X-Lab 项目报告

Written By 杨家铭 in Group "实验不与理论作队"

一、程序功能介绍

我们的项目名为X-Lab,旨在为身处实验室进行科研的老师和同学搭建一个类似工具“百宝箱”的平台，综合适配了多学科、多领域功能，集成多种科研、学习、工作中常用的工具，助力实验室科研同学的体验，提高效率。通过不断的制作和实现过程，我们丰富了其功能并进行了扩展，使其更加全面和实用。以下是X-Lab的主要功能介绍：

详细功能：

1. 注册/登录：运用小型sql数据库存储用户数据
2. 用户前端修改个人主页和相关资料，以及密码（通过邮箱验证确保安全性）
3. 界面跳转：设计了首页、欢迎页、实验室界面、api查阅文献等，通过采用stackWidget结构，利用栈的原理实现前端不同界面的顺畅显示和跳转
4. 云实验室：可以自由增删以及搭配、移动仪器
5. 扩展性与API调用：集成了Google Serp API，能够访问Google Scholar并解析返回的json文件，为科研人员提供便捷的文献检索功能。同时，支持LaTeX等扩展，方便在平台上进行科研论文的撰写
6. 用户管理：制作了管理员界面，能闭环管理整个平台

二、项目各模块与类设计细节

1. 协作与任务管理模块

- README 文档
  - **Git 合作方式**：项目目前使用 `main` 分支进行开发，每次开始修改代码前需要先执行 `git pull`。若在修改过程中远程 `main` 分支有更新，可通过 `git stash` 保存当前修改，`git pull` 拉取更新后再使用 `git stash apply` 恢复修改。
  - 任务安排：在协作过程中明确了各个成员接下来需要完成的任务，（可能最终未完全更新）例如：
    - `wyx`：完成忘记密码的功能，以及主页面的用户信息显示、修改、头像上传等功能。
    - `yjm`：完成左侧边栏中实验室的子栏、个人资料的弹窗，实现第三个鼠标点击后跳转界面查看会议等文献并调用查找文献的 API，同时记录了一些优化点，如侧边栏图标优化、实验室仪器参数和个数添加等。
    - `lhr`：完成实验室的基本功能，初步规划 `to-do-list` 功能。

2. 用户界面模块

以其中的项目卡片组件为例：

- `MainWindow::createProjectCard` 函数（411trial/mainwindow.cpp）
  - **功能**：创建一个项目卡片组件，用于在界面上展示项目的标题和描述信息。
  - 实现细节：

- 创建一个 `QWidget` 作为卡片容器，并设置其样式，包括背景颜色、边框半径、内边距等。
- 在卡片内部使用 `QVBoxLayout` 布局，添加标题标签和描述标签。
- 为卡片添加悬停阴影效果，增强视觉体验。

```
// 项目卡片组件
QWidget* MainWindow::createProjectCard(const QString &title, const QString &desc)
{
    QWidget *card = new QWidget;
    card->setStyleSheet(R"(
        QWidget {
            background: white;
            border-radius: 8px;
            padding: 16px;
            border: 1px solid #e1e4e8;
        }
        QLabel#title {
            font-size: 16px;
            font-weight: 600;
            color: #24292e;
        }
        QLabel#desc {
            color: #586069;
            font-size: 14px;
        }
    )");

    QVBoxLayout *layout = new QVBoxLayout(card);

    QLabel *titleLabel = new QLabel(title);
    titleLabel->setObjectName("title");

    QLabel *descLabel = new QLabel(desc);
    descLabel->setObjectName("desc");
    descLabel->setWordWrap(true);

    layout->addWidget(titleLabel);
    layout->addWidget(descLabel);
    layout->addStretch();

    // 添加悬停效果
    QGraphicsDropShadowEffect *shadow = new QGraphicsDropShadowEffect;
    shadow->setBlurRadius(10);
    shadow->setColor(QColor(0,0,0,15));
    shadow->setOffset(2, 2);
    card->setGraphicsEffect(shadow);

    return card;
}
```

3. 插件管理模块

插件数据结构

- `PluginData` 结构体（`411trial/pluginmanager.h`）
  - **功能**：用于存储插件的基本信息，包括名称、版本、描述、图标、安装状态和分类等。
  - **定义**：

```
// 插件数据结构
struct PluginData {
    QString name;
    QString version;
    QString description;
    QString icon;
    bool installed;
    QString category;
};
```

插件卡片组件

- `PluginCard` 类（`411trial/pluginmanager.h`）
  - **功能**：实现插件卡片的显示和交互功能，包括安装和打开插件的按钮。
  - 实现细节：
    - 继承自 `QFrame`，在构造函数中接收插件数据并调用 `setupUI` 函数进行界面设置。
    - 提供 `onInstallClicked` 和 `onOpenClicked` 槽函数，分别处理安装和打开插件的操作。
    - 发出 `pluginClicked` 信号，用于通知主窗口插件被点击。

主窗口类

- `PluginManager` 类（`411trial/pluginmanager.h`）
  - **功能**：作为插件管理的主窗口，负责插件的初始化、搜索过滤、分类显示等功能。
  - 实现细节：
    - 继承自 `QMainWindow`，在构造函数中调用 `setupUI` 函数进行界面初始化。
    - 提供 `initializePlugins` 函数用于初始化插件数据。
    - 提供 `filterPlugins` 函数用于根据搜索文本过滤插件显示。
    - 提供 `showAllPlugins` 和 `showCategoryPlugins` 函数分别用于显示所有插件和特定分类的插件。

LaTeX 编辑器窗口类

- `LaTeXEditor` 类（`411trial/pluginmanager.h`）
  - **功能**：实现一个 LaTeX 编辑器窗口，支持编译、预览、保存等功能。
  - 实现细节：
    - 继承自 `QDialog`，在构造函数中调用 `setupUI` 函数进行界面初始化。

    - 提供 `onCompileClicked`、`onPreviewClicked`、`onSaveClicked` 等槽函数，分别处理编译、预览、保存等操作。
    - 提供 `compileLaTeX` 函数用于编译 LaTeX 代码，`generateHTMLPreview` 函数用于生成 HTML 预览。

4. 用户管理相关模块

一、模块概述

用户管理模块主要负责系统的用户身份验证、信息管理及权限控制，包含用户登录、注册、密码管理及资料修改等核心功能，同时支持管理员对普通用户信息的编辑操作。

二、核心功能实现

1. 登录功能

- 验证机制
    - 支持普通用户和管理员账户登录，管理员账户为 `xlab@xlab.com`，密码 `XLab`，登录后进入账户管理界面。
    - 登录逻辑在 `user.md` 中更新，具体实现可能涉及账号密码校验、会话管理。
- 界面交互
    - 登录界面可能包含账号密码输入框、记住密码选项及登录按钮。
    - 错误处理：当账号密码错误时显示提示信息。

2. 注册功能

- 邮箱验证
    - 注册流程使用邮箱验证码机制，发送邮箱为 `xlabmailsystem@163.com`（密码在群内查看）。
    - 后端逻辑：生成随机验证码，通过 SMTP 协议发送至用户邮箱，验证环节需匹配用户输入的验证码。
- 信息收集
    - 注册表单可能收集用户姓名、学号、邮箱等基础信息，部分字段为必填项。

3. 密码管理

- 忘记密码功能
    - 由 `wyx` 负责开发，流程可能为：用户输入注册邮箱 → 发送密码重置链接 / 验证码 → 验证后设置新密码。
    - 安全机制：验证码有效期限制、密码强度校验（如长度、字符类型要求）。
- 修改密码
    - 登录状态下允许用户修改当前密码，需先验证原密码，再输入新密码并确认。

4. 用户信息管理

- 个人资料维护
    - 主页面显示用户基本信息（姓名、学号、头像等），支持编辑修改。
    - 头像上传功能：允许用户从本地选择图片，前端预览后上传至服务器，可能涉及图片格式和大小校验。

- 管理员权限
    - 管理员登录后可进入账户管理界面，编辑其他用户的姓名、学号等信息（不显示密码），后续计划扩展更多管理功能。

三、关键组件与数据结构

- 用户模型类
    - 可能定义 `User` 类存储用户信息，包含字段：账号、密码（加密存储）、姓名、学号、邮箱、头像路径、权限类型等。
- 登录控制器
    - 处理登录请求，调用数据库查询验证账号密码，生成并管理用户会话。
- 邮箱服务类
    - 封装 SMTP 发送功能，用于发送验证码和密码重置邮件，配置信息如邮箱账户、密码已在文档中提供。

5. 构建相关模块

- Makefile 文件（411trial/build/Desktop_Qt_6_9_0_MinGW_64_bit-Debug/Makefile）
    - **功能**：记录了项目构建过程中所需的 Qt 模块和配置文件，确保项目能够正确编译和链接。
    - **包含的模块**：包含了大量的 Qt 模块，如 `qt_lib_gui`、`qt_lib_qml`、`qt_lib_quick` 等，以及这些模块的私有文件。

三、小组成员分工情况

- **杨家铭**：负责大部分前端模块的开发，以及用户修改密码开发，还有扩展支持模块的开发，另外还包括用户界面的设计和实现，以及界面跳转逻辑的处理，最后进行了视频录制和报告撰写。
- **王韻晰**：负责用户登陆注册模块的开发，包括数据库的设计和管理，以及API调用的实现。
- **李昊润**：负责实验室界面的开发，以及视频的剪辑与后期。

四、项目总结与反思

总结

X-Lab项目成功实现了多学科、多领域功能的集成，为科研人员提供了一个便捷、全面的科研工具。通过前端、后端和扩展支持模块的协同工作，项目满足了实验室及科研人员在日常研究中的多样化需求。同时，项目也锻炼了我们团队成员的协作能力和技术实力，给我们了解和体验一个完整项目的工作流程提供了宝贵契机。

反思

- **分工方式**：最初计划按前端、后端进行分工，但在实际开发过程中发现这种方式容易导致代码阅读和理解上的困难。后续我们改为按模块和功能进行分工，使得模块功能之间相对独立，提高了开发效率。
- **工具使用**：在github的使用上，我们最初觉得git操作复杂，但后来通过实践发现，利用vscode连接git进行代码管理并不困难。同时，我们也意识到需要勤pull以保持代码库的同步。
- **前端设计**：在前端设计方面，我们应该从一开始就确定一个统一的风格和模板，以避免后续修改界面展示时的麻烦和不一致性。

后续改进方向

- **插件支持**：继续更新和完善插件支持，以满足更多科研领域的需求。
- **实验室功能优化**：进一步优化实验室功能，增加可兼容的学科和领域，提高其实用性和便捷性。
- **用户反馈**：积极收集用户反馈和建议，不断改进和完善项目功能，提升用户体验。