Computer Science and Application 计算机科学与应用, 2025, 15(10), 137-150
Published Online October 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.1510256

Hans汉斯

基于Python的人事管理系统的设计与实现

闫利淼[1]，马君锡[1]，刘小华[1,2*]

[1]新疆理工职业大学人工智能学院，新疆 喀什
[2]深圳职业技术大学人工智能学院，广东 深圳

收稿日期：2025年9月11日；录用日期：2025年10月13日；发布日期：2025年10月22日

摘　要

传统人事管理以人工操作为主，在考勤核对、薪资核算等日常事务中效率低下且易生差错，更难以适配企业规模扩张后的动态管理需求。本文设计开发了一套人事管理系统，以Python Django为后端框架，搭配MySQL数据库存储数据，通过HTML页面实现前端交互，集成角色权限管控、员工信息维护、考勤汇总、薪资核算发放及部门架构可视化等核心模块。系统采用模块化设计，借助权限设置保障数据安全，界面设计侧重易用性与交互体验。应用验证显示，该系统可显著提升人事管理效率，削减管理成本，为人力资源决策提供扎实的数据支撑。

关键词

人事管理系统，Python，Django，MySQL，Web应用

Design and Implementation of a Python-Based Personnel Management System

Limiao Yan[1], Junxi Ma[1], Xiaohua Liu[1,2*]

[1]College of Artificial Intelligence, Xinjiang Vocational University of Technology, Kashi Xinjiang
[2]College of Artificial Intelligence, Shenzhen Polytechnic University, Shenzhen Guangdong

Received: September 11, 2025; accepted: October 13, 2025; published: October 22, 2025

Abstract

Traditional personnel management primarily relies on manual operations, which leads to inefficiency and high error rates in daily tasks such as attendance verification and salary calculation. Moreover, it struggles to adapt to the dynamic management needs arising from enterprise-scale expansion. This

*通讯作者。

文章引用: 闫利淼, 马君锡, 刘小华. 基于 Python 的人事管理系统的设计与实现[J]. 计算机科学与应用, 2025, 15(10): 137-150. DOI: 10.12677/csa.2025.1510256

paper designs and develops a personnel management system, which uses Python Django as the backend framework, paired with MySQL database for data storage and HTML pages for frontend interaction. The system integrates core modules including role permission management, employee information maintenance, attendance summary, salary calculation and distribution, and department structure visualization. Adopting a modular design, it ensures data security through permission settings, and its interface design emphasizes usability and interactive experience. Application verification shows that the system can significantly improve the efficiency of personnel management, reduce management costs, and provide solid data support for human resource decision-making.

Keywords

Personnel Management System, Python, Django, MySQL, Web Application

1. 引言

在当今竞争激烈的商业环境中，人力资源管理已成为企业核心竞争力的关键支撑，其管理效率与质量直接影响企业的可持续发展节奏[1]。传统人事管理模式高度依赖手工记录、Excel 表格操作，致使数据散落于各类文档与表单之中，不仅查询时需在海量文件里艰难检索，统计工作也极为繁琐，且人工计算极易出错[2]；据相关行业报告显示，约 72%的 HR 将“事务性工作过多”视为最大挑战，像算薪、考勤统计、员工入转调离等流程占据了他们 60%以上的工作精力[3]，大量时间被消耗在基础事务上，难以投入到人才发展、战略规划等核心工作中。并且随着员工数量的增多，人事数据迅猛增长，传统管理手段在数据处理效率、准确性及整合利用上的短板愈发突出，难以契合信息化时代对高效、精准管理的需求[4]，甚至可能因数据误差引发薪资核算争议、考勤记录纠纷等问题，影响企业内部稳定性[5]。

在此背景下，信息化人事管理系统成为破解传统管理困境的核心路径。这类系统通过标准化重塑人事管理流程，实现员工信息录入、考勤统计、薪资核算等环节的自动化处理，将分散数据集中存储于结构化数据库，借助数据处理技术达成快速检索与多维度统计分析，既能减少人工干预带来的误差，又能将 HR 从繁琐事务中解放出来[6]。从技术适配性来看，Python 凭借语法简洁、库资源丰富的优势，在人事信息管理系统开发中展现出显著潜力，已有实践通过 Python 实现职工信息的高效存储与动态查询，大幅提升数据处理效率；而从行业现状来看，国外人事管理系统如 SAP SuccessFactors、Oracle HCM 虽发展成熟，涵盖人力资源全生命周期管理，但普遍存在成本高昂、定制化灵活性不足的问题，中小企业采购后适配成本往往超出预算 20%以上[7]；国内多数人事管理系统或聚焦单一业务场景，或依赖传统技术栈开发，在功能扩展性与新兴技术融合方面存在局限，难以完全满足企业动态变化的管理需求。因此，开发一款基于 Python 技术、适配性强且功能全面的信息化人事管理系统，对提升企业人事管理效能、降低运营成本具有重要的现实意义。

2. 系统需求分析

2.1. 功能需求

2.1.1. 用户认证模块

支持管理员、人事专员、部门主管、普通员工等多角色的登录与注册；

实现用户信息维护、密码修改、权限分配等功能；
记录用户登录日志，确保操作可追溯。

2.1.2. 员工信息管理模块

支持员工基本信息(姓名、性别、年龄、学历等)的录入、查询、修改和删除；
实现员工档案的批量导入导出功能；
提供员工信息的多条件组合查询和统计分析。

2.1.3. 考勤管理模块

支持员工日常签到、签退记录的录入和统计；
统计员工考勤信息，设定考勤规则显示员工考勤状态；
生成员工考勤报表，记录员工统计信息。

2.1.4. 薪资管理模块

支持薪资模板的自定义配置，包括基本工资、绩效工资、津贴、扣款等项目；
实现薪资的自动计算和发放记录管理；
生成员工薪资单，并配置导出功能。

2.1.5. 部门管理模块

支持部门信息的增删改查操作；
统计各部门的人员数量，展示部门结构等关键信息。

2.2. 非功能需求

2.2.1. 安全性需求

采用密码加密存储，防止敏感信息泄露；
实现基于角色的访问控制(RBAC)，确保数据访问安全；
定期自动备份数据库，防止数据丢失。

2.2.2. 可扩展性需求

系统架构采用模块化设计，支持功能模块的灵活增减；
数据库设计预留扩展字段，适应未来业务变化；
支持与企业现有 OA、财务系统等进行数据对接。

3. 系统设计

3.1. 总体架构设计

系统采用 B/S (浏览器/服务器)架构，基于 MVC (模型－视图－控制器)设计模式进行开发，总体架构分为表现层、业务逻辑层和数据访问层，如图 1 所示。

表现层：负责用户界面展示和交互，采用 HTML5、CSS3 和 JavaScript 实现，结合 Django 模板引擎实现动态页面渲染。

业务逻辑层：处理核心业务逻辑，包括用户认证、权限控制、数据校验等，通过 Django 视图和服务类实现。

数据访问层：负责数据的存储和读取，通过 Django ORM 框架与 MySQL 数据库交互，实现数据的持久化存储。

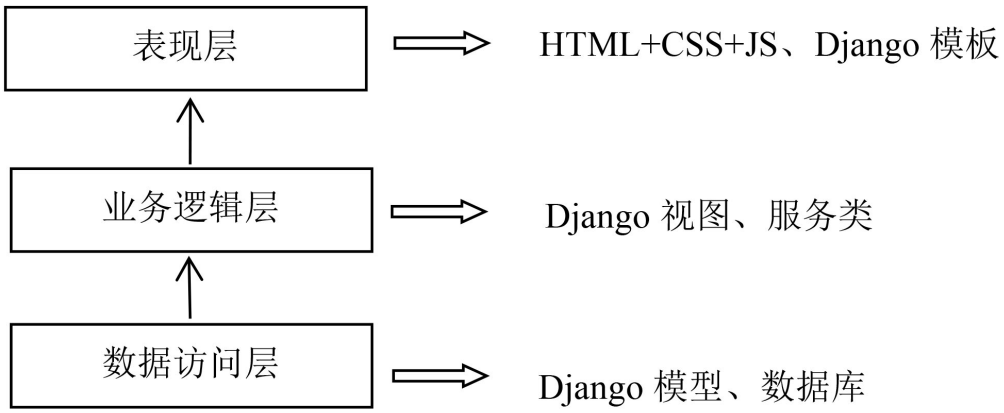

Figure 1. Diagram of the overall system framework
图 1. 系统总体框架图

3.2. 模块设计

系统采用模块化设计，各模块之间通过接口进行通信，保持相对独立。主要功能模块如图 2 所示。

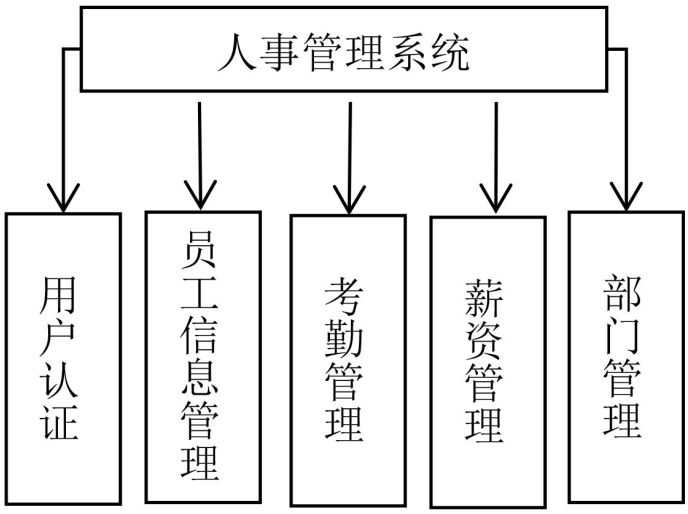

Figure 2. Diagram of the system's main functional modules
图 2. 系统主要功能模块图

用户认证模块：负责用户身份认证和权限管理；
员工信息管理模块：处理员工基本信息的全生命周期管理；
考勤管理模块：管理员工的出勤记录和考勤流程；
薪资管理模块：实现薪资计算、发放和统计分析；
部门管理模块：维护部门信息和组织架构；

3.3. 数据库设计

3.3.1. 用户表(User)

存储系统用户信息，包括用户名、密码、角色等字段，如表 1 所示。

Table 1. User table structure
表 1. 用户表结构

字段名	数据类型	长度	主键	描述
id	int	11	是	用户 ID
username	varchar	50	否	用户名
password	varchar	128	否	加密密码
role	int	2	否	用户角色
is_active	tinyint	1	否	是否激活
create_time	datetime	-	否	创建时间

3.3.2. 员工表(Employee)

存储员工基本信息，如表 2 所示。

Table 2. Employee table structure
表 2. 员工表结构

字段名	数据类型	长度	主键	描述
id	int	11	是	员工 ID
emp_no	varchar	20	否	员工编号
name	varchar	50	否	姓名
gender	tinyint	1	否	性别
birthday	date	-	否	出生日期
department_id	int	11	否	所属部门 ID
position	varchar	50	否	职位
hire_date	date	-	否	入职日期
status	tinyint	1	否	在职状态

3.3.3. 考勤表(Attendance)

存储员工考勤记录信息，如表 3 所示。

Table 3. Attendance table structure
表 3. 考勤表结构

字段名	数据类型	长度	主键	描述
id	int	11	是	记录 ID
emp_id	int	11	否	员工 ID
check_in_time	datetime	-	否	签到时间
check_out_time	datetime	-	否	签退时间
status	tinyint	1	否	签到状态
check_date	date	-	否	考勤日期

3.3.4. 薪资表(Salary)

存储员工薪资信息，如表 4 所示。

Table 4. Salary table structure
表 4. 薪资表结构

字段名	数据类型	长度	主键	描述
id	int	11	是	薪资 ID
emp_id	int	11	否	员工 ID
basic_salary	decimal	10.2	否	基本工资
performance_salary	decimal	10.2	否	绩效工资
allowance	decimal	10.2	否	津贴
deduction	decimal	10.2	否	扣款
total_salary	decimal	10.2	否	总薪资
pay_month	varchar	6	否	发放月份

3.3.5. 部门表(Department)

存储员工部门信息，如表 5 所示。

Table 5. Department table structure
表 5. 部门表结构

字段名	数据类型	长度	主键	描述
id	int	11	是	部门 ID
dept_no	varchar	20	否	部门编号
name	varchar	50	否	部门名称
parent_id	int	11	否	父部门 ID
manager_id	int	11	否	部门经理 ID
description	text	-	否	部门描述

4. 系统实现

4.1. 开发环境

操作系统：Windows 11；
开发语言：Python 3.11；
后端框架：Django 4.0；
数据库：MySQL 9.4.0；
前端框架：Bootstrap 5；
开发工具：PyCharm 2024.3.1.1。

4.2. 核心模块实现

4.2.1. 用户认证模块

用户认证模块基于 Django 内置的认证系统实现，扩展了角色权限功能，关键代码如图 3 所示。在用户认证过程中，用户角色主要分为管理员、人事专员、部门主管以及普通员工，各种角色都有对应的权限，比如人事专员是员工数据的主要管理者，而普通员工等则无此权限，图 3 中员工数据导入功能权限

仅赋予人事专员。在登录过程中，首先验证输入，其次核对账号密码，登录成功后使用日志功能记录下用户登录的时间信息，并以 Json 格式数据响应登录请求，跳转回系统主界面，表示登录成功。

```python
class IsHRUser(permissions.BasePermission):  17 用法
    """只允许HR用户访问"""
    def has_permission(self, request, view):
        # 检查用户是否已登录且角色是HR
        return request.user and request.user.is_authenticated and request.user.role == 'HR'
/api/employees/import/
class EmployeeImportView(APIView):  1 个用法
    """员工数据导入视图"""
    permission_classes = [IsHRUser]
    def post(self, request):
        serializer = EmployeeImportSerializer(data=request.data)
        if serializer.is_valid():
            file = serializer.validated_data['file']
            try:...
            except Exception as e:
                return Response( data: {'error': str(e)}, status=http_status.HTTP_400_BAD_REQUEST)

        return Response(serializer.errors, status=http_status.HTTP_400_BAD_REQUEST)
```

Figure 3. User permission control code
图 3. 用户权限控制代码

4.2.2. 员工信息管理模块

员工信息管理模块实现了员工信息的增删改查等功能，支持批量导入导出。实现批量导入导出的主要代码如图 4、图 5 所示。在批量导入员工数据流程中，如图 4 所示，遍历 EXCEL 文件中的每一行数据后，先查找数据行中的部门是否存在，然后创建或更新员工，该过程需要处理性别、学历以及状态等信息。

```python
# 遍历数据行
for index, row in df.iterrows():
    try:
        # 查找部门
        dept_name = str(row.get( key: '部门',  default: '')).strip()
        department = None
        if dept_name:
            department = Department.objects.filter(name=dept_name).first()
        # 创建或更新员工
        emp_no = str(row.get( key: '员工编号',  default: '')).strip()
        if not emp_no:
            error_count += 1
            errors.append(f"行 {index+1}: 员工编号不能为空")
            continue
        # 处理性别
        gender_map = {'男': Employee.Gender.MALE, '女': Employee.Gender.FEMALE, '其他': Employee.Gender.OTHER}
        gender = gender_map.get(str(row.get( key: '性别',  default: '')).strip(), Employee.Gender.MALE)
        # 处理学历
        education_map = {...}
        education = education_map.get(str(row.get( key: '学历',  default: '')).strip(), Employee.Education.BACHELOR)
        # 处理状态
        status_map = {...}
        status = status_map.get(str(row.get( key: '状态',  default: '')).strip(), Employee.Status.PROBATION)
        # 创建或更新员工记录
        Employee.objects.update_or_create(
            emp_no=emp_no,
            defaults={...}
        )
        success_count += 1
```

Figure 4. Bulk import code for employee information
图 4. 员工信息批量导入代码

在批量导出员工数据流程中，如图 5 所示，首先通过获取查询参数后构建组合查询集，比如指定部门、状态等，也可以具体到名字、员工号以及手机号等具体信息，以便取出筛选过后的员工数据，其次序列化数据后转换为 DataFrame 格式，选择需要导出的列进行排序后，生成可下载的 EXCEL 文件，用户可通过浏览器端的下载链接将 Excel 文件格式的员工导出数据保存到本地备用。

```
# 获取查询参数
department_id = request.query_params.get('department')
status = request.query_params.get('status')
search = request.query_params.get('search')
# 构建查询集
queryset = Employee.objects.all()
# 应用过滤
if department_id:
    queryset = queryset.filter(department_id=department_id)
if status:
    queryset = queryset.filter(status=status)
if search:...
# 序列化数据
serializer = EmployeeSerializer(queryset, many=True)
data = serializer.data
# 转换为DataFrame
df = pd.DataFrame(data)
# 选择需要导出的列并排序
columns = [...]
df = df[columns]
# 重命名列名
df.columns = [...]
# 创建Excel文件
output = BytesIO()
with pd.ExcelWriter(output, engine='openpyxl') as writer:...
# 准备响应
output.seek(0)
response = HttpResponse(
```

Figure 5. Bulk export code for employee information
图 5. 员工信息批量导出代码

4.2.3. 考勤管理模块

考勤管理模块实现了员工签到签退、考勤统计等功能，关键代码如图 6 所示。

在员工考勤情况统计流程中，只有人事专员或部门经理才有权限进入；首先需要解析参数，获取需统计月份的天数、指定部门等信息，构建组合查询集，按员工进行分组统计，筛选出指定部门中所有员工的考勤记录，并进行归纳整理，形成完整统计信息。在该流程中，设定了两个参数值来区分员工考勤情况的异常与否，同时设定了三种考勤状态来标示员工考勤统计信息，例如当考勤异常天数占整个月的比率超过 0.1 未达到 0.5 时，考勤情况标注为“需关注”，当比率超过 0.5 时，则直接标注为“异常”，其他情况下异常考勤比率未超过 0.1，标注为“正常”。

```python
/api/attendance/statistics/monthly/
class MonthlyAttendanceStatisticsView(APIView):  1 个用法
    """月度考勤统计视图"""
    permission_classes = [IsHRUser | IsDepartmentManager]
    def get(self, request):
        year_month = request.query_params.get('month', timezone.now().strftime('%Y-%m'))
        try:
            year, month = map(int, year_month.split('-'))
            first_day = date(year, month,  day: 1)
            last_day = (first_day + relativedelta(months=1)) - relativedelta(days=1)
            _, days_in_month = calendar.monthrange(year, month)
        except (ValueError, TypeError):...
        # 获取部门和员工筛选条件
        department_id = request.query_params.get('department')
        employee_id = request.query_params.get('employee')
        # 构建查询集
        queryset = AttendanceRecord.objects.filter(
            check_date__gte=first_day,
            check_date__lte=last_day
        )
        if department_id:...
        if employee_id:...
        # 按员工分组统计
        employee_stats = []
        employees = Employee.objects.all()
        if department_id:...
        if employee_id:...
        # 统计考勤记录
        employee_ids = queryset.values_list( *fields: 'employee_id', flat=True).distinct()
        employees = employees.filter(id__in=employee_ids)
```

Figure 6. Employee attendance statistics code
图 6. 员工考勤情况统计代码

4.2.4. 薪资管理模块

薪资管理模块主要实现薪资计算、发放记录管理等功能，关键代码如图 7 所示。

以薪资计算为例，该部分作为薪资管理模块的核心内容，首先通过解析请求中的参数，获取到查询月份、部门以及薪资模板等信息，薪资模板可以在后台进行配置，如果没有成功获取到薪资模板，则使用默认模板，模版中包含员工的基础工资、绩效工资基数、津贴、奖金、社保以及公积金等配置信息。

其次计算每位员工的薪资，该过程中需要先检查是否已经存在指定员工于指定月份下的薪资记录，如果存在的话对该条薪资记录进行更新，并记录更新状态；接下来通过员工 ID、起止日期来筛选出员工在指定月份的考勤记录，计算出勤天数和加班时长，其中的加班时长按每天工作时长与标准 8 小时工作时长的差值来计算。

最后根据员工出勤率对其基本工资、效益工资等薪酬项目进行折算，得到应发工资，其中员工出勤率即为出勤天数与当月总天数的比值，当月的基础薪资、效益工资、津贴等部分需要乘以出勤率进行折

算；在此基础上扣除个税后，得到最终的实发工资，其中的个税计算遵循国家收税标准，应发收入扣除纳税起征工资、社保以及公积金等款项后得到纳税工资，纳税工资如果为负数，则自动调整为 0，然后乘以税率，得到个税额。

```
# 计算薪资明细
month_days = (last_day - first_day).days + 1
attendance_rate = Decimal(str(work_days)) / Decimal(str(month_days))
basic_salary = (template.basic_salary / month_days) * work_days
performance_salary = template.performance_basic * attendance_rate
allowance = template.allowance * attendance_rate
bonus = template.bonus * attendance_rate
overtime_pay = ((template.basic_salary / month_days / CONFIG_WORK_HOUR)
                *Decimal(str(overtime_hours)) * CONFIG_SALARY_OVER_RATIO)
insurance = template.insurance
provident_fund = template.provident_fund
reward = performance_salary + allowance + bonus + overtime_pay
total_income = basic_salary + reward
# 计算个税
taxable_income = total_income - insurance - provident_fund - CONFIG_TAX_BASE_SALARY
tax = taxable_income * (template.tax_rate / 100) \
    if taxable_income > CONFIG_TAX_BASE_SALARY else Decimal('0')
total_deduction = insurance + provident_fund + tax
```

Figure 7. Code for calculating employee salary details
图 7. 计算员工薪资明细代码

4.2.5. 部门管理模块

部门管理模块主要实现部门信息展示、统计以及搜索等功能。

以查询部门信息为例，首先通过 prefetch_related 进行优化查询，获取部门信息。该步骤中的 prefetch_related 是解决“多对一”和“多对多”关系中频繁出现的“N + 1 查询问题”的有效方法，以当前查询所有部门关联的员工为例，默认情况下 Django 需要先生成 1 条 SQL 语句查询所有部门，然后为每个部门单独生成 1 条 SQL 语句查询其下辖的所有员工，假设部门有 N 个，那么该过程就会产生 N + 1 条 SQL 语句，查询次数较多，影响性能；当使用 prefetch_related 时，第一步与默认情况相同，第二步则直接通过外键关联条件一次性获取所有部门关联的员工，只需要执行 1 条 SQL 语句，部门的增加也不会额外增加查询次数，可以大幅减少数据库交互开销。

其次构造树形结构，收集各部门以及其下辖的所有员工信息，传递回前端，页面会利用 Canvas 等画布工具对部门信息进行统计展示，以树状结构清晰表示出部门隶属关系以及部门统计信息。

4.3. 实现效果展示

4.3.1. 系统主页面展示

用户默认进入主页面，可以登录后使用相关功能，使用过程中会涉及权限控制，主页面分成四大模块，每个模块都附带功能说明部分；用户在点击后即可进入对应功能区，右上角会显示用户名，用户可以很方便地进行登录或下线操作。系统主界面如图 8 所示。

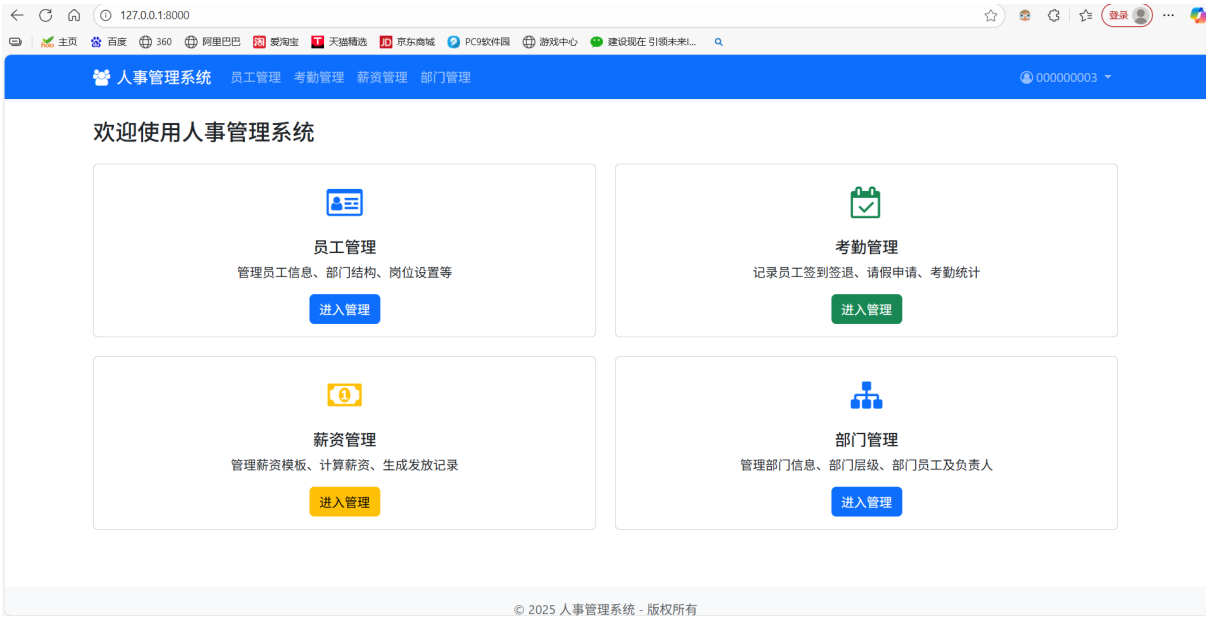

Figure 8. System main page
图 8. 系统主页面

4.3.2. 员工管理功能展示

在员工管理界面，满足权限的用户可通过搜索功能查看员工信息，默认展示所有员工信息；通过状态列表可清晰观察到员工在职状态，点击查看按钮可查看员工的完整信息，点击编辑按钮则可对员工信息进行编辑并保存；右上角的添加员工按钮可以进行添加操作，同时可通过导入导出按钮对员工信息进行批量添加或者备份保存，具体情况如图 9 所示。

Figure 9. Employee management function page
图 9. 员工管理功能页面

4.3.3. 考勤管理功能展示

在考勤管理界面，满足权限的用户可以通过指定月份和部门，查看对应的员工考勤统计情况，如图 10 所示。考勤管理模块根据员工的异常出勤天数与后台系统配置的比例系数作对比，可总结出员工的考勤状态，方便管理人员了解员工的考勤情况；正常率是指正常出勤的员工数占全体员工的比例，统计数据均以图表形式直观展示；用户通过点击导出统计按钮则可将考勤情况以表格形式保存，下载到本地。

Figure 10. Attendance management function page
图 10. 考勤管理功能页面

4.3.4. 薪资管理功能展示

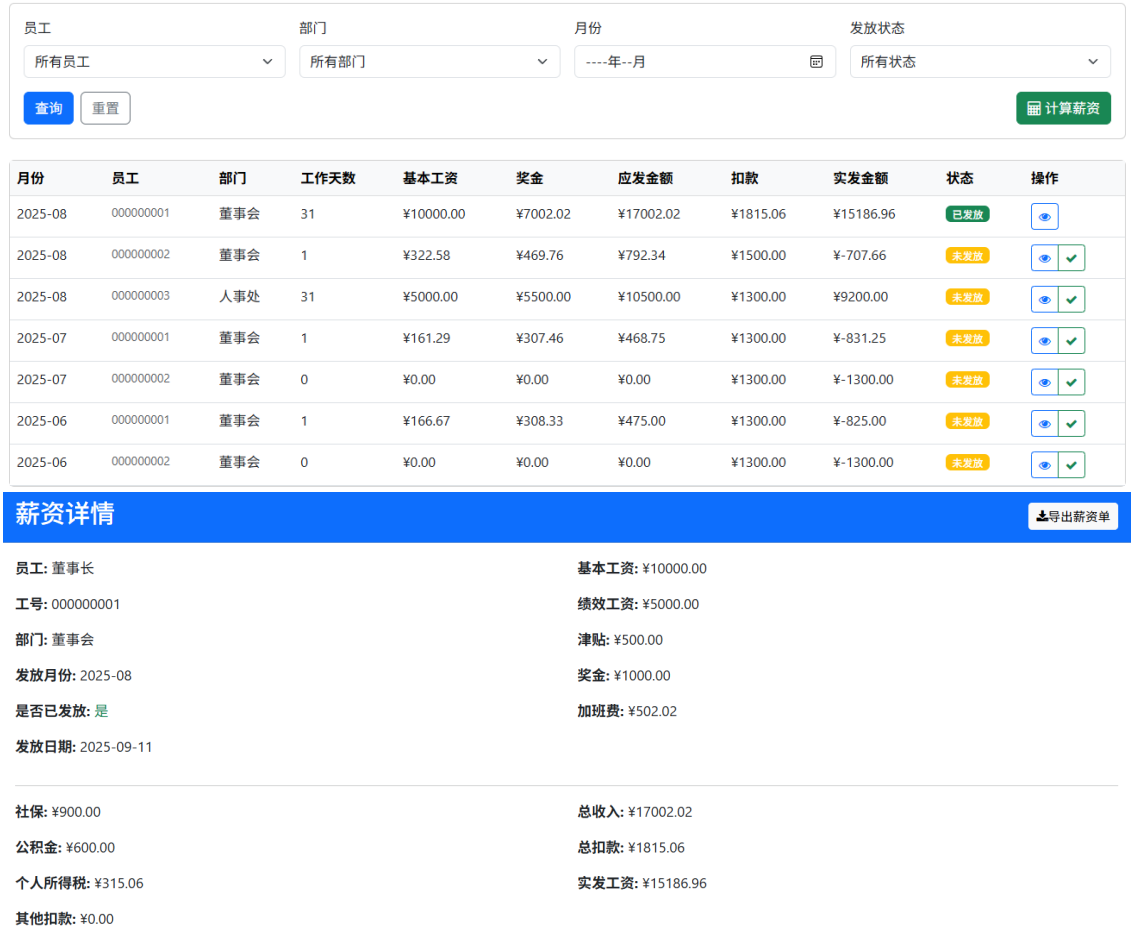

Figure 11. Salary management function page
图 11. 薪资管理功能页面

在薪资管理界面，可查看员工薪资详情以及发放情况，如图 11 所示。满足权限的用户可以先通过点击计算薪资按钮为指定部门的员工生成工资单，该过程中需要选定计算月份、部门以及薪资模板，点击开始计算后后台系统便会生成对应的工资单；在薪资管理界面可通过点击操作列中的查看图标，查询指定员工的薪资详情，薪资详情里会列出员工薪资构成明细，同时可通过点击右上角的导出薪资单按钮对薪资单详情进行保存；员工薪资计算生成后默认为未发放状态，可通过点击操作列中的发放图标完成薪资的发放，发放完成后薪资单状态会更新。

4.3.5. 部门管理功能展示

在部门管理界面，满足权限的用户通过搜索功能查看指定部门结构以及部门统计信息，默认展示所有部门，如图 12 所示；部门结构图会以树状层次结构将该部门以及其下辖的子部门都标示出来，部门统计图则以柱状图清晰展示部门员工数量；部门详细统计信息则在页面底部，每一行数据都会提供部门名称、上级部门、部门经理与员工数量等信息，同时提供编辑和查看功能；页面右上角则提供添加部门以及导出部门信息的功能。

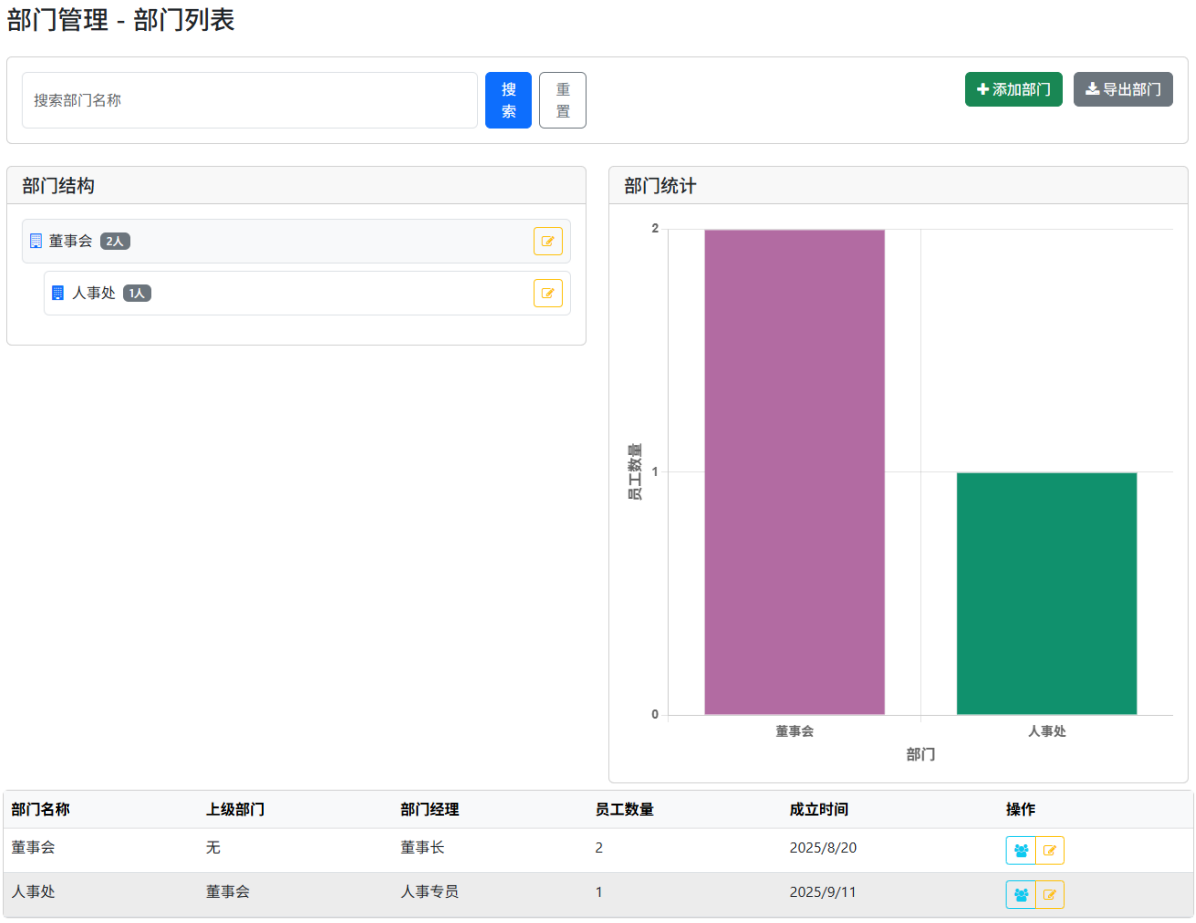

Figure 12. Department management function page
图 12. 部门管理功能页面

5. 系统测试

本次测试采用黑盒测试策略，以用户操作场景为核心验证维度，重点检测员工端与管理后台的界面

响应逻辑及数据交互准确性。测试覆盖点击操作、表单输入、数据提交、页面跳转、条件筛选、文件导出等全量交互行为，聚焦于三大核心业务链路：员工信息管理(含新增时工号自动补全为 00000000X 格式、用户与员工数据关联校验)、部门架构维护(含上下级部门树形展示、员工数量统计图表渲染、部门数据导出完整性)、薪资流程处理(含薪资项自动计算、发放状态同步、薪资单导出完整性)。特别针对关键节点进行验证，包括工号生成规则有效性、部门删除时的子部门存在性校验、柱状图数据与实际统计一致性、薪资计算逻辑准确性等。测试结果表明，各功能模块响应及时，数据交互无异常，未出现逻辑漏洞、数据失配或界面卡顿现象，管理操作流程顺畅，系统功能实现与需求预期高度吻合；在安全性方面，通过模拟 SQL 注入等方式对系统进行安全性测试，结果表明，系统能够有效抵御常见的网络攻击，用户密码采用加密存储，敏感操作需要权限验证，符合安全性需求。

6. 结语

本文设计并实现了一套基于 Python 的人事管理系统，用 Django 框架与 MySQL 数据库设计实现用户管理、员工信息维护、考勤统计、薪资核算与部门架构管理等核心功能，并通过系统测试验证了系统的有效性与可靠性，为人事管理信息化改造与升级提供了有效解决方案。

未来将开发移动端应用，支持员工通过手机完成签到、签退等高频操作；集成人脸识别功能，进一步提升考勤管理的便捷性；后续将持续迭代功能、优化用户体验，确保系统高效助力人事管理。

参考文献

[1] 刘秀丽. 基于 Python 语言的职工信息管理系统设计与实现[J]. 现代信息科技, 2023, 7(23): 35-41.

[2] 毛娟. Python 中利用 xlwings 库实现 Excel 数据合并[J]. 电脑编程技巧与维护, 2023(9): 61-62, 134.

[3] 王亮, 左文涛. 大数据收集与分析中 Python 编程语言运用研究[J]. 计算机产品与流通, 2020(1): 22.

[4] 张华, 翟新军, 胥勇, 等. Python 在集控大数据应用的研究[J]. 价值工程, 2023, 42(21): 84-86.

[5] 尹周军. 基于 Python 语言的钢铁企业管理平台开发与应用[J]. 天津冶金, 2021(3): 54-56.

[6] 沈杰. 基于 Python 的数据分析可视化研究与实现[J]. 科技资讯, 2023, 21(2): 14-17, 54.

[7] Nakagawa, Y., Abe, Y. and Isobe, M. (2024) Ticketing and Crowd Management System for Attraction Facilities: An Aquarium Case Study. *Journal of Disaster Research*, **19**, 303-315. https://doi.org/10.20965/jdr.2024.p0303