# 湖南大学实验报告

| **课程名称** | **程序设计** | | | | |
|---|---|---|---|---|---|
| 项目名称 | 宿舍管理系统 | | | | |
| 项目代码量 | 约3000行 | | | | |
| 完成时间 | 2024年12月30日 | | | | |
| 组长姓名 | 符航康 | 学号 | 202408040228 | 贡献比例 | 70% |
| 组员姓名 | 陈奇乐 | 学号 | 202408040216 | 贡献比例 | 30% |

## 引言

本报告详细介绍了我在开发宿舍管理系统过程中的主要工作、所获得的技术收获以及功能拓展的实现。通过此次项目的开发，我深入理解了面向对象编程（OOP）的核心概念，掌握了C++语言在实际项目中的应用，并且积累了数据库操作与管理的实战经验。项目的成功实施不仅提升了我的编程能力，也增强了我在系统设计与问题解决方面的综合素质。

## 实验目的

通过本次宿舍管理系统的开发，我希望能实现以下几个学习目标：

1. **掌握面向对象编程的基本思想**：通过开发宿舍管理系统，我可以深入理解面向对象编程（OOP）中的封装、继承、多态等核心概念，掌握如何将这些概念应用到实际的项目中，以提高代码的可读性、可维护性和扩展性。
2. **增强数据库操作的实战能力**：在项目中涉及到数据库的管理，我希望通过与 SQLite3 的集成，学习如何在 C++ 中进行数据库的基本操作，比如连接数据库、查询数据、插入和更新数据等，从而掌握数据库的操作技巧。
3. **提升程序设计与模块化能力**：通过将系统分为不同的模块（如学生管理、宿舍管理、维修管理等），我可以更加理解模块化设计的重要性，并学习如何将一个复杂的系统分解为多个相对独立的模块，以提高系统的结构化程度和可维护性。
4. **锻炼问题分析和解决能力**：在项目开发过程中，我将遇到不同的技术难题和错误，如何定位问题并高效解决它们是我非常希望通过这次实验提升的能力。我希望通过调试和修改代码，学会如何从错误日志中找到问题的根源，并有针对性地进行优化和修复。
5. **强化团队协作与沟通能力**：如果是小组合作开发，我希望通过与团队成员的分工协作，提升自己的沟通能力、团队合作能力和项目管理能力，学习如何在团队中进行有效的协作，推动项目的进展。

6. **提升实际编程能力与工程实践经验**：通过从头到尾完成宿舍管理系统的开发，我希望能在实践中锻炼自己的编程能力，尤其是在 C++ 语言的使用上，并加深对软件开发过程的理解，积累实际的项目开发经验。
7. **学习如何设计和实现用户交互界面**：通过设计和实现系统的用户界面，我希望能更好地理解用户体验（UX）设计的基本原则，并掌握如何将其应用到软件开发中，使程序不仅功能完整，还能提供良好的用户体验。

通过本次实验，我希望不仅能提升自己的编程能力和项目开发能力，还能学会如何在实际的工程中进行有效的设计与开发，培养出一名更加全面的软件开发者的素质。

# 项目概述

本宿舍管理系统是一个综合性的程序，旨在为学生和管理员提供高效、便捷的宿舍管理功能。整个系统基于C++语言开发，涵盖了从学生登录、宿舍申请、维修请求到管理员管理用户和宿舍等一系列操作，满足了宿舍管理过程中的各种需求。系统的主界面由一个控制台界面构成，启动后通过加载动画和提示信息逐步引导用户进入系统。通过设置支持UTF-8编码，确保了中文的正常显示。进入主菜单后，用户可以选择进入学生菜单或管理员菜单，系统在其中进行不同角色的功能处理。该设计遵循用户友好、流程清晰的原则，用户可以通过清晰的菜单选项完成各自的操作。所有菜单选项都采用了逐行输出的形式，并结合了慢速打印、清屏等效果，提升了用户体验。

系统的核心功能可分为两大类：学生功能和管理员功能。在学生功能方面，系统首先提供了学生登录功能。学生需要输入学号和密码进行身份验证。如果学生的密码是默认密码，系统会提示学生进行密码修改。登录成功后，学生可以通过菜单进入宿舍信息查询、住宿调整申请、维修请求提交等功能。学生能够查看宿舍楼和房间的详细信息、申请调整住宿、提交维修请求以及查看系统通知和自己的请求记录。此外，学生还可以在系统内修改自己的密码，确保账户安全性。

管理员功能则更为复杂，主要包括宿舍楼管理、用户管理、报表生成、维修处理等多个模块。管理员通过登录界面输入账号和密码进行身份验证。成功登录后，管理员将进入包含宿舍楼管理、用户管理、报表生成、维修处理中心等功能的菜单。宿舍楼管理模块可以进行宿舍楼信息的添加、删除、修改和查询，管理员可以根据需求对宿舍楼进行灵活配置。在用户管理模块中，管理员可以对学生进行增、删、查、改等操作，包括查看用户信息、修改密码等。在维修处理模块中，管理员能够处理学生提交的维修请求，确保宿舍设施的正常运行。管理员还可以生成报表，查看宿舍的使用情况和入住率，帮助学校管理者进行决策。

在设计上，系统对用户输入进行了严格的验证，避免了因输入错误导致的系统崩溃。每个操作都伴随着详细的提示信息，确保用户在操作过程中的顺畅与明确。所有的菜单选项都采用了相同的结构，操作逻辑简洁且一致，用户可以快速熟悉并进行各项功能的操作。

数据的处理上，系统依赖一个核心的UserManager类来管理用户的操作。学生和管理员的所有交互信息，包括登录验证、用户信息查看、密码修改等，都通过UserManager进行管理。该类设计了多个函数来处理不同的任务，如学生注册、用户登录、密码修改等，同时也提供了丰富的查询

和管理接口供管理员使用。例如，在宿舍管理模块中，管理员可以轻松地对宿舍楼、房间进行管理，操作灵活且高效。

## 数据库模块设计与实现

为了确保系统能够高效、安全地存储和管理大量的数据，本宿舍管理系统采用了SQLite数据库进行数据存储。数据库模块的设计目标是实现宿舍信息、用户信息、维修请求和费用记录的持久化存储，并通过SQL查询和更新操作提供快速的数据访问。SQLite是一个轻量级的嵌入式数据库，适合于本项目的规模，不需要复杂的数据库管理系统，且具备较好的性能。

数据库模块的核心部分是`Database`类，它通过C++的SQLite3库与数据库进行交互。该类封装了所有数据库操作，如数据库连接、数据查询、数据插入、更新和删除等功能。它提供了灵活的接口，支持带参数的查询和事务管理，确保操作的原子性和数据一致性。通过数据库模块，系统能够自动维护学生信息、宿舍房间的使用情况以及维修请求的记录。管理员可以随时查看数据库中的数据，确保系统能够实时反映最新的操作。

在系统启动时，`Database`类会打开数据库文件，并初始化数据库连接。通过执行SQL语句，系统可以查询学生的住宿信息、房间的使用状态等数据。当学生或管理员进行相关操作时，数据库模块将根据操作类型（插入、更新、查询等）执行相应的SQL语句，并返回操作结果。例如，学生申请调整宿舍时，系统会查询当前宿舍楼和房间的使用情况，确保该操作不会导致宿舍房间超载。

为了防止SQL注入攻击，系统在所有数据库查询中都使用了参数化查询，通过SQLite的`sqlite3_bind_*`系列函数将用户输入的参数绑定到SQL语句中，从而避免了直接拼接字符串带来的安全风险。

## 开发过程中的问题与解决方案

在宿舍管理系统的开发过程中，我遇到了多方面的挑战，主要集中在输入验证、数据库操作以及用户体验优化上。

首先，在用户输入的验证与错误处理方面，最初系统在接收用户输入时未能有效处理非法输入，导致程序在输入非预期字符时崩溃。为了解决这一问题，我在每个输入环节添加了循环和条件判断，确保用户输入的数据有效。通过使用`cin.fail()`清除输入缓冲区，并提示用户重新输入，系统的稳定性得到了显著提升。

其次，在数据库操作过程中，由于涉及复杂的SQL查询和多表联接，查询结果有时未能按预期返回。通过逐步调试SQL语句，输出调试信息并验证每个查询的正确性，我成功优化了查询结构，确保查询结果的准确性。此外，在处理带参数的查询时，初期出现了绑定参数失败的情况，经过深入学习SQLite3的文档和实践操作，我掌握了正确的参数绑定方法，避免了SQL注入风险。

另外，在用户体验的优化上，我发现菜单结构复杂且用户操作不够直观。通过重新设计菜单布局，统一操作逻辑，并结合慢速打印和清屏效果，提升了系统的交互性和用户友好性。每次操作

后，系统都会提示用户继续或返回上一级，避免了因不清楚当前状态而导致的操作失误。

## 收获与总结

通过本次宿舍管理系统的开发，我在多个方面取得了显著的进步。首先，面向对象编程的理论知识得到了实际应用，通过设计和实现多个类，深入理解了封装、继承和多态的概念。其次，数据库操作能力得到了极大提升，通过与SQLite3的集成，掌握了数据库的基本操作和高级查询技巧，确保了数据的安全性和一致性。

在项目设计与模块化方面，我学会了如何将复杂系统拆分为多个独立模块，各模块之间通过清晰的接口进行交互，提高了系统的可维护性和扩展性。同时，解决开发过程中遇到的问题，如输入验证、SQL注入防范等，锻炼了我的问题分析与解决能力。

此外，用户体验的优化让我认识到细节设计的重要性，通过优化界面布局和操作流程，提升了系统的易用性和用户满意度。团队协作与项目管理的经验也在本次开发中有所体现，学会了如何有效地分工合作，确保项目按时完成。

总的来说，本宿舍管理系统的开发不仅让我在技术层面得到了全面提升，也培养了我在实际项目中进行需求分析、系统设计、编码实现和问题解决的综合能力。未来，我将继续深化这些技能，并探索更多先进的技术，以进一步提升软件开发的水平。

详细报告见后页：

# 01 主函数开发

## 1. 概述

本项目是一个基于命令行的宿舍管理系统，旨在通过简单的交互方式，提供学生和管理员分别管理宿舍、用户信息、维修请求等功能。系统主要分为学生和管理员两个角色，分别具有不同的菜单和操作权限。通过一系列功能模块，学生可以查看宿舍信息、申请调整住宿、提交维修请求等；管理员则可以管理宿舍楼、用户、维修请求等。

---

## 2. 项目功能描述

### 2.1 主程序与菜单设计

主程序的入口是一个主菜单，用户可以选择进入学生菜单、管理员菜单，或退出系统。通过 `getChoice` 函数获取用户输入，并根据选择执行相应的操作。输入检查通过循环实现，确保用户输入合法（如范围检查、类型检查等）。

- **主菜单**：提供三项基本功能选择：
  - 学生菜单
  - 管理员菜单
  - 退出系统

### 2.2 学生登录与功能模块

学生登录模块包括学生身份验证、密码检查、密码修改等功能。系统首先要求学生输入学号和密码，并通过 `userManager->loginUser` 验证身份。若登录成功，学生将进入个人功能菜单。

- **学生菜单**：
  - 查看宿舍楼和房间信息
  - 申请住宿调整（如入住、退宿、换宿）
  - 提交维修请求
  - 查看系统通知和请求
  - 修改密码
  - 退出登录

在学生登录时，若检测到密码为默认密码（例如学号），系统会提醒学生修改密码。

### 2.3 管理员登录与功能模块

管理员功能模块与学生功能模块有所不同。管理员需要输入管理员账号和密码进行身份验证。管理员登录后，可以执行更复杂的操作，如管理宿舍楼、用户信息、维修请求、住宿安排等。

- **管理员菜单**：
  - 管理宿舍楼信息
  - 用户管理（添加、删除、查看用户）
  - 生成报表（如宿舍使用情况、入住率等）
  - 维修处理中心
  - 住宿管理中心
  - 退出登录

管理员通过各种子菜单来执行任务，包括用户管理、宿舍楼管理、住宿安排等。

### 2.4 住宿调整功能

宿舍调整模块允许学生申请入住、退宿、换宿等操作。每个操作会触发相应的后续处理逻辑，如检查是否有待审批的请求，确认申请是否符合条件等。

### 2.5 管理员权限功能

管理员可以：

- **宿舍管理**：添加、删除宿舍楼，查看和修改宿舍楼信息。
- **用户管理**：通过学号或姓名查看、添加、删除用户，修改用户密码。
- **报表生成**：生成宿舍使用情况报表和入住率报表。

---

## 3. 关键功能实现

### 3.1 输入验证与错误处理

在各个菜单选项中，输入处理是非常关键的部分。通过 `getChoice` 函数进行多次输入验证，确保用户输入的选项在有效范围内，并能够处理错误输入（如输入非数字字符时清空输入缓冲区）。

遇到输入错误时，系统会反复提示用户进行正确输入，保证操作的稳定性和用户体验。

### 3.2 密码安全

在学生和管理员登录时，密码输入采用隐藏模式，用户输入时密码不会显示在屏幕上。系统还支持输入`*`来查看密码，确保用户在登录过程中能够保密密码。

### 3.3 宿舍管理与申请调整

在学生菜单中，学生可以通过“申请住宿调整”功能提交入住、退宿或换宿的请求。如果存在待审批的请求，系统会阻止学生重复提交申请，避免无效的操作。

管理员拥有权限审批学生的住宿申请并进行相应的住宿安排。通过`dormManageMenu`，管理员可以安排住宿、处理学生申请等。

### 3.4 用户与宿舍楼管理

管理员可以通过`manageUsers`菜单管理用户，包括添加、删除用户、查询用户信息等；同时，管理员还可以通过`manageDormitories`菜单进行宿舍楼的管理，添加、删除宿舍楼，或查看宿舍楼的信息。

---

## 4. 开发过程中的问题与解决方案

### 4.1 输入合法性校验

在系统的初步开发过程中，出现了输入时未能正确校验的情况。例如，在获取用户菜单选择时，若用户输入了非法字符，系统会直接崩溃。为此，开发过程中添加了大量的输入校验代码，通过`cin.fail()`清除输入缓冲区，确保程序不会因错误输入而异常退出。

### 4.2 密码保护与修改（陈奇乐完成）

在实现密码输入和修改时，遇到了密码显示的问题。最初密码的输入没有加密处理，导致密码显示在终端上。解决方案是通过`clearScreen`和`passwordHide`等方法隐藏用户输入的密码，并在必要时进行密码修改提醒。

---

## 5. 收获与反思

通过本次宿舍管理系统的开发，深刻认识到了良好的用户体验与输入验证的重要性。特别是在输入部分，通过精心设计输入框架与错误处理机制，极大地提升了系统的稳定性。同时，项目中也加深了对C++中面向对象编程的理解，掌握了如何利用类和对象进行系统功能的模块化。

# 02 数据库模块函数开发

## 数据库模块函数开发

### 项目背景与需求

在本项目中，我们使用 SQLite3 数据库与 C++ 相结合，完成宿舍管理系统的数据库模块。该模块负责处理系统中的数据存储、查询、更新等任务，包括对宿舍信息、用户信息、房间使用情况等数据的管理。本部分报告主要关注于 `Database` 类的实现，该类提供了与数据库的接口，并确保与宿舍管理系统的其他模块之间的数据交互。

### 项目目标

1. **数据库连接与管理**：确保能够正确连接数据库并执行查询与更新操作。
2. **查询与执行操作**：支持多种形式的 SQL 操作，包括非查询操作（如插入、更新）和查询操作（带参数查询、存在性检查等）。
3. **事务与数据一致性**：确保操作完成后，数据库数据的一致性与完整性。

---

## 数据库模块结构与实现

### 1. `Database` 类设计

`Database` 类是与 SQLite 数据库交互的核心类。它通过 `sqlite3` 库提供的接口，负责建立和关闭数据库连接，以及执行各种数据库操作。该类包含以下重要成员和方法：

- **构造函数**：`explicit Database(const string &dbPath)`
  用于打开数据库连接，如果数据库连接失败则输出错误信息。
- **析构函数**：`~Database()`
  用于关闭数据库连接并释放资源。
- `execute` **方法**：
  执行非查询 SQL 语句（如 `INSERT`、`UPDATE`、`DELETE`）。如果数据库连接失败，方法会返回 `false` 并输出错误信息。
- `query` **方法**：
  执行查询 SQL 语句并打印结果。该方法通过 `sqlite3_step` 执行查询，并通过 `sqlite3_column_text` 获取每一列的数据。
- `queryExists` **方法**：
  用于执行存在性检查的查询，判断某个条件是否符合，返回布尔值结果。

- `queryWithParams` **方法**：
  用于执行带有参数的查询。通过 `sqlite3_bind_text` 绑定查询参数，确保 SQL 注入防范。
- `executeWithParams` **方法**：
  用于执行带参数的非查询操作。与 `queryWithParams` 方法相似，通过绑定参数执行操作，适用于 `INSERT`、`UPDATE` 等语句。
- `updateRoomStatus` **方法**：
  更新宿舍房间的占用状态。通过 SQL 子查询统计 `student_rooms` 表中的学生数量，并更新 `rooms` 表中的 `occupied` 字段。
- `getQueryResult` **方法**：
  获取查询结果中的某一列值，通常用于获取查询的结果。

## 2. 数据库操作流程

- **打开数据库**：
  在 `Database` 类的构造函数中，通过 `sqlite3_open` 打开 SQLite 数据库。如果连接失败，会输出错误信息。
- **执行 SQL 语句**：
  在执行 SQL 时，使用 `sqlite3_exec` 处理非查询类型的 SQL 语句，如插入、删除、更新等。如果是查询类型的操作（如 `SELECT`），使用 `sqlite3_prepare_v2` 编译 SQL 语句，并通过 `sqlite3_step` 执行。查询过程中，根据返回结果逐行处理数据。
- **参数绑定**：
  对于带有动态参数的 SQL 语句，使用 `sqlite3_bind_text` 将参数绑定到 SQL 语句中，避免 SQL 注入。

## 3. 数据库查询与更新

- **查询操作**：
  通过 `query` 方法，查询结果会以列为单位进行打印。每次查询返回的数据会显示在控制台，方便管理员查看。
- **带参数查询**：
  使用 `queryWithParams` 方法时，我们将 SQL 查询中的参数进行绑定。这样能确保用户输入的数据安全，并防止恶意 SQL 注入。
- **更新房间状态**：
  通过 `updateRoomStatus` 方法，系统能自动更新每个房间的占用情况。该方法通过查询 `student_rooms` 表中每个房间的学生人数，并更新 `rooms` 表中的对应字段。

## 4. 错误处理

数据库操作过程中，遇到的错误通过输出错误信息来提示用户。例如，如果 SQL 执行失败，`sqlite3_errmsg` 会返回详细的错误信息，以便开发人员定位问题。

# 遇到的问题与解决方案

## 1. 数据库连接失败

在最初的开发阶段，我们遇到数据库无法打开的错误。通过检查路径和权限，发现由于数据库文件路径错误导致无法正确连接数据库。最终，我们改进了数据库路径的管理，确保路径是动态计算的，并通过日志记录了数据库连接的详细错误信息。

## 2. SQL 查询语法错误

在多次调试过程中，执行的 SQL 查询语句可能存在语法错误，导致查询结果为空或返回错误。解决此问题时，我们采用了逐步检查 SQL 语句的方式，通过输出调试信息来验证每个查询的正确性。

## 3. 参数绑定的问题

在处理带参数的 SQL 查询时，初期出现了绑定参数失败的情况。通过深入阅读 SQLite3 的文档，我们解决了使用 `sqlite3_bind_text` 绑定字符串类型参数的问题，并确保每个绑定的参数都能正确映射到 SQL 语句中。

---

# 收获与反思

1. **SQL 注入防范**：
   在处理数据库时，确保 SQL 注入防范是至关重要的。在项目开发中，我们通过绑定参数（而不是直接拼接字符串）来有效防止 SQL 注入攻击，从而提高了系统的安全性。
2. **调试技巧**：
   在调试过程中，我们学会了如何通过 `sqlite3_errmsg` 获取详细的错误信息，帮助快速定位问题。对于复杂查询，逐步拆解查询并打印结果是高效的调试方式。
3. **数据库优化**：
   在更新房间状态时，使用了 `UPDATE` 和 `SELECT` 语句的结合。这个操作需要对数据库进行一定的优化，例如使用子查询来避免多次查询，确保系统性能。
4. **项目整体设计**：
   通过设计一个数据库模块，我们将与数据库交互的操作进行了封装，降低了其他模块与数据库的耦合度。这个设计使得数据库操作的逻辑更加清晰，易于维护和扩展。

---

# 总结

本数据库模块的实现确保了宿舍管理系统能够高效、稳定地与数据库交互。通过对查询、执行等操作的封装，我们提高了代码的可读性和可维护性。在开发过程中，我们不仅掌握了 SQLite 的使用，还学会了如何高效地处理带参数的 SQL 查询和避免 SQL 注入等常见安全问题。

# UserManager模块开发报告

## UM01 通用功能

### 1. 项目概述

在本项目中，`UserManager` 类负责管理宿舍管理系统中的所有用户（如学生、管理员等）的相关操作，提供了一些基础的用户管理功能。为了确保系统的灵活性和扩展性，通用功能模块被设计为独立的功能区块，负责处理与数据库交互、验证用户身份、查询数据等常见操作。本章节将详细介绍 `UserManager` 类中的通用功能，包括如何与数据库进行交互、用户操作的验证、以及如何保障用户输入的有效性等。

### 2. 主要功能及实现步骤

#### 2.1 数据库查询与执行

- **查询并打印（`query`）**
  该方法用于执行数据库查询，并直接在控制台输出结果。通过传递SQL查询语句，它与数据库交互并显示查询结果。
  **实现步骤**：
  - 输入SQL查询语句
  - 调用 `db.query(SQL)` 来执行查询
  - 返回并打印查询结果
- **执行（`execute`）**
  执行不需要返回结果的数据库操作（如插入、更新、删除）。
  **实现步骤**：
  - 输入SQL操作语句
  - 调用 `db.execute(SQL)` 来执行该操作
- **查询但不打印（`queryExists`）**
  该方法用于执行查询操作，但不输出结果，仅用于验证某条记录是否存在。
  **实现步骤**：
  - 输入SQL查询语句
  - 使用 `db.queryExists(SQL)` 判断是否存在相关数据
  - 返回布尔值结果（`true` 或 `false`）

#### 2.2 用户验证与注册

- **一键获取所有用户信息 (`getAllUsers`)**
  查询并显示所有用户的信息。
  **实现步骤**:
  - 使用 `db.query("SELECT * FROM users;")` 查询所有用户信息
  - 在控制台中打印所有查询结果
- **获取并返回ID (`get_ID`)**
  该方法用于获取学生的学号，首先进行学号输入，并验证学号是否存在。
  **实现步骤**:
  - 循环要求用户输入学号
  - 检查该学号是否存在（通过调用 `queryExists`）
  - 如果学号存在且有效，返回该学号
- **用户ID存在性检查 (`IDExists`)**
  该方法用于检查指定的学生ID是否存在于数据库中。
  **实现步骤**:
  - 使用SQL查询验证学生ID是否存在
  - 返回布尔值，表示ID是否有效
- **用户注册 (`registerUser`)**
  用于注册新的用户（学生或管理员），该方法会将用户的基本信息（如学号、姓名、性别等）插入到数据库中。
  **实现步骤**:
  - 接受用户输入的学号、密码、姓名、性别等信息
  - 使用SQL插入语句，将用户信息插入到 `users` 表
  - 执行插入操作，并根据结果反馈注册成功或失败
- **修改密码 (`userPasswordChange`)**
  该方法允许用户修改其密码。
  **实现步骤**:
  - 提示用户输入新密码
  - 更新数据库中对应用户的密码字段
  - 根据执行结果输出修改成功或失败的提示

## 2.3 宿舍管理与查询

- **宿舍是否存在 (`dormitoryExistsByName`)**
  该方法用于检查指定的宿舍名称是否已经存在于数据库中。
  **实现步骤**:
  - 输入宿舍名称
  - 使用SQL查询检查宿舍楼是否存在
  - 返回布尔值，表示该宿舍楼是否已存在

- **获取宿舍ID（`getDormitoryIDByName`）**
  用于根据宿舍名称获取宿舍的唯一ID。
  **实现步骤**：
  - 根据宿舍名称查询对应的宿舍ID
  - 如果宿舍存在，返回宿舍ID，否则返回失败信息

## 2.4 学生入住与选择宿舍

- **是否入住（`isStudentCheckedIn`）**
  该方法用于检查某个学生是否已经入住宿舍。
  **实现步骤**：
  - 输入学生ID
  - 查询该学生是否已在 `users` 表中标记为已入住
  - 返回布尔值，表示学生是否已入住
- **选择有效的宿舍（`selectValidRoom`）**
  用于学生选择合适的宿舍楼及房间。
  **实现步骤**：
  - 首先查询所有符合性别要求并且有空房间的宿舍楼
  - 提示用户输入宿舍楼名称，验证该名称是否合法并且有空房
  - 根据宿舍楼名称查询空房间，并允许学生选择房间号
  - 验证房间号的有效性，确保该房间为空且符合要求

## 2.5 用户登录

- **用户登录（`loginUser`）**
  用于用户登录验证，学生或管理员通过输入学号和密码进行登录。
  **实现步骤**：
  - 输入用户ID和密码
  - 查询数据库验证用户的ID、密码和身份（管理员或普通用户）
  - 如果验证成功，返回登录成功信息；否则返回失败提示

# 3. 遇到的问题与解决方案

## 3.1 问题：用户输入的验证与错误处理

在实现学生注册、登录、修改密码等功能时，涉及用户输入数据的验证（如学号是否已存在、密码格式是否正确等）。
**解决方案**：

- 为每个功能添加了循环和条件判断，确保用户输入的数据有效。

- 通过数据库查询操作验证学号、用户名等是否已存在，防止重复注册。

## 3.2 问题：数据库查询的性能与异常处理

在处理查询时，由于需要对数据库进行多次查询，可能会影响系统的性能，尤其在大量数据操作时可能会出现延迟或错误。
**解决方案**：

- 优化了SQL查询语句，避免多余的查询和重复操作。
- 加强了异常处理机制，避免数据库查询失败时程序崩溃。

## 3.3 问题：用户选择宿舍时的数据一致性

在宿舍选择功能中，学生可以选择宿舍楼及房间。然而，如果多个学生同时选择相同的房间，可能会造成数据不一致。
**解决方案**：

- 在选择宿舍时，增加了房间的实时查询，确保学生选择的房间在其操作时是空闲的。
- 使用了锁机制保证房间选择的同步性。

# 4. 收获总结

- **数据库操作的熟练度提高**：通过频繁的数据库查询、插入和更新操作，增强了对SQL语句的掌握和优化能力。
- **用户体验的优化**：通过增强错误处理、输入验证等措施，提升了用户操作的流畅性和系统稳定性。
- **系统的扩展性设计**：在开发过程中，考虑了未来可能需要扩展的功能，尤其是在用户管理和宿舍管理的设计上，确保了系统具有较好的扩展性。

通过这部分功能的开发，系统的基础功能得到了完善，提供了一个稳定的框架，支持学生和管理员的日常操作，同时也为后续功能的扩展提供了良好的基础。

# UM02 宿舍管理功能

在宿舍管理功能模块中，主要包含了宿舍楼的增、删、查、改等操作。该模块的目的是为了使管理员能够方便地管理宿舍楼及其下属房间的信息。具体实现的功能包括宿舍楼的添加、删除、查看以及信息的修改。

## 1. 添加宿舍楼

添加宿舍楼的功能首先要求管理员输入宿舍楼的名称、性别和位置等基础信息。在输入过程中，通过不断检查输入的有效性，确保宿舍楼名称的唯一性和性别的合法性。对于每个新增的宿舍楼，还需要自动生成并插入若干房间，房间的数量和床位数由管理员指定。

**步骤：**

1. 输入宿舍楼名称，并检查该名称是否已存在。如果宿舍楼名称已存在，则要求管理员重新输入。
2. 输入宿舍楼性别，且必须选择“男”或“女”，否则要求重新输入。
3. 输入宿舍楼位置，提供位置描述。
4. 通过SQL插入语句将宿舍楼的信息写入数据库。
5. 获取新插入宿舍楼的ID，并输入房间数量和每个房间的床位数。
6. 自动生成房间号，并将房间信息插入数据库。

**遇到的问题与解决：**

- 问题：如果输入的宿舍楼名称已经存在，系统无法继续进行后续操作。
- 解决：在输入宿舍楼名称时，增加对名称是否已存在的检查，若存在，则提示重新输入。

**收获：**

- 学会了如何在数据库中管理多表数据，尤其是如何通过宿舍楼ID来关联并自动生成房间数据。

## 2. 删除宿舍楼

删除宿舍楼时，首先会列出所有宿舍楼的信息供管理员选择。管理员输入目标宿舍楼名称后，系统会检查该宿舍楼是否存在。如果存在，则继续进行删除操作，否则提示管理员该宿舍楼不存在。在删除宿舍楼前，还需要检查该宿舍楼下是否有房间以及房间是否已入住。如果有学生入住，系统会阻止删除操作。

**步骤：**

1. 显示所有宿舍楼信息供管理员选择删除。
2. 输入待删除的宿舍楼名称，检查其是否存在。
3. 获取该宿舍楼下所有房间的ID，并检查每个房间的入住情况。
4. 如果房间已经被学生入住，则提示管理员处理学生的退宿事宜。
5. 如果该宿舍楼下无房间或所有房间为空房，则允许删除。
6. 确认删除操作，删除该宿舍楼及其相关房间。

**遇到的问题与解决：**

- 问题：删除宿舍楼时需要判断是否有学生入住，如果有学生入住，不能直接删除。
- 解决：遍历宿舍楼下所有房间，并查询每个房间是否有学生入住，只有在所有房间空置时才能进行删除操作。

**收获：**

- 掌握了如何处理带有外键关系的删除操作，并能够通过多表联动检查数据是否符合删除条件。
- 深入理解了如何设计删除操作时的逻辑，确保数据一致性和完整性。

## 3. 查看所有宿舍楼信息

查看所有宿舍楼信息的功能是为了让管理员了解各宿舍楼的基本情况，包括宿舍楼的名称、性别、位置、房间数量、总容量、已入住人数以及占用率等。这些信息通过SQL查询汇总并打印出来。

**步骤：**

1. 使用SQL查询语句获取所有宿舍楼的基本信息及其下属房间的数据。
2. 汇总信息并按宿舍楼分组，计算每个宿舍楼的房间数、总容量、已入住人数以及占用比率。
3. 打印查询结果，展示给管理员。

**遇到的问题与解决：**

- 问题：如何汇总每个宿舍楼的房间数量和入住情况。
- 解决：使用SQL的聚合函数，如COUNT、SUM等来统计每个宿舍楼的房间数量、总容量及已入住人数，并计算占用比率。

**收获：**

- 通过这个功能，加深了对SQL查询的理解，尤其是如何使用JOIN和GROUP BY来汇总和统计信息。

## 4. 修改宿舍信息

修改宿舍信息的功能允许管理员对已有宿舍楼的名称和位置进行修改。管理员可以选择要修改的宿舍楼，并根据需要修改其名称或位置。

**步骤：**

1. 显示所有宿舍楼的信息。
2. 输入要修改的宿舍楼名称，并检查该宿舍楼是否存在。
3. 提供修改内容的选择（宿舍楼名称或位置）。
4. 根据管理员的选择，更新数据库中的宿舍楼信息。

**遇到的问题与解决：**

- 问题：如何确保管理员修改的内容是有效的，并且操作简便。
- 解决：提供清晰的操作选项，并通过循环和条件判断确保管理员输入的内容符合预期。

**收获：**

- 深刻理解了如何通过SQL语句进行条件更新，以及如何设计用户交互流程以确保操作的顺利进行。

# 总结

宿舍管理功能模块的实现，使得管理员可以方便地管理宿舍楼及其房间的信息。通过这一模块的开发，我掌握了数据库操作的多种方式，包括数据的插入、删除、查询和更新。同时，解决了在管理过程中可能遇到的多个问题，例如数据一致性、外键关系的处理等。

# UM03 房间管理功能

## 1. 功能概述

在宿舍管理系统中，房间管理功能是管理员管理宿舍楼中各个房间及其入住学生信息的核心部分。该功能主要包括查看房间信息、删除房间、修改房间信息等操作。通过这一系列功能，管理员能够方便地管理宿舍楼中的每个房间及其住户情况。

## 2. 房间管理功能的具体实现

### 2.1 查看选定宿舍楼的所有房间信息和入住学生

该功能允许管理员查看指定宿舍楼内所有房间的信息，包括房间号、容量以及已入住人数。同时，还会显示入住在这些房间中的学生姓名和学号。

- **功能流程**：
  - 管理员首先输入宿舍楼名称。
  - 系统根据宿舍楼名称查询该楼的ID，进而查询该楼下所有房间的信息。
  - 如果查询到房间信息，系统将列出每个房间的房间号、容量以及已入住人数。
  - 如果有学生入住，系统会进一步列出该房间内所有学生的姓名和学号。
- **关键问题与解决**：
  - **问题**：如何在展示房间信息的同时，查询并展示房间内所有入住学生的信息？
  - **解决方案**：通过多表连接查询，结合房间ID和学生信息表，实现了房间与学生的关联查询，确保每个房间的入住学生信息可以一并展示。
- **收获**：
  - 通过实现多表连接查询，加深了对SQL JOIN操作的理解，提高了数据库查询优化的能力。

### 2.2 房间管理菜单

管理员通过房间管理菜单对选定的房间进行进一步的操作。菜单提供了多个选项，如查看房间入住信息、删除房间、修改房间信息等。

- **功能流程**：
  - 管理员选择一个宿舍楼，并进一步选择一个具体房间。
  - 系统展示该房间的基本信息，包括房间号、容量和已入住人数。
  - 管理员可以选择进入房间管理菜单，进行房间信息的查看、删除或修改操作。
  - 对于已经入住学生的房间，删除操作会受到限制，管理员需要处理学生退宿问题。
- **关键问题与解决**：

    - **问题**：如何处理已入住学生的房间删除操作？
    - **解决方案**：在删除房间前，系统首先检查房间的入住状态，若有学生入住，则提示管理员处理学生的退宿。这样避免了因学生未退宿而造成的删除操作失败。
- **收获**：
    - 深入思考了系统操作中的业务逻辑，特别是处理与学生入住状态相关的操作，确保系统功能的合理性和安全性。

## 2.3 查看入住信息

管理员可以查看某个房间的所有入住学生信息，包括学生的姓名和学号。该功能使得管理员能够实时了解每个房间的入住情况。

- **功能流程**：
    - 管理员在选择房间后，可以通过“查看入住信息”选项查看该房间内的所有学生。
    - 系统通过查询与房间ID和学生信息相关的数据库表，列出所有在该房间内的学生。
- **关键问题与解决**：
    - **问题**：如何保证在查询入住学生时，不遗漏任何一个入住的学生？
    - **解决方案**：使用多表查询，并通过JOIN操作连接学生、房间和宿舍楼信息，确保查询结果的完整性。
- **收获**：
    - 学会了如何有效地进行多表联合查询，以便获取更多关联信息，增强了处理复杂查询的能力。

## 2.4 删除房间

删除房间功能允许管理员删除不再使用或需要重组的房间。但在删除房间时需要考虑房间的入住情况。

- **功能流程**：
    - 管理员选择删除房间时，系统首先检查该房间是否有学生入住。
    - 如果房间内有学生，系统提示管理员处理学生退宿。
    - 如果房间内没有学生，系统执行删除操作，移除该房间的所有记录。
- **关键问题与解决**：
    - **问题**：删除房间时，如何确保学生的入住记录先行处理？
    - **解决方案**：在删除房间之前，系统会检查该房间的入住人数，如果有学生入住，则首先显示这些学生的详细信息并要求管理员处理退宿后再进行删除操作。
- **收获**：
    - 在实现删除操作时，增加了业务逻辑判断，保证删除操作符合实际业务流程，避免了误操作。

### 2.5 修改房间信息

管理员可以修改房间的各项基本信息，包括房间号、容量和维修状态。

- **功能流程**：
  - 管理员选择修改房间信息后，系统会提供修改选项，包括房间号、房间容量和房间状态（例如“正常”或“维修中”）。
  - 根据管理员的选择，系统会更新对应的房间信息。
- **关键问题与解决**：
  - **问题**：如何确保房间容量修改后不小于现有入住人数？
  - **解决方案**：系统对修改房间容量做了限制，确保新容量大于现有的入住人数，避免出现容量不足的情况。
- **收获**：
  - 在设计修改功能时，考虑了数据一致性和合理性，确保了房间容量修改的有效性和可操作性。

## 3. 总结与收获

通过实现房间管理功能，我们对数据库操作、用户交互和业务逻辑有了更深入的理解。在开发过程中，我们不仅实现了基本的增删改查操作，还特别考虑到了业务流程中的细节问题，比如房间删除前的学生退宿处理，房间容量的修改限制等。通过这些实现，我们提高了对系统设计的综合能力，学会了如何设计合理的用户操作流程，以及如何确保系统在不同情况下都能稳定运行。

# UM04 用户管理功能

在宿舍管理系统中，用户管理功能主要包括添加用户、删除用户、查询用户信息等操作。这个功能旨在帮助宿舍管理员有效地管理系统中的用户，确保用户的账户信息、入住情况以及联系方式等数据的准确性和完整性。以下是实现过程的详细描述。

## 1. 添加用户

用户的添加功能是系统中最基本的操作之一，管理员通过此功能可以新增学生用户，并为其分配必要的账户信息。具体步骤如下：

- **输入用户数量**：管理员首先输入需要添加的用户数量。若输入 `0` 则返回上一级。
- **学号验证**：每添加一位用户，系统首先要求输入学号。如果该学号已经存在于系统中，系统会提示用户重新输入，直到学号唯一为止。
- **生成密码**：若学号长度小于等于6位，默认将学号作为密码；若学号长度大于6位，则使用学号后六位作为默认密码。
- **输入基本信息**：管理员需要输入用户的姓名、性别和联系方式。性别需为“男”或“女”，如输入无效则会继续提示。
- **用户注册**：系统通过调用注册用户的接口，将用户信息保存到数据库中。

这个功能确保了用户信息的完整性，并且通过学号唯一性和自动生成密码，减少了人为错误。

## 2. 删除用户

删除用户的功能主要用于移除系统中不再需要的用户。操作步骤如下：

- **检查入住情况**：删除前，系统会首先检查该用户是否已入住。如果该用户已入住，则会提示管理员先处理退宿事宜。
- **确认删除**：系统会显示该用户的基本信息并要求管理员确认是否删除。如果管理员选择“yes”，则继续删除操作；若选择“no”，则取消操作。
- **执行删除**：确认删除后，系统会执行数据库中的删除操作，将该用户从系统中移除。

在这个过程中，删除操作会被严格要求确认，以防止误删操作。系统还确保了删除的用户没有未处理的入住信息，避免删除过程中出现数据不一致的情况。

## 3. 查询用户信息

查询用户信息功能是用户管理模块的核心之一，管理员可以通过不同的方式查询系统中的用户信息：

- **通过学号查询**：管理员可以输入学号来查询单个用户的信息。如果用户未入住，则显示用户的基本信息；如果用户已入住，则同时显示其所在宿舍楼和房间号。
- **通过姓名查询**：此功能支持模糊查询，通过姓名的一部分即可查询到相关用户。如果查询结果为空，系统会提示用户未找到相关信息。
- **查询所有用户信息**：管理员可以选择查看系统中所有用户的信息。该功能将展示每个用户的学号、姓名、性别、联系方式以及是否已分配宿舍信息，便于管理员快速浏览所有用户的基本情况。

该功能模块的设计使得管理员可以灵活查询用户信息，满足不同的查询需求，并且能根据查询结果对用户进行后续的管理操作。

## 4. 遇到的问题与解决方案

在开发用户管理功能时，我们遇到了一些技术问题，主要包括以下几个方面：

- **学号重复问题**：用户注册时，学号的唯一性非常重要。在最初的设计中，未能充分检查学号是否重复，导致用户添加时容易出现冲突。为了解决这一问题，我们在添加用户时增加了学号的唯一性验证，确保每个学号只能注册一次。
- **删除用户时的退宿问题**：在删除已入住的用户时，我们发现如果没有先处理退宿问题，删除操作可能导致数据的不一致。为此，我们设计了退宿的预处理步骤，在删除用户前检查其入住状态，确保在删除前妥善处理其宿舍信息。
- **查询功能的性能问题**：随着系统中用户数量的增加，查询所有用户信息的功能可能会导致性能问题。为此，我们在查询时使用了适当的索引，并确保查询条件明确，避免全表扫描，以提高查询效率。

## 5. 收获与总结

在开发过程中，我们不仅解决了学号唯一性验证、删除用户时的退宿问题，还通过多次优化查询和删除操作的效率，使得系统在实际运行时更加流畅。通过实现这一模块，我们更加深入地理解了数据库管理中的一些关键概念，如事务处理、数据一致性及索引优化等。

# UM05 住宿管理功能

## 功能描述

在宿舍管理系统中，住宿管理功能涵盖了学生的宿舍安排与退宿管理，包括宿舍的选择、入住与退宿的操作、处理住宿申请等。以下将对各个主要功能模块进行详细描述。

### 1. 安排住宿

安排住宿功能主要包括通过选择宿舍楼与房间号，为学生安排住宿。这一过程首先通过系统提供的接口进行宿舍与房间选择，然后插入相关记录到数据库中。

#### 关键步骤：

- **选择有效的房间**：系统根据学生的学号选择可用的房间，并且通过 `selectValidRoom` 方法确保选择的房间尚未满员。
- **输入备注**：系统允许管理员在入住过程中输入可选的备注信息。
- **入住记录插入**：入住记录被插入到 `check_in_out_records` 表中，以便追踪学生的住宿历史。
- **更新学生入住状态**：通过设置学生的 `isCheckedIn` 属性为1，标记学生已经入住。
- **房间状态更新**：调用 `db.updateRoomStatus` 方法更新房间的状态，以反映当前的入住情况。

### 2. 安排退宿

退宿操作涉及到学生从宿舍中退房，并更新系统中的相关记录。管理员首先确认是否继续退宿操作，然后输入退宿备注信息，最后系统通过一键退宿功能完成整个退宿流程。

#### 关键步骤：

- **确认退宿操作**：管理员输入确认信息 `yes` 或 `no` 以决定是否继续退宿操作。
- **查询学生住宿信息**：系统查询学生当前的宿舍楼与房间号，并输出相应的住宿信息。
- **退宿记录插入**：退宿事件作为记录插入到 `check_in_out_records` 表中，确保每一次退宿都有详细的记录。
- **更新入住状态**：更新学生的 `isCheckedIn` 状态为0，标识该学生已退宿。
- **删除房间分配记录**：从 `student_rooms` 表中删除学生与房间的关系记录。
- **房间状态更新**：与安排住宿功能一样，系统会调用 `db.updateRoomStatus` 方法，确保房间状态得到及时更新。

## 3. 处理学生住宿申请

住宿申请处理功能用于管理学生的住宿请求，包括入住、换宿或退宿申请。管理员需要查看待审批的申请并决定是否批准每一项申请。

### 关键步骤：

- **查看待审批申请**：系统从数据库中查询所有待审批的住宿申请，并展示给管理员。
- **选择处理请求**：管理员选择需要处理的请求ID，并查看该请求的详细信息。
- **审批操作**：管理员可以选择通过申请、拒绝申请或返回。对于通过的申请，系统会自动完成相应的住宿安排或退宿操作。
- **更新申请状态**：每个处理过的申请都会更新审批状态（已审批或已拒绝），并记录审批时间。

## 4. 查看入住退宿记录

系统允许管理员查看所有学生的入住与退宿记录，以便了解学生的住宿历史。管理员可以选择查询某一学生的具体记录，或者查看所有学生的历史操作记录。

### 关键步骤：

- **查询所有记录或指定学生记录**：管理员可以选择查看所有学生的入住退宿记录，或者只查询特定学生的记录。
- **展示记录**：系统输出包括学生ID、房间ID、事件时间、记录类型等详细信息，帮助管理员做出合理的管理决策。

## 5. 查看未入住学生

通过 `checkUserNotCheckedIn` 功能，管理员能够查看所有未入住的学生，以便更好地进行宿舍安排。

### 关键步骤：

- **查询未入住学生**：系统查询所有没有入住宿舍的学生，展示包括学生学号、姓名、联系方式等信息。
- **展示结果**：显示未入住的学生信息，供管理员参考与后续处理。

# 遇到的问题与解决方案

## 问题 1：宿舍房间满员时无法安排住宿

在实现 `quickArrangeAccommodation` 方法时，系统会检查所选房间是否已满。若房间已满，系统将无法为学生安排住宿。为了避免这种情况，采用了SQL查询来判断房间是否有剩余空间，并在

插入记录之前确认房间是否能容纳更多学生。

**解决方案**：引入`r.occupied < r.capacity`条件，确保只有当房间未满时才允许安排入住。

## 问题 2：退宿操作过程中无法正确获取学生的宿舍信息

在`quickArrangeCheckOut`过程中，查询学生的宿舍信息时若数据库返回为空，说明未能正确匹配学生的宿舍与房间信息。

**解决方案**：通过修改SQL查询语句，确保学生的宿舍信息能够正确匹配，避免因为数据不一致而导致操作失败。

## 问题 3：处理住宿申请时存在重复审批

由于学生可能会重复提交相同的住宿申请，系统需要确保每个请求只被审批一次。

**解决方案**：使用`unordered_set`来存储申请ID，避免重复处理已经审批过的申请。每个有效的申请会在系统中标记为“已审批”或“已拒绝”，并删除处理过的申请。

# 收获

1. **数据库操作优化**：通过深入了解如何使用SQL语句管理学生的住宿信息，我提升了在数据库查询、更新和删除操作方面的技能。合理的SQL设计可以确保数据一致性和操作的高效性。
2. **系统设计思维**：在设计住宿管理功能时，我学会了如何拆解问题，逐步实现每个功能模块，并确保系统的可扩展性和可维护性。例如，将每项操作分解为“快速入住”、“快速退宿”等小模块，有利于后续的维护与升级。
3. **错误处理**：在处理学生住宿信息时，尤其是入住和退宿操作，我深刻意识到数据验证和错误处理的重要性。通过增强系统的异常处理机制，可以避免操作失败或数据丢失。

# UM06 报修处理与报表管理

## 报修处理功能

### 功能描述

报修处理功能旨在处理宿舍中学生提交的报修请求。管理员通过此功能能够查看所有未处理的维修请求，并根据请求内容决定是否处理该请求。处理后，系统会更新报修请求的状态，并在房间没有其他未处理报修的情况下，将房间状态更新为“正常”。该功能的核心在于确保维修请求得到及时的响应，同时合理更新宿舍房间状态。

### 具体步骤

1. **查询未处理的报修请求**
   首先，系统会查询数据库中的所有“未处理”状态的维修请求，并显示其详细信息，包括报修ID、学生ID、房间ID、报修种类、描述以及报修时间。这些信息供管理员参考，帮助其决定处理哪些报修请求。
2. **检查是否存在待处理报修请求**
   如果没有未处理的报修请求，系统会提示管理员当前没有待处理的报修请求，并直接退出处理流程。
3. **管理员选择报修请求**
   管理员通过输入报修ID来选择具体的报修请求。如果输入的ID无效或不存在，系统会进行提示并要求重新选择。
4. **确认处理报修请求**
   一旦管理员选择了有效的报修ID，系统会向管理员确认是否确认处理该报修请求。如果管理员选择“是”，系统将继续执行处理步骤，否则会取消操作。
5. **更新报修请求状态**
   如果管理员确认处理报修请求，系统将更新数据库中的报修请求状态为“已处理”，并记录处理时间。处理完成后，管理员可以继续进行其他操作。
6. **更新房间状态**
   根据报修请求的房间ID，系统会检查该房间是否还有其他未处理的维修请求。如果该房间还有待处理的报修请求，系统会将房间状态保持为“维修中”；否则，系统会将房间状态更新为“正常”。

## 遇到的问题与解决方案

- **问题一：查询未处理报修请求时出现结果为空**
  在查询未处理的维修请求时，可能会遇到没有待处理报修的情况。此时，系统应提示管理员

当前没有待处理请求。
**解决方案**：通过检查查询结果，如果返回值为空，则输出相应提示，避免系统误操作。

- **问题二：管理员输入无效的报修ID**
  如果管理员输入了无效的报修ID（例如不存在的ID或已处理的ID），系统需要验证ID的有效性。
  **解决方案**：通过SQL查询判断输入的报修ID是否存在且状态为“未处理”，若无效则提示管理员重新输入。
- **问题三：房间状态更新时的逻辑错误**
  如果房间仍有未处理的报修请求，房间状态不应直接更新为“正常”。
  **解决方案**：在更新房间状态之前，先通过查询检查该房间是否有其他未处理的维修请求。

## 从中获得的收获

- **数据验证的重要性**：在处理报修请求的过程中，确保输入数据和系统中的数据一致性非常重要。通过有效的查询和验证，可以避免因错误操作导致的系统异常。
- **逻辑思维的锻炼**：在开发过程中，如何处理不同状态之间的关系，尤其是在房间状态与报修请求之间的逻辑判断，增强了我的逻辑思维能力。
- **用户体验的提升**：通过设计简洁、清晰的操作流程，确保管理员能够快速、准确地完成任务，提高了系统的易用性。

# 报表管理功能

## 功能描述

报表管理功能主要用于生成宿舍的入住率报表。管理员可以根据时间范围（当前月或当前年）生成对应的入住率报表，帮助宿舍管理人员及时掌握宿舍的入住情况。该功能通过查询数据库，统计每个宿舍楼在特定时间范围内的入住人数，并计算入住率，最终以表格形式展示给管理员。

## 具体步骤

1. **选择时间范围**
   管理员可以选择生成报表的时间范围：当前月或当前年。系统会提供选择界面，并根据用户输入的选择生成相应的报表。
2. **生成当前月的入住率报表**
   如果选择当前月，系统会查询宿舍楼中所有房间在当前月份内的入住情况，统计每个宿舍楼的总房间容量，并计算入住率。入住率的计算方式为：已入住人数 / 总房间容量。
3. **生成当前年的入住率报表**
   如果选择当前年，系统会查询宿舍楼中所有房间在当前年度内的入住情况，统计每个宿舍楼的总房间容量，并计算入住率。此过程与当前月的报表类似，只是时间范围不同。
4. **查询并展示报表结果**
   生成SQL查询语句后，系统会执行查询操作并展示报表结果。管理员可以看到每个宿舍楼的

名称、已入住人数、总房间容量和入住率。报表结果会清晰地展示出来，供管理员分析和决策。

5. **操作提示**
   在报表生成过程中，系统会显示“正在查询中”的提示，查询完成后再显示报表结果。用户可以按任意键继续。

## 遇到的问题与解决方案

- **问题一：无效的时间范围选择**
  如果管理员输入了无效的时间范围（如不选择时间范围或选择错误的选项），系统需要进行提示并要求重新输入。
  **解决方案**：通过检查用户输入的时间范围是否有效，若无效，则提示重新输入。
- **问题二：查询结果为空**
  在某些情况下，查询结果可能为空（例如没有入住记录）。这时，系统应确保能够正常处理空结果，并提示管理员相应信息。
  **解决方案**：使用条件判断，确保查询结果为空时不会影响系统运行，且向管理员提供明确的反馈。
- **问题三：长时间查询操作**
  由于报表生成涉及大量数据查询，可能会出现长时间查询的情况。
  **解决方案**：通过在查询过程中添加进度提示（如“正在查询中”），提升用户体验，避免管理员误以为系统出现了问题。

## 从中获得的收获

- **数据统计与查询优化**：在处理报表生成时，通过优化SQL查询语句和合理的数据库索引，提升了查询效率，减少了长时间等待。
- **用户交互的设计**：通过提供简单明了的选择界面，使管理员能够轻松选择时间范围，避免因界面复杂而产生误操作。
- **动态提示与反馈机制**：在查询过程中加入进度提示，增强了系统的互动性和用户体验，让管理员在等待过程中感到更为流畅和舒适。

# 总结

通过实现报修处理和报表管理功能，**UserManager** 系统的功能得到了显著扩展。这两个功能不仅提升了管理员的工作效率，同时也增强了系统的交互性和可操作性。在开发过程中，我深刻体会到数据验证、逻辑设计、用户体验等多方面的重要性。通过合理的数据库查询和界面设计，能够使管理员高效、准确地处理任务，并进一步提高宿舍管理系统的综合水平。

# UM07 学生功能

# 学生功能开发报告

## 1. 功能概述

在宿舍管理系统的“学生功能”模块中，我们实现了多个关键功能，旨在为学生提供全面的宿舍管理服务，包括注册、入住申请、退宿申请、房间信息查询、维修请求、换宿请求等功能。该模块的设计目标是确保学生能够便捷地通过系统进行宿舍相关的各项操作，提升系统的易用性和学生的满意度。

## 2. 主要功能模块

### 2.1 学生注册功能

学生注册功能允许新生或未注册的学生在系统中创建账户。通过输入学号、密码、姓名、性别和联系方式，学生能够成功注册并获取系统账号。

- **步骤**：
  1. 学生提供个人信息（学号、密码、姓名、性别、联系方式）。
  2. 系统检查注册信息的有效性。
  3. 注册信息插入到数据库中。
  4. 注册成功后，系统会显示成功消息。
- **问题与解决**：
  - **问题**：系统需要保证学号的唯一性，以避免重复注册。
  - **解决方案**：在注册前进行学号的查询，确保数据库中没有重复的学号记录。
- **收获**：通过学生注册功能的实现，提升了对数据库操作的理解，尤其是在插入数据和确保数据唯一性方面。

### 2.2 查看宿舍楼和房间信息

学生可以查看自己宿舍楼和房间的详细信息，包括宿舍楼名称、房间号、房间容量、已入住人数及维修状态。只有已入住学生才可以查询这些信息。

- **步骤**：
  1. 验证学生是否已经入住。
  2. 如果已入住，查询数据库，获取该学生所在宿舍楼和房间的信息。
  3. 如果未入住，提供申请入住的选项。
- **问题与解决**：

    - **问题**：如何确保只有已入住的学生能够查看自己的宿舍信息。
    - **解决方案**：在数据库中增加“已入住”字段，并通过查询检查学生的入住状态，只有已入住的学生才能查看房间信息。
- **收获**：加深了对数据库连接操作和条件查询的理解，尤其是涉及到多表连接的查询操作。

## 2.3 申请入住

学生可以通过系统提交入住申请。系统会在确认学生未入住的情况下，向数据库中插入申请记录，并告知学生申请成功。

- **步骤**：
    1. 学生提出入住申请。
    2. 系统查询学生是否已经入住，如果未入住，插入入住申请记录。
    3. 提交成功后，系统告知学生申请已提交。
- **问题与解决**：
    - **问题**：避免学生重复申请入住。
    - **解决方案**：在申请之前，通过查询数据库检查学生是否已经入住。
- **收获**：通过此功能，了解了如何通过SQL语句插入记录并确保业务逻辑的一致性。

## 2.4 申请退宿

学生可以提交退宿申请，系统会验证学生是否已经入住，并根据入住情况处理退宿请求。

- **步骤**：
    1. 学生提出退宿申请。
    2. 系统查询学生的入住状态，如果学生未入住，则无法退宿。
    3. 如果已入住，系统插入退宿申请记录。
- **问题与解决**：
    - **问题**：确保学生只能在已入住的情况下提出退宿申请。
    - **解决方案**：查询数据库验证学生的入住状态，只有已入住的学生才可以退宿。
- **收获**：加强了对数据验证和状态管理的理解，确保了业务逻辑的正确性。

## 2.5 请求换宿

学生可以请求换宿，系统将学生选择的目标宿舍和房间信息插入申请记录中，并提交给宿舍管理部门处理。

- **步骤**：
    1. 学生选择目标宿舍和房间。
    2. 系统根据学生的选择生成换宿申请记录。
    3. 提交申请后，系统反馈成功消息。

- **问题与解决**：
  - **问题**：如何确保目标房间可用，避免重复选择已满的房间。
  - **解决方案**：在提交换宿请求前，查询目标房间的可用状态，确保该房间未满。
- **收获**：增强了数据库查询和插入的操作技巧，特别是在房间状态管理方面。

## 2.6 提交维修请求

学生可以提交维修请求，报告宿舍房间中的问题（如水电故障等），并选择具体的维修类型和问题描述。

- **步骤**：
  1. 学生提供房间信息、报修类型及描述。
  2. 系统生成维修请求并插入数据库。
  3. 系统更新房间的维修状态。
- **问题与解决**：
  - **问题**：如何确保学生只能报修自己所在房间的问题。
  - **解决方案**：通过查询学生的房间信息，确保报修请求的房间与学生的实际房间一致。
- **收获**：通过此功能的实现，提升了对用户输入验证和数据一致性的处理能力。

## 2.7 查看通知

学生可以查看自己的系统通知，了解与自己相关的消息（如审批状态变更等）。

- **步骤**：
  1. 学生查询通知。
  2. 系统展示通知列表或单条通知内容。
- **问题与解决**：
  - **问题**：如何管理和展示通知。
  - **解决方案**：创建通知记录表，并通过学生ID查询相关通知。
- **收获**：通过实现该功能，学习了如何管理通知数据，并增强了对数据表设计的理解。

## 2.8 查看所有请求

学生可以查看自己所有的申请记录（如入住申请、退宿申请等），包括申请类型、状态、宿舍楼、房间号等信息。

- **步骤**：
  1. 学生查询所有的申请记录。
  2. 系统按时间倒序展示所有相关记录。
- **问题与解决**：
  - **问题**：如何有效地展示申请记录并区分不同类型的申请。

    - **解决方案**：使用SQL查询按申请时间排序，并通过状态区分不同类型的请求。
- **收获**：通过此功能，掌握了如何处理学生申请的历史记录，并对数据库的查询和排序操作有了更深入的理解。

### 2.9 查看未审批的请求

学生可以查看自己有待审批的请求，便于及时跟进申请的审批状态。

- **步骤**：
    1. 学生查询待审批的申请记录。
    2. 系统返回所有状态为“待审批”的申请。
- **问题与解决**：
    - **问题**：如何准确筛选待审批的请求。
    - **解决方案**：在SQL查询中增加条件筛选，确保只返回待审批的申请记录。
- **收获**：通过此功能，进一步了解了如何管理和筛选数据库中的状态字段。

## 3. 总结与收获

在“学生功能”模块的开发过程中，我们深入了解了如何通过数据库操作实现学生的各类宿舍管理需求。通过注册、申请入住、申请退宿、维修请求等功能的实现，提升了我们对数据库查询、插入、更新等操作的掌握。此外，开发过程中还遇到了一些问题，如如何确保数据的一致性、如何避免重复申请等，但通过精心设计数据库结构和使用有效的查询条件，成功解决了这些问题。