.0规范，强制版本控制（/v1/...）",

"文件交互": "CSV采用RFC 4180标准，Excel模板限定.xlsx格式",

"消息协议": "WebSocket消息体必须包含seq\_num字段用于重试校验"

}

**七、数据库设计**

档案表--需要入职时间计算薪酬

员工id --k

姓名

身份证号 --唯一

年龄

当前入职时间

是否在职

--通过触发器或应用逻辑确保重新入职时自动更新该字段。

--数据一致性风险：重新入职时需手动更新入职时间，容易遗漏。

测试样例：

1. 档案表测试

测试用例：员工离职后重新入职，生成新员工ID。

验证点：

档案表中当前入职时间是否更新为新入职时间。

雇佣历史表中是否新增一条记录，且原记录的离职时间被填充。

2. 雇佣历史表测试

测试用例：插入一条离职时间为空的记录（模拟在职员工）。

预期结果：允许插入

目的：验证在职员工的雇佣记录合法性。

3. 登录表级联删除测试

测试用例：删除档案表中某员工，检查登录表对应记录是否自动删除。

验证点：

数据库外键是否设置ON DELETE CASCADE。

雇佣历史表

记录id --自增主键

身份证号 --外键（引用档案表.身份证号）

入职时间 --非空

离职时间

离职原因

账号表--密码与账号通过前端确保格式符号要求

员工id --k，f

账号 --唯一

密码（加密后）

测试样例：

插入相同账号的不同员工id，验证是否报错。

删除档案表员工时，验证登录表是否级联删除。

打卡记录表--与前端配合处理只打上班，隔天打卡，一整天都不打卡的情况，避免重复打卡

员工id --k,f

日期 --k

上班/下班 --k

时间

测试样例：

同一员工同一天插入两次记录（上班、下班），验证是否允许。

测试隔天打卡和全天未打卡的边界情况。

同一员工同一类型（如上班）多次打卡，验证是否被拒绝（依赖前端逻辑）。

23:59打卡次日00:01下班，验证日期字段是否正确处理。

薪酬表--薪酬通过脚本自动计算，但也应该可以手动更改

员工id --k,f

日期 --k

薪酬

测试样例：

脚本计算与手动修改冲突时，验证最终值优先级。

消息表

消息id --k，自增

员工id --f

时间

消息[json格式]

是否已读

类型

测试样例：

同一员工同一秒收到多条消息，验证是否报错。

高频并发消息插入（如同秒千条），验证性能及自增主键连续性。

部门信息表

部门id --k

部门名称

职位表

职位id --k

部门id --f

职位名称

员工-部门关联表 --若兼职，则通过在职位表中新增职位（如主管兼测试人员）实现

员工id --k,f

部门id --f

职位id --f

测试样例：

尝试插入无员工的部门或职位，验证是否允许。

查询空部门的职位列表，验证是否返回正确结果。

插入员工至多个部门，验证是否报错（根据业务需求调整表结构）。

审批表

审批id--自增主键

员工id --f

审批内容[json格式]

类型

测试样例：

同一员工提交多个审批请求，验证是否成功插入。

招聘表--每个人每天应当只能申请一个职位

姓名

日期 --k

身份证号 --k

学历

期望部门id --f，k

期望岗位id --f，k

测试样例

插入不存在的部门ID或职位ID，验证外键约束是否报错。

同一人同一天申请不同岗位，检查是否被允许（根据业务规则调整主键）。

任务表

任务id --k，自增

完成率--0-100

内容--json格式，前端低代码生成

分配人--默认为'CEO'

分配时间

开始时间

预期结束时间

实际结束时间

任务分配表

任务id--k,f

员工id --k,f

外键级联策略：

-- 删除任务时，同步删除分配记录

ALTER TABLE 任务分配表 ADD CONSTRAINT fk\_task

FOREIGN KEY (任务id) REFERENCES 任务表(任务id) ON DELETE CASCADE;

测试设计要点

唯一性约束：测试所有唯一键（如身份证号、账号）的冲突场景。

外键约束：验证删除档案表员工时，关联表（如登录表、打卡记录表）是否级联删除或阻止操作。

业务逻辑：

自动计算薪酬后手动修改的权限控制。

打卡逻辑处理（如隔天打卡、全天未打卡）。

并发场景：

同时发送多条消息或审批请求，验证主键冲突。

多人同时入职同一身份证号，验证唯一约束。

数据一致性：

修改员工姓名后，检查关联表冗余字段是否同步更新。

部门结构调整后，验证招聘表的期望部门是否有效。

二、核心模块实施标准

2.1 智能档案管理

数据导入服务

输入:

协议: HTTP Multipart

文件限制:

- 格式: CSV/Excel

- 大小: ≤100MB

- 必填列: [工号, 姓名, 部门]

处理逻辑:

1. 异步任务队列: Celery(data\_import)

2. 数据清洗:

- 工号正则: ^[A-Z]{2}\d{6}$

- 部门编码验证AD域对照表

3. 异常处理:

- 错误文件生成error\_code.csv

- 三次失败触发SNMP告警(1.3.6.1.4.1.5432.1)

输出:

成功: {"task\_id":"xxx", "success\_count":952}

失败: HTTP 422 + 错误定位

2.2 入职自动化引擎

流程节点定义

xml

<!-- Camunda流程定义 -->

<process id="onboarding">

<sequenceFlow sourceRef="createAccount" targetRef="equipmentRequest"/>

<serviceTask id="syncAD" camunda:expression="${adService.createAccount(applicant)}"/>

<boundaryEvent id="timeoutAlert" attachedToRef="managerInterview">

<timerEventDefinition>

<timeDuration>PT24H</timeDuration>

</timerEventDefinition>

</boundaryEvent>

</process>

2.3 薪酬计算服务

多账套处理

分库策略

CREATE SCHEMA schema\_company\_001;

CREATE TABLE salaries (

id BIGINT PRIMARY KEY,

employee\_id VARCHAR(10) CHECK (employee\_id ~ '^[A-Z]{2}\d{6}$'),

base\_salary DECIMAL(12,2) ENCRYPTED USING 'AES256'

) TABLESPACE company\_001;

三、非功能需求实施标准

3.1 性能保障

# 压力测试命令

jmeter -n -t onboarding.jmx -l report.jtl

-Dthreads=3000 -Drampup=120 -Dduration=3600

3.2 扩展设计

// 设备驱动接口

@FunctionalInterface

public interface DeviceDriver {

default boolean healthCheck() {

return pingDevice();

}

List<AttendanceRecord> fetchRecords(LocalDate date);

}

**七、结论建议**  
本系统采用分阶段交付策略，首期聚焦核心功能打磨，二期构建数据智能底座，形成差异化竞争优势。建议成立跨部门的"技术伦理委员会"，重点关注AI算法的公平性审查，同时建立自动化合规检查流水线，确保系统符合《个人信息保护法》要求。