**《专业生产劳动1》**

课程设计报告

（ 2024 —2025学年 第1学期）

题 目**：**  **学生宿舍报修信息管理系统**

专 业**： 计算机应用工程（专升本）**

班 级**： 2405班**

学 号**： 2431020120511**

姓 名： **彭文成**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2024 | 年 | 12 | 月 | 30 | 日 |

# 一、引言

### **（一）项目背景**

在学校宿舍管理中，传统的报修方式效率低下且容易出现信息管理混乱的问题。为了提高报修管理的效率和准确性，方便学生、报修管理员和维修员之间的信息交互，特开发此学生宿舍报修信息管理系统

1. **项目目标**

本系统旨在实现学生宿舍报修信息的电子化管理，提高报修流程的效率和透明度，具体目标包括：

1. 学生能够方便快捷地提交报修信息，并可随时查看报修状态。
2. 报修管理员可以高效地管理学生和维修员信息，灵活查看和筛选报修信息，并准确派工。
3. 维修员能够及时获取派工任务，详细记录维修过程和结果，确保维修工作的顺利完成。

## **二、系统需求分析**

### **（一）功能需求**

1. ****学生功能需求****
   1. ****登录****：通过学号和密码登录系统，确保个人信息安全。
   2. ****报修信息录入与管理****
      1. 录入报修内容（如水电故障、家具损坏等）、楼号、房号、联系电话等信息，可选择上传现场图片辅助说明故障情况。
      2. 保存未提交的报修信息，方便后续修改和补充。
      3. 提交报修信息后，系统自动将状态设置为待受理。
   3. ****个人报修信息查看****：查看自己提交的报修信息及其处理状态。
2. ****报修管理员功能需求****
   1. ****登录****：使用管理员账号和密码登录系统，获取管理权限。
   2. ****学生管理****：对学生信息进行增删改查操作，包括学号、姓名、专业、班级等信息。
   3. ****维修员管理****：管理维修员信息，如工号、姓名、联系方式、擅长维修类型等，可进行添加、修改、查询和删除操作。
   4. ****报修信息管理****
      1. 按状态（待受理、已派工、维修结束等）筛选报修信息，方便快速定位需要处理的任务。
      2. 按时间倒序排序报修信息，优先处理最新提交的报修。
      3. 查看报修详细信息，包括报修内容、提交时间、学生信息等。
   5. ****派工操作****：对于待受理的报修信息，选择指定维修员进行派工，同时将报修信息状态更新为已派工，并生成派工单，派工单初始状态为未处理。
3. ****维修员功能需求****
   1. ****登录****：凭借工号和密码登录系统，进入工作界面。
   2. ****派工信息查看****：查看分配给自己的派工信息，包括报修单号、报修地点、报修人信息、派工时间等，了解维修任务详情。
   3. ****维修单填写与提交****
      1. 输入报修单单号，系统自动填充报修地点和报修人信息。
      2. 详细填写故障分析、维修过程和维修结果等内容。
      3. 提交维修单后，将派工单状态更新为已处理，报修单状态更新为维修结束。

### **（二）数据需求**

1. ****学生信息****：学号、姓名、专业、班级、密码等。
2. ****报修管理员信息****：管理员账号、密码等。
3. ****维修员信息****：工号、姓名、联系方式、擅长维修类型、密码等。
4. ****报修信息****：报修单号、楼号、房号、报修内容、报修人、报修时间、联系电话、状态（待受理、已派工、维修结束等）、现场图片（可选）。
5. ****派工单信息****：派工单编号、报修单号、维修员工号、派工时间、状态（未处理、已处理）。

### **（三）性能需求**

1. 系统应具备快速响应能力，确保学生提交报修信息、管理员和维修员操作的及时性，一般操作响应时间应控制在 1 - 3 秒内。
2. 支持高并发访问，能够满足学校宿舍区大量学生同时使用的需求，保证系统稳定运行，避免出现卡顿或崩溃现象。
3. 数据存储安全可靠，防止数据丢失或损坏，定期进行数据备份，并提供数据恢复功能。

### **（四）安全需求**

1. 用户身份认证：采用安全的登录验证机制，防止非法用户登录系统。对密码进行加密存储，保障用户密码安全。
2. 数据访问权限控制：根据用户角色（学生、报修管理员、维修员）分配不同的数据访问权限，确保用户只能操作其权限范围内的数据。
3. 防止数据篡改：在数据传输和存储过程中，采用加密技术和数据校验机制，防止数据被恶意篡改。

## **三、系统设计**

### **（一）总体架构设计**

1. 本系统采用 B/S 架构，学生、报修管理员和维修员通过浏览器访问系统，无需安装额外客户端软件，方便使用和维护。
2. 系统架构分为三层：表现层、业务逻辑层和数据访问层。表现层负责与用户交互，展示界面和接收用户输入；业务逻辑层处理系统的核心业务逻辑，如报修流程、派工逻辑等；数据访问层负责与数据库进行数据交互，实现数据的存储、查询和更新操作。

### **（二）功能模块设计**

1. ****学生模块****
   1. ****登录子模块****：负责验证学生登录信息，与数据库中的学生信息进行比对，认证成功后建立用户会话，记录用户登录状态。
   2. ****报修信息管理子模块****
      1. 报修信息录入界面：提供表单供学生填写报修信息，包括楼号、房号、报修内容、联系电话等，支持现场图片上传功能。
      2. 信息保存与提交功能：未提交的报修信息可临时保存至本地缓存或数据库，提交后触发系统工作流，将报修信息状态更新为待受理，并将数据持久化存储。
      3. 个人报修信息查询界面：以列表形式展示学生自己提交的报修信息，包括报修单号、报修时间、状态等关键信息，点击可查看详细内容。
2. ****报修管理员模块****
   1. ****登录子模块****：验证管理员登录账号和密码，确保管理员身份合法，登录成功后进入管理控制台。
   2. ****学生管理子模块****
      1. 学生信息维护界面：提供学生信息的添加、修改、删除和查询功能，可批量导入或导出学生数据，方便管理。
      2. 学生信息验证与约束：对输入的学生信息进行合法性验证，如学号唯一性、姓名格式等，确保数据准确无误。
   3. ****维修员管理子模块****
      1. 维修员信息管理界面：实现维修员信息的全生命周期管理，包括添加新维修员、修改维修员信息、查询维修员列表、删除维修员等操作。
      2. 维修员技能与任务分配：记录维修员擅长维修类型，为派工操作提供数据支持，确保任务分配合理高效。
   4. ****报修信息管理子模块****
      1. 报修信息查询与筛选界面：提供多维度的查询和筛选条件，如按状态、时间范围、楼号等，方便管理员快速定位目标报修信息。
      2. 派工操作界面：在待受理报修信息列表中，选择维修员进行派工操作，生成派工单并更新相关数据状态，同时记录派工历史记录，便于跟踪和追溯。
3. ****维修员模块****
   1. ****登录子模块****：验证维修员身份信息，登录成功后进入维修工作界面，展示与维修员相关的任务和信息。
   2. ****派工信息查看子模块****：以列表形式展示分配给维修员的派工任务，包括报修单号、报修地点、报修人、派工时间等详细信息，方便维修员了解任务详情。
   3. ****维修单管理子模块****
      1. 维修单填写界面：根据报修单号自动加载报修地点和报修人信息，维修员填写故障分析、维修过程和维修结果等内容，支持富文本编辑和图片上传功能。
      2. 维修单提交与状态更新：提交维修单后，系统自动将派工单状态更新为已处理，报修单状态更新为维修结束，并通知相关人员维修结果。

### **（三）数据库设计**

1. ****数据库概念结构设计****
   1. 通过 E - R 图描述实体之间的关系，主要实体包括学生、报修管理员、维修员、报修信息、派工单等。例如，学生与报修信息之间是一对多关系（一个学生可以提交多个报修信息），维修员与派工单之间是一对多关系（一个维修员可以接收多个派工单），报修信息与派工单之间是一对一关系（一个报修信息对应一个派工单）。
2. ****数据库逻辑结构设计****
   1. 根据 E - R 图设计数据库表结构，主要包括以下几个表：
   2. ****学生表（student）****：学号（主键）、姓名、专业、班级、密码等字段。
   3. ****报修管理员表（admin）****：管理员账号（主键）、密码等字段。
   4. ****维修员表（maintainer）****：工号（主键）、姓名、联系方式、擅长维修类型、密码等字段。
   5. ****报修信息表（repair\_info）****：报修单号（主键）、楼号、房号、报修内容、报修人（关联学生表学号）、报修时间、联系电话、状态等字段。
   6. ****派工单表（work\_order）****：派工单编号（主键）、报修单号（关联报修信息表报修单号）、维修员工号（关联维修员表工号）、派工时间、状态等字段。

### **（四）接口设计**

1. ****学生接口****
   1. ****登录接口（/student/login）****：接收学生学号和密码，验证身份后返回登录结果（成功或失败）及相关信息（如学生姓名、专业等）。
   2. ****报修信息录入接口（/student/repair/submit）****：接收学生提交的报修信息，包括楼号、房号、报修内容、联系电话、现场图片（可选）等，保存并返回报修单号。
   3. ****个人报修信息查询接口（/student/repair/query）****：根据学生学号查询其提交的报修信息列表，返回包含报修单号、报修时间、状态等信息的列表。
2. ****报修管理员接口****
   1. ****登录接口（/admin/login）****：验证管理员账号和密码，登录成功后返回管理员相关信息和权限令牌。
   2. ****学生管理接口（/admin/student/\*）****：包括添加学生信息（/admin/student/add）、修改学生信息（/admin/student/update）、删除学生信息（/admin/student/delete）、查询学生信息（/admin/student/query）等接口，用于对学生信息的全面管理。
   3. ****维修员管理接口（/admin/maintainer/\*）****：提供类似学生管理接口的功能，用于管理维修员信息，如添加（/admin/maintainer/add）、修改（/admin/maintainer/update）、删除（/admin/maintainer/delete）、查询（/admin/maintainer/query）等操作。
   4. ****报修信息管理接口（/admin/repair/\*）****
      1. 报修信息查询接口（/admin/repair/query）：支持多条件查询和筛选，如按状态、时间等，返回符合条件的报修信息列表。
      2. 派工接口（/admin/repair/dispatch）：接收维修员工号和报修单号，进行派工操作，生成派工单并更新相关状态信息。
3. ****维修员接口****
   1. ****登录接口（/maintainer/login）****：验证维修员身份，登录成功后返回维修员相关信息和工作任务列表。
   2. ****派工信息查看接口（/maintainer/workorder/query）****：查询分配给维修员的派工任务列表，返回包含报修单号、报修地点、派工时间等信息的列表。
   3. ****维修单填写与提交接口（/maintainer/repair/complete）****：接收维修员填写的维修单信息，包括故障分析、维修过程、维修结果等，更新维修单和派工单状态。

### **（五）系统安全设计**

1. ****用户认证与授权****
   1. 采用基于 Token 的认证机制，用户登录成功后，服务器颁发 Token 给客户端，客户端在后续请求中携带 Token 进行身份验证。
   2. 基于 Spring Security 框架实现用户权限管理，根据用户角色（学生、报修管理员、维修员）分配不同的权限，对系统资源进行访问控制。
2. ****数据加密****
   1. 对用户密码进行加密存储，使用安全的加密算法（如 BCrypt）对密码进行哈希处理，确保密码在数据库中以密文形式存储，提高密码安全性。
   2. 在数据传输过程中，使用 HTTPS 协议对数据进行加密传输，防止数据在网络中被窃取或篡改。
3. ****输入校验与过滤****
   1. 在前端和后端对用户输入的数据进行严格的校验和过滤，防止 SQL 注入、XSS 攻击等安全漏洞。例如，对用户输入的特殊字符进行转义处理，对输入数据的格式和范围进行检查。
4. ****日志记录与审计****
   1. 记录系统操作日志，包括用户登录、报修信息提交、派工操作、维修单填写等关键操作，以便进行审计和追踪，及时发现和处理安全问题。

## **四、系统实现**

### **（一）开发环境与技术选型**

1. ****开发环境****
   1. 操作系统：Windows/Linux
   2. 开发工具：IntelliJ IDEA
   3. 数据库管理系统：MySQL
   4. 后端框架：Spring Boot（已写，但是不符合要求，这里直接使用JavaEE）
   5. 前端框架：[可选择 Vue.js 或 React.js 等流行前端框架]（已写，但是不符合要求，这里直接使用控制台）
   6. 接口文档工具：Swagger（已使用，但是不符合要求）
2. ****技术选型理由****
   1. Spring Boot：提供了快速搭建企业级 Java 应用程序的能力，简化了配置和部署过程，具有强大的生态系统和丰富的组件支持，能够提高开发效率。
   2. MySQL：是一种成熟、稳定且开源的关系型数据库管理系统，能够满足系统对数据存储和管理的需求，具有良好的性能和可扩展性。
   3. [前端框架]：具有高效的组件化开发模式、良好的用户体验和丰富的 UI 组件库，方便构建美观、易用的前端界面，与后端进行数据交互。
   4. Swagger：方便生成和维护接口文档，便于团队成员之间的协作和接口测试，提高开发效率和接口的稳定性。