码上穿衣
MASHANGCHUANYI

北京大学
PEKING UNIVERSITY

穿搭推荐与管理系统作业报告

组员及分工

黄禄临

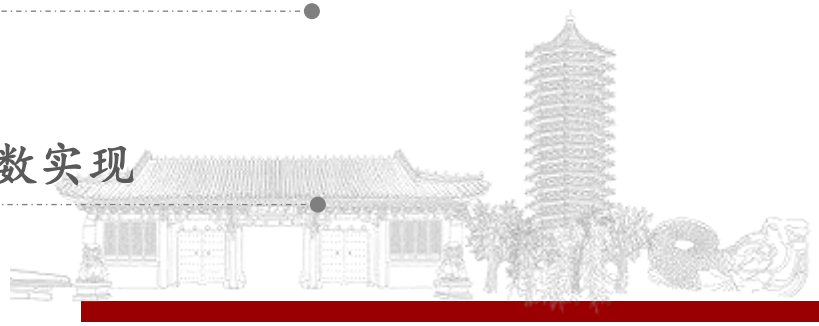
组织；推荐算法设计；界面美化

边天奕

界面设计与通信；界面美化

陈俊宇

信息存储设计；功能函数实现



一. 功能介绍

登录

用户使用已注册信息登录

注册

用户自定义用户名和密码

主页

四大功能跳转

一. 功能介绍

我的衣橱

用户的电子衣橱，可以增删或者修改自己的所有衣物

穿搭推荐

用户选择自己期望的风格与喜好颜色，输入当前天气、季节，系统根据用户拥有衣物和评分算法给出推荐的穿搭

穿搭日历

记录用户过往穿搭，用户可以根据日期找到当天的穿搭

用户信息管理

用户可以重新设置自己的密码，保护信息安全

二.UI 直接绘制部分

简单界面直接绘制：注册、登录、调整用户信息、主页、穿搭日历、服装档案、穿搭推荐等

外观：
使用 gridLayout 进行排版，让排版更加有序
利用 frame，实现边框样式调整和内部颜色填充

信息输入：
利用 QComboBox（下拉菜单选择）、QSpinBox（填写数字）、屏幕取色器（选择颜色）等按钮实现新增衣服信息和衣服推荐信息的输入

利用QColor类实现颜色的提取，利用屏幕取色器可准确提取衣服颜色，确保信息准确

通信：
通过userID传输用户信息，切换页面


```
void showHistory::returnMainPage(){
    this->close();
    MainWindow *myMainWindow = new MainWindow(userID);
    myMainWindow->show();
}
```

二.UI 代码实现部分

复杂页面：衣橱展示、推荐衣物等

展示衣橱中的衣服或系统推荐的穿搭时，衣服数量大、个数不定，UI界面复杂。我们通过代码实现而非直接绘制UI，通过ScrollLayout实现可滑动页面，将所有衣服加入可滑动页面中。主要设计思路如下：

将单件衣服通过Grid Layout
以每行两件的格式
加入对应类别Layout中

将Scroll Layout连同其
他按钮添加进总页面中

单件衣服 → **按类别的Layout** → **Scroll Layout** → **总页面**

设置特定Layout格式
将衣服标签（薄厚、款式等）
和照片加入单件衣服Widget

将多个类别的Layout
（上衣、裤子等）
按照纵向排列方式
加入Scroll Layout

二.信息存储与调用

使用Qt提供的QSettings类，将运行时数据存入本机注册表

QSettings原生函数：

```
QSettings settings("magicCloset", "LoginSystem");
settings.contains("lastUserId")
settings.setValue("lastUserId",0);
settings.value("size", 0)
settings.beginGroup("account_" + username);
settings.beginReadArray("clothes");
settings.beginWriteArray("clothes");
```

初始化并打开特定注册表

查询当前目录下是否存在值

设置值

获取值

创建或打开新分组

创建或打开可读数组

创建或打开可写数组

数据库功能函数：

```
saveToSettings(QSettings &settings, int id)
loadFromSettings(QSettings &settings)
deleteCloth(int userId, const Cloth& targetCloth)
getUserClothes(int userId)
```

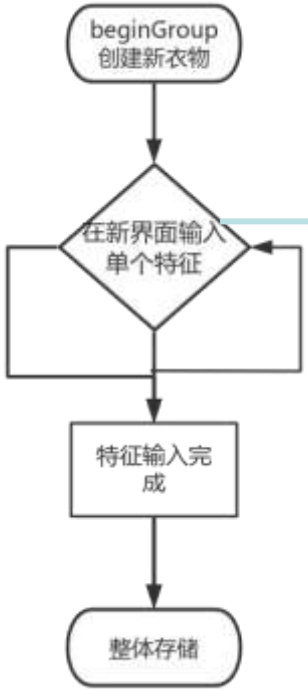
单件衣物的增删改查

Cloth类的设计

二.数据存储结构

```
UserContent/                          → 用户系统
└── id_123/                           → 特定用户数据
    ├── clothes/                      → 衣服数据
    │   ├── 0/                        当前衣物存储
    │   ├── 1/                        → cloth_0/
    │   └── size                      → cloth_1/
    │                                 当前衣物总数
    └── history/                      → 历史穿搭数据
        ├── 2023-11-15/               历史记录存储
        │   ├── 0/                    日期目录
        │   │   ├── 0/                第0次记录
        │   │   ├── 1/
        │   │   └── size
        │   ├── 1/                    本次记录衣物数
        │   └── size                  第1次记录
        └── 2023-11-16/               当天记录总数
            └── ...                   新日期
                                      相同结构
```

新衣物数据存储流程：



衣物特征包括：

类型
风格
图案
材质
厚度
季节
摆放位置

二. 推荐算法设计细节

北京大学 PEKING UNIVERSITY

一、数据收集

上传衣物时会收集衣服的类型、风格、材质、图案、厚度和适用季节等特征；在用户需要推荐时也会收集用户的场景、天气、季节和偏好等信息，并作为参数传入。

二、算法评分

核心是根据衣物特征与场景信息和用户偏好的适配性进行赋分。评分模型中风格和天气优先：风格体现商务、运动等场景的硬性约束，季节、气温也是穿搭的首要考虑因素；同时评分采用协同过滤原则，特征相似的也赋予低位分数；此外，根据时尚趋势进行加权赋分，如当季流行颜色、款式进行加分。

```
if(season == "春"){//春季适合浅和的色调
    if((item.getQColor().hueF() >= 0.05f && item.getQColor().hueF() <= 0.2f && item.getQColor().saturationF() <= 0.7f)
        score += 0.2f;
}
else if(season == "夏"){//夏季适合明亮清爽的色调
    if((item.getQColor().hueF() <= 0.4f || item.getQColor().hueF() >= 0.55f) && item.getQColor().valueF() > 0.7f)
        score += 0.2f;
}
else if(season == "秋"){//秋季适合偏色调
    if(item.getQColor().hueF() >= 0.05f && item.getQColor().hueF() <= 0.15f)
        score += 0.2f;
}
else{//冬季适合深色和冷色调
    if((item.getQColor().hueF() <= 0.4f || item.getQColor().hueF() >= 0.55f) && item.getQColor().valueF() <= 0.7f)
        score += 0.2f;
}
```

三、排序选择搭配

根据分数排序并选出搭配的衣服套装。

二.代码展示

北京大学 PEKING UNIVERSITY

Closet类：将衣物按类型设置多个layout

```
layoutTop->setSpacing(15);
layoutUnder->setSpacing(15);
layoutDress->setSpacing(15);
layoutShoes->setSpacing(15);
layoutAccessory->setSpacing(15);

layoutTop->setContentsMargins(2,20,2,20);
layoutUnder->setContentsMargins(2,20,2,20);
layoutDress->setContentsMargins(2,20,2,20);
layoutShoes->setContentsMargins(2,20,2,20);
layoutAccessory->setContentsMargins(2,20,2,20);
```

通过scrollLayout实现可滑动界面

```
scrollLayout->addLayout(layoutTop);
scrollLayout->addLayout(layoutUnder);
scrollLayout->addLayout(layoutDress);
scrollLayout->addLayout(layoutDress);
scrollLayout->addLayout(layoutShoes);
scrollLayout->addLayout(layoutAccessory);
```

登录系统对应函数

```
// 检查用户是否已存在
inline bool isUserExist(QSettings &settings, const QString &groupName);
// 存储账户密码
inline bool saveAccount(const QString &username, const QString &password);
// 检查密码正确
inline bool checkPassword(const QString &inputUsername, const QString &inputPassword);
```

高封装的数据接口

```
void saveToSettings(QSettings &settings, int id) const; // 保存至数据库
static Cloth loadFromSettings(QSettings &settings); // 从数据库加载衣服
static bool deleteCloth(int userId, const Cloth& targetCloth); // 删除衣物
static void addDailyOutfit(int userId, const QList<Cloth>& outfit); // 存至穿搭历史
static QList<QList<Cloth>> getDailyOutfits(int userId, const QDate& date); // 读取穿搭历史
static QList<Cloth> getUserClothes(int userId); // 读取用户衣物
```

根据用户偏好颜色与衣物颜色的适配度赋分

```
float calculateColorHarmony(const QColor& color1,const QColor& color2){
    float hueDiff = qAbs(color1.hueF() - color2.hueF());
    if(hueDiff > 0.5f)
        hueDiff = 1.0f - hueDiff;
    float satDiff = qAbs(color1.saturationF() - color2.saturationF());//计算饱和度差异
    float valueDiff = qAbs(color1.valueF() - color2.valueF());//计算明度差异
    float harmonyScore = hueDiff * 0.6f + satDiff * 0.2f + valueDiff *0.2f;
    return 1.0f - harmonyScore;
}
```

排序函数

```
QList<Cloth> recommendOutfit(const int& userID,const QString& occasion,const QString& season,
                             const QString& weather,const int temperature,const QColor& userPreferredColor){
    QList<Cloth> wardrobe = Cloth::getUserClothes(userID);

    QList<QPair<Cloth,float>> scoredItems;
    for(const auto& item : wardrobe){
        float score = calculateMatchScore(item,occasion,season,weather,temperature);
        scoredItems.append(qMakePair(item,score));
    }
    if(userPreferredColor.isValid()){
        for(auto& pair : scoredItems){
            float colorScore = calculateColorHarmony(pair.first.getQColor(),userPreferredColor);
            pair.second += colorScore*0.3f;
        }
    }
    std::sort(scoredItems.begin(),scoredItems.end(),
        [](const QPair<Cloth,float>&a,const QPair<Cloth,float>& b){
        return a.second > b.second;
    });
```

三. 项目总结和反思

本次项目我们从日常中很多人不懂穿搭又想穿的好看的窘境中获得灵感设计了一个穿搭管理和推荐系统，实现了用户管理、衣橱管理、系统推荐、穿搭日历等基本功能。

我们的项目的亮点如下：使用qcolor类，用户可以直接在上传的图片上选取自己衣服的颜色，使得取色更为精确；考虑用户个性化需要，允许用户选择偏好或者完全由系统推荐；推荐算法综合考虑颜色（对比度、饱和度、明度等）、天气、场合和衣服信息等因素，较为全面。

当然我们的设计也存在一些不足，如我们推荐算法没有收集用户信息，不能像米兰、上海现有的人工颜色测试等用AI给出适合用户的颜色、风格等信息，只能依靠用户给出的偏好；同时我们的算法在衣服的适配性上可能存在不足，选出的衣物的风格有一定可能不适配；此外我们的界面的风格过于多元，整体上有割裂感等等。希望以后能早点选好整体风格和设计出更好的算法

与此同时，由于我们小组多为线上交流，总存在信息不对等和回复不及时问题，希望以后多一点在一起设计的时间，增强面对面交流。