

QT 作业报告

项目名称：今天吃什么（WhatToEat）

小组编号：26（发际线与我作战）

小组成员：翟睿辰 肖凯文 温锦程

指导教师：郭炜、刘家瑛

一、项目简介：

随着北京大学食堂菜品种类的日益丰富，同学们在选择餐食时往往感到迷茫。在快节奏的校园生活中，下了课的同学在走到食堂的 10min 路程中就要想好“今天去食堂吃什么？”这一艰巨的、困难的、富有挑战性的问题。为了解决这一痛点，我们开发了一个基于 QT 的智能食堂菜品推荐系统，旨在通过提供详尽的菜品信息和筛选功能，帮助同学们快速做出餐食选择，从而节省时间并提升用餐体验。一个能够提供即时、个性化推荐的工具，将极大地提升同学们的用餐决策效率。

二、程序功能介绍

2.1 功能简介

完成并维护了一个北京大学部分食堂的菜品列表，可以添加/删除菜品并且可以按照给定的标签（例如价格、辣度、食堂等）缩小范围。有具体的菜品描述和图片，帮助同学们在不知道吃什么的时候挑选今天的餐食。



(概览图)

2.2 主要功能

窗口：主要通过 QT 实现。左侧展示符合条件的菜品（可滚动浏览）和点击选中的菜品详情页，右侧有不同的筛选标准，通过捕获鼠标点击来更改当前的筛选条件（MatchLabels）

菜品：数据用 json 的方式存储，每个派生类有不同的标签“辣/不辣”“米饭/面食”“价格”等，通过搜索名称（搜索与关键字匹配度最高的 name）或者通过勾选条件进行条件过滤，最后给出一个符合条件的菜品集合，展示在左侧展示栏中。

维护：左上角有一个“+”用于添加菜品，点击可以调出来 new_dish_ui，输入菜品名称、标签、描述、价格并选择图片即可添加新菜品。每个菜品的右下角有“×”可以删除菜品（调用 Dish 的 Remove 函数）。

三、项目实施

3.1 模块设计

本项目主要分成两个模块：显示模块和数据模块。

3.1.1 显示模块主要包括符合条件的菜品展示，菜品详情页和添加新菜品的窗口布置
菜品展示：



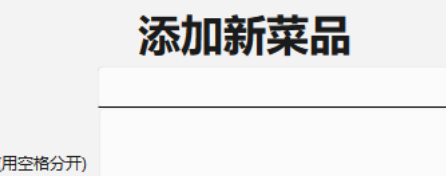
包括一个可用鼠标滚轮控制上下滚动的展示框，里面是符合条件的 Dishes

菜品详情页：




可以看到选中的菜品的详情，包括名称、价格、标签和具体描述，还有实拍图片可供参考（无美颜 无滤镜、所见即所得）

添加新菜品：



The screenshot shows a dialog box titled "添加新菜品" (Add New Dish) with a close button (X) in the top right corner. The dialog contains four input fields: "菜名" (Dish Name), "标签(用空格分开)" (Tags (separated by spaces)), "描述" (Description), and "价格" (Price). The "价格" field is a numeric input with a value of "0" and up/down arrows. To the right of the price field is a button labeled "选择图片路径" (Select Image Path). At the bottom of the dialog are "OK" and "Cancel" buttons.

点击左上角的可以添加新菜品↑，打开一个新的窗口，在其中输入相关信息即可添加新菜品。

3.1.2 数据模块主要包括菜品的存储（以 json 格式存储）和通过 1.搜索关键字 2.价格 3.勾选合适的标签 来进行筛选，最终给出一个符合条件的菜品 list

目前存储进去的菜品清单:

```

    "description": "4800块钱买到一块好大的鸡排饭！",
    "img_path": "/Resources/Images/DaJiangJun.jpg",
    "labels": [
        "米饭",
        "鸡排",
        "李五食堂",
        "超送紫菜",
        "鸡蛋"
    ],
    "name": "大将军鸡排饭",
    "price": 14
},
{
    "description": "黄焖鸡不用多说了吧，看看每天中午二排多长的队伍，就知道黄焖鸡有多好吃了！",
    "img_path": "/Resources/Images/HuangMouJi.jpg",
    "labels": [
        "米饭",
        "自选辣度",
        "鸡排",
        "鸡蛋二蛋",
        "有紫菜",
        "排队时间较长"
    ],
    "name": "黄焖鸡",
    "price": 15
},
{
    "description": "李五的土豆牛肉盖饭，土豆丝酸酸开胃",
    "img_path": "/Resources/Images/TuDoNiouFan.jpg",
    "labels": [
        "米饭",
        "鸡排",
        "牛肉",
        "李五食堂",
        "自选辣度",
        "卤蛋"
    ],
    "name": "土豆牛肉盖饭",
    "price": 16
},
{
    "description": "其实他家还有好多多样的饭，鸡排只要13，猪肉16，而且还会送你炸好饭",
    "img_path": "/Resources/Images/FaiJiangBanFan.jpg",
    "labels": [
        "米饭",
        "鸡排",
        "牛腩",
        "鸡排",
        "肥羊",
        "鸡排",
        "李五食堂",
        "自选辣度",
        "酸豆角/花生米"
    ],
    "name": "肥羊排饭",
    "price": 19
},
{
    "description": "作为山西人来说，我只认可北京的煎鱼中規中矩吧，个人喜欢吃宽面",
    "img_path": "/Resources/Images/YouFuKian.jpg",
    "labels": [
        "煎鱼",
        "煎煎煎煎",
        "李五食堂",
        "自选辣度",
        "油条辣"
    ],
    "name": "油泼宽",
    "price": 12
},
{
    "description": "这个鱼片蛮好吃的，有一點点辣，对执来说还蛮下饭的(不吃辣拜拜人)",
    "img_path": "/Resources/Images/TengJiaoYu.jpg",
    "labels": [
        "米饭",
        "酸辣",
        "鱼片",
        "李五食堂",
        "自选辣度"
    ],
    "name": "酸辣鱼片拌饭",
    "price": 18
},
{
    "description": "菜园的海苔饭，印象里还是蛮好吃的，没吃过的可以去尝尝",
    "img_path": "/Resources/Images/HaiTaiYu.jpg",
    "labels": [
        "米饭",
        "鸡排",
        "肉松",
        "鸡蛋二蛋",
        "海苔"
    ],
    "name": "海苔饭",
    "price": 14
}

```

```

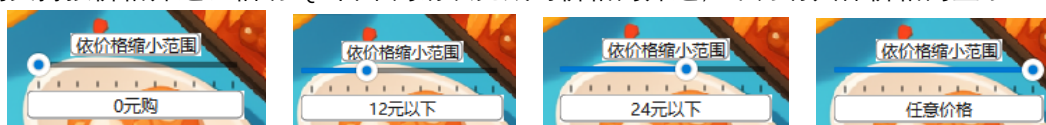
"description": "这个鸡排好像吃的人挺多的，口碑还行。但我其实觉得有点腻？个人意见仅供参考！",
"img_path": "/Resources/Images/JiPaiFan.jpg",
"labels": [
    "米饭",
    "鸡排",
    "鸡排",
    "鸡排",
    "家园二层",
    "自选素菜"
],
"name": "鸡排饭",
"price": 16
},
{
"description": "///待补充///",
"img_path": "/Resources/Images/TieBanCai.jpg",
"labels": [
    "米饭",
    "汤类",
    "排骨",
    "田园二厨",
    "自选素菜"
],
"name": "铁板菜",
"price": 16
},
{
"description": "“肥牛有整整一碗，米饭泡到汤里刚好一碗的量，稍微有点辣，我觉得蛮不错的。”",
"img_path": "/Resources/Images/JinJiangFeiNiu.jpg",
"labels": [
    "米饭",
    "麻辣",
    "配牛",
    "家园三层",
    "无果菜"
],
"name": "金汤肥牛",
"price": 18
},
{
"description": "“香鱼进门左手边的鱼粉，蛮好吃的。”",
"img_path": "/Resources/Images/JinYangYuFen.jpg",
"labels": [
    "鱼粉",
    "酸辣",
    "鱼片",
    "香鱼食堂",
    "油麦菜"
],
"name": "金汤鱼粉",
"price": 16
},
{
"description": "“数学课和英语均有售卖，我每天都喝，性价比很高。”",
"img_path": "/Resources/Images/Water.jpg",
"labels": [
    "清凉",
    "健康",
    “大众饮品”,
    “解暑”,
    “不辣”
],
"name": "水",
"price": 0
},
{
"description": "“与实际预期的口感差别较大，需要配水才能下咽，没吃过的朋友们要注意了。”",
"img_path": "/Resources/Images/GaoShu.jpg",
"labels": [
    “口味较硬”,
    “理科数字饼”,
    “营养丰富”,
    “高级美味”
],
"name": “高等数学”,
"price": 85
},
{
"description": "“这个应该是他家最鲜的饭了，红汤内肉量不算，肥肠欠之。不能吃就尝尝。”",
"img_path": "D:/Qt/Projects/WhatToEat-master/build/Desktop_Qt_6_7_1_MinGW_64_bit-Debug/Resources/I",
"labels": [
    "米饭",
    "特辣",
    "培根",
    "田园二厨"
],
"name": "香辣鸡丝拌饭",
"price": 16
}

```

可以在此输入想吃的关键词，然后点击“搜索”，程序会进行关键词搜索，并按照相似度进行排序



右侧支持按价格筛选。借助 Qt 自带的滑块完成对价格的筛选，下方有具体价格的显示



还可以通过勾选特定的食堂或者口味进行筛选：



这些操作会更改筛选条件，最后在所有的菜品中筛选完后给左侧的展示框一个更新过后的 list，并在其中展示。

3.2 类设计

3.2.1 展示模块中的菜品(Dish_ui)和菜品详情(DishDetail_ui)均由 Qt 自带的 QWidget 派生而来

```
dishui.h
#ifndef DISHUI_H
#define DISHUI_H

#include <QWidget>
#include "Dish.h"
namespace Ui {
class DishUI;
}

class DishUI : public QWidget
{
    Q_OBJECT

public:
    explicit DishUI(QWidget *parent = nullptr);
    ~DishUI();
    void update(Dish &d);

signals:
    void dishUIClicked(Dish &d);
    void dishUIDelete(Dish& d);

private slots:
    void on_pushButton_clicked(); // Slots name
    void on_pushButton_2_clicked();

private:
    Ui::DishUI *ui;
    Dish d;
};

#endif // DISHUI_H
```

```
dishdetailui.h
#ifndef DISHDETAILUI_H
#define DISHDETAILUI_H

#include <QWidget>
#include "Dish.h"
namespace Ui {
class dishDetailUI;
}

class dishDetailUI : public QWidget
{
    Q_OBJECT

public:
    explicit dishDetailUI(QWidget *parent = nullptr);
    ~dishDetailUI();

public slots:
    void update(Dish &d);

private:
    Ui::dishDetailUI *ui;
};

#endif // DISHDETAILUI_H
```


具体实现 (.cpp)

```
dishui.cpp
#include "dishui.h"
#include <QCoreApplication>
#include "ui_dishui.h"
DishUI::DishUI(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::DishUI)
{
    ui->setupUi(this);
    resize(480, 120);
}

DishUI::~DishUI()
{
    delete ui;
}

void DishUI::update(Dish &d)
{
    // 设置菜名
    ui->dishNameText->setText(d.name);
    ui->dishNameText->setFont(QFont("Arial", 20));
    // 设置价格
    ui->priceText->setText(to_string((int) d.price).c_str());

    QPixmap pixmap;
    // very stupid , but useful for now
    if (d.img_path.startsWith("/Resources/Images"))
        pixmap.load(QCoreApplication::applicationDirPath() + d.img_path);
    else
        pixmap.load(d.img_path);

    if (pixmap.isNull()) {
        qDebug() << "Can not find image at:" << d.img_path;
    } else {
        // 缩放图片
        pixmap = pixmap.scaled(180, 120);
        ui->imageLabel->setPixmap(pixmap);
    }

    QString labelsText = "";
    for (auto label : d.labels) {
        labelsText += '#' + label + ' ';
    }
    ui->labelsText->setText(labelsText);
    this->d = d;
}

void DishUI::on_pushButton_clicked()
{
    qDebug() << d.name << "clicked!";
    emit dishUIClicked(d);
}

// delete btn
void DishUI::on_pushButton_2_clicked()
{
    qDebug() << "del" << d.name;
    emit dishUIDelete(d);
}
```

```
dishdetailui.cpp
#include "dishdetailui.h"
#include <QCoreApplication>
#include "ui_dishdetailui.h"

dishDetailUI::dishDetailUI(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::dishDetailUI)
{
    ui->setupUi(this);
}

dishDetailUI::~dishDetailUI()
{
    delete ui;
}

void dishDetailUI::update(Dish &d)
{
    qDebug() << "dishDetailUI update to:" << d.name;
    // 设置菜名
    ui->nameText->setText(d.name);
    ui->nameText->setFont(QFont("Arial", 30));
    // 设置价格
    ui->priceText->setText(to_string((int) d.price).c_str());
    ui->priceText->setFont(QFont("Arial", 20));

    QPixmap pixmap;
    // very stupid , but useful for now
    if (d.img_path.startsWith("/Resources/Images"))
        pixmap.load(QCoreApplication::applicationDirPath() + d.img_path);
    else
        pixmap.load(d.img_path);

    if (pixmap.isNull()) {
        qDebug() << "Can not find image at:" << d.img_path;
    } else {
        // 缩放图片
        pixmap = pixmap.scaled(180, 120);
        ui->imageLabel->setPixmap(pixmap);
    }

    QString labelsText = "";
    for (int i = 0; i < d.labels.count(); i++) {
        if (i % 2 == 1)
            labelsText += '#' + d.labels[i] + '\n';
        else {
            labelsText += '#' + d.labels[i] + " ";
        }
    }
    ui->labelText->setText(labelsText);

    ui->detailLabel->setText(d.description);
}
```

而新建菜品的 newdish 由 QDialog 派生而来：

```
newdishui.h
#ifndef NEWDISHUI_H
#define NEWDISHUI_H

#include <QDialog>
#include "Dish.h"

namespace Ui {
class NewDishUI;
}

class NewDishUI : public QDialog
{
    Q_OBJECT

public:
    explicit NewDishUI(QWidget *parent = nullptr);
    ~NewDishUI();

private slots:
    void on_lineEdit_textChanged(const QString &arg1);
    void on_spinBox_valueChanged(int arg1);
    void on_textEdit_textChanged();
    void on_pushButton_clicked();
    void on_buttonBox_accepted();
    void on_textEdit_2_textChanged();

signals:
    void dishAddComplete(Dish &d);

private:
    Ui::NewDishUI *ui;
    Dish d;
};

#endif // NEWDISHUI_H
```

```
newdishui.cpp
#include "newdishui.h"
#include "ui_newdishui.h"
#include <QFileDialog>

NewDishUI::NewDishUI(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::NewDishUI)
{
    ui->setupUi(this);
}

NewDishUI::~NewDishUI()
{
    delete ui;
}

// dish name changed
void NewDishUI::on_lineEdit_textChanged(const QString &arg1)
{
    d.name = arg1;
}

// dish price changed
void NewDishUI::on_spinBox_valueChanged(int arg1)
{
    d.price = arg1;
}

// label changed
void NewDishUI::on_textEdit_textChanged()
{
    QString labelStr = ui->textEdit->toPlainText();
    QList<QString> labels = labelStr.split(' ');
    labels.removeAll(' ');
    d.labels = labels;
}

// select path
void NewDishUI::on_pushButton_clicked()
{
    QString path = QFileDialog::getOpenFileName(
        this, tr("选择图片文件"), QCoreApplication::applicationDirPath(), tr("Image Files (*.jpg *.png)"));
    qDebug() << "path choose:" << path;
    d.img_path = path;
}

void NewDishUI::on_buttonBox_accepted()
{
    // TODO: 判断输入是否符合要求
    emit dishAddComplete(d);
}

// description changed
void NewDishUI::on_textEdit_2_textChanged()
{
    d.description = ui->textEdit_2->toPlainText();
}
```

3.2.2 对于菜品，我们写了一个类 Dish，每个菜品都是一个 Dish 类，有自己的名字、价格、图片路径、标签和介绍，并内嵌了转化为 Json 格式的函数 toJson()，并重载了比较运算符便于后续搜索操作。

```
Dish.h
#ifndef DISH_H
#define DISH_H

#include <vector>
#include <QJsonArray>
#include <QJsonObject>
#include <QJsonValue>
using namespace std;

class Dish
{
public:
    // 菜品名称
    QString name = "";

    // 菜品价格
    double price = 0;

    // 图片路径
    QString img_path = "";

    // 标签列表
    QList<QString> labels = {};

    // 介绍
    QString description = "";

    Dish(QJsonObject &jsonObj)
    {
        this->name = jsonObj.value("name").toString();
        this->price = jsonObj.value("price").toDouble();
        this->img_path = jsonObj.value("img_path").toString();
        this->description = jsonObj.value("description").toString();
        auto tmpArr = jsonObj.value("labels").toArray();
        for (auto element : tmpArr) {
            this->labels.push_back(element.toString());
        }
    }

    Dish(){};

    QJsonObject toJson()
    {
        QJsonObject jsonObj;
        jsonObj.insert("name", name);
        jsonObj.insert("price", price);
        jsonObj.insert("description", description);
        jsonObj.insert("img_path", img_path);

        QJsonArray arr = {};
        for (auto qstr : labels) {
            arr.append(QJsonValue(qstr));
        }

        jsonObj.insert("labels", arr);
        return jsonObj;
    }

    friend bool operator==(const Dish &d1, const Dish &d2) { return d1.name == d2.name; }

    friend bool operator==(const Dish &d1, const QString &name) { return d1.name == name; }
};

#endif // DISH_H
```

搜索通过下页的 utils.h 实现：

```

utils.h
Filter

#ifndef UTILS_H
#define UTILS_H

#endif // UTILS_H

#include "Dish.h"
// if not
// 按价格给Dish排序
inline bool priceCmp(const Dish &d1, const Dish &d2)
{
    if (d1.price == d2.price) {
        return d1.name < d2.name;
    } else {
        return d1.price < d2.price;
    }
}

// 按名字给Dish排序
inline bool nameCmp(const Dish &d1, const Dish &d2)
{
    if (d1.name == d2.name) {
        return d1.price < d2.price;
    } else {
        return d1.name < d2.name;
    }
}

// 计算Dish与keyWord相似度
inline double calcSimilarity(const Dish &d, const QString &keyWord)
{
    // 目前计算的方法是：看Dish.name和Labels与keyWord有几个共同字符
    int repeatedChar = 0;
    for (QChar c : keyWord) {
        if (d.name.contains(c)) {
            repeatedChar++;
        }

        for (auto label : d.labels) {
            if (label.contains(c)) {
                repeatedChar++;
            }
        }
    }
    return repeatedChar;
}

// 按搜索关键字相似度给Dish排序
class similarityCmp
{
public:
    similarityCmp(QString &keyWord)
    : keyWord(keyWord)
    {}

    bool operator()(const Dish &d1, const Dish &d2)
    {
        return calcSimilarity(d1, keyWord) > calcSimilarity(d2, keyWord);
    }
};

// 条件过滤器
class Filter
{
public:
    // 最高价格
    double Mprice = 100000;
    // 必须至少符合一个的Label
    QList<QString> matchLabels = {};
    // 必须不包含的Label
    QList<QString> unmatchLabels = {};

    Filter(double Mprice = 100000,
           QList<QString> matchLabels = {},
           QList<QString> unmatchLabels = {})
    {
        this->Mprice = Mprice;
        this->matchLabels = matchLabels;
        this->unmatchLabels = unmatchLabels;
    }

    // 判断Dish是否被过滤，不符合条件的返回true
    bool operator()(Dish &d)
    {
        if (d.price > Mprice)
            return true;

        for (auto label : unmatchLabels) {
            if (d.labels.contains(label)) {
                return true;
            }
        }

        // 为空说明没有限制
        if (matchLabels.empty())
            return false;

        for (auto label : matchLabels) {
            if (d.labels.contains(label)) {
                return false;
            }
        }

        return true;
    }
};

```

3.2.3 主窗口 MainWindow 及其实现： MainWindow 从 Qt 自带的 QMainWindow 派 生而来

背景图

Qt 自带的滑块，实现依照价格的筛选

后面全部是重复的函数，类似的搜索逻辑筛
选其他食堂和“米饭”、“面食”，“辣”、“不辣”...

```
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow){
    ui->setupUi(this);

    //ui->scrollAreaWidgetContents->installEventFilter(this);
    //ui->scrollArea->setWidgetResizable(true); //使scrollArea可滚动
    //setCentralWidget(ui->scrollArea);
    QVBoxLayout *layout = new QVBoxLayout();
    layout->addWidget(&dishDetailUI);
    ui->dishDetailUIContainer->setLayout(layout);
}

MainWindow::~MainWindow(){
    delete ui;
}

bool MainWindow::eventFilter(QObject *watched, QEvent *event){
    //if (watched == ui->scrollAreaWidgetContents && event->type() == QEvent::Paint) {
    //paintWidget(dishDataLoader.curDishes);
    //}

    //return QWidget::eventFilter(watched, event);
}

void MainWindow::paintEvent(QPaintEvent *) //画背景图 {
    QPainter p(this); //创建一个painter,指定窗口为画布
    p.drawPixmap(this->rect(), QPixmap(":/background")); //画背景图
}

void MainWindow::paintWidget(QList<Dish> curDishes){
    //创建一个painter,指定窗口为scrollAreaWidgetContents
    //QPainter pl(ui->scrollAreaWidgetContents);

    // 添加若干张图片到布局中
    //dishDataLoader.filt(filter, nameCmp);
    for (int i = 0; i < dishDataLoader.curDishes.size(); ++i) {
        // 在scrollAreaWidgetContents上绘制图片
        //pl.drawPixmap(60, 100 + i * 70, 480, 60, QPixmap(QDir::currentPath() + dishDataLoader.curDishes[i].img_path));
        //QDebug() << QDir::currentPath() + dishDataLoader.curDishes[i].img_path;
        // *QMyLabel *myLabel = new QMyLabel(ui->scrollAreaWidgetContents);
    }

    //if(dishNum > 7){
    //ui->scrollAreaWidgetContents->setMinimumSize(0, 600 + (dishNum - 7) * 70); //改变小widget的大小
    //}

void MainWindow::mousePressEvent(QMouseEvent *e){
    qDebug() << e->pos();
}

void MainWindow::on_search_clicked(){
    keyword = ui->textEdit->toPlainText();

    updateDishUI();
}

void MainWindow::on_priceSlider_valueChanged(int value){
    qDebug() << "price slider value changed to:" << value;
    Mprice = value == 100 ? value : value * 0.4;
    if (Mprice < 100 and Mprice > 0) {
        ui->priceBoundLabel->setText((std::to_string(Mprice) + ("元以下")).c_str());
    } else if (Mprice == 100) {
        ui->priceBoundLabel->setText("任意价格");
    } else {
        ui->priceBoundLabel->setText("6元购");
    }

    ui->priceBoundLabel->setAlignment(Qt::AlignCenter);

    updateDishUI();
}

void MainWindow::updateDishUI(){
    // 先按条件过滤和排序菜品

    if (keyword == "") {
        // 没有搜索内容时按价格排序
        dishDataLoader.filt(Filter(Mprice, matchLabels, unmatchedLabels, priceCmp);
    } else {
        // 有搜索内容时按相似度排序
        dishDataLoader.filt(Filter(Mprice, matchLabels, unmatchedLabels, similarityCmp(keyword));
    }

    // TODO: 绘制菜品
    QVBoxLayout *vLayout = new QVBoxLayout();
    //vLayout->size
    QWidget *qwgt = new QWidget();
    for (auto d : dishDataLoader.curDishes) {
        DishUI *dishUI = new DishUI(this);
        dishUI->update(d);
        vLayout->addWidget(dishUI);
        connect(dishUI, &DishUI::dishUIClicked, &dishDetailUI, &dishDetailUI::update);
        connect(dishUI, &DishUI::dishUIDelete, this, &MainWindow::rmv_dish);
    }
    qwgt->setLayout(vLayout);
    ui->dishScrollArea->setWidget(qwgt);
}

void MainWindow::rmv_dish(Dish& d) {
    dishDataLoader.removeDish(d);
    updateDishUI();
}

void MainWindow::add_matchLabel(QString label){
    // 避免重复
    if (matchLabels.contains(label))
        return;

    matchLabels.append(label);
    updateDishUI();
}

void MainWindow::rmv_matchLabel(QString label){
    if (not matchLabels.contains(label))
        return;

    matchLabels.removeAll(label);
    updateDishUI();
}

// 家园一层
void MainWindow::on_flavor_yes_stateChanged(int arg1){
    QString lbl = "家园一层";
    if (arg1 == 0) {
        rmv_matchLabel(lbl);
    } else {
        add_matchLabel(lbl);
    }
}
```


三、小组成员分工：

肖凯文：完成了菜品(Dish)类的编写和相关的数据处理(similarityCmp 和 Filter)

温锦程：负责 ui 部分的工作（背景图和 MainWindow 整体框架的搭建）

翟睿辰：菜品数据集的编写和视频展示&作业报告（以及添加了部分条件的筛选）

四、项目总结与反思

4.1 项目总结

项目成果：成功开发了一个基于 QT 的智能食堂菜品推荐系统，实现了菜品信息的展示、筛选和个性化推荐，有效解决了学生在食堂选择餐食时的困惑。

技术实现：通过使用 QT 框架，我们实现了一个用户友好的界面，并通过 JSON 格式有效管理了菜品数据。搜索和筛选功能的实现，提高了系统的实用性和响应速度。

团队协作：在项目开发过程中，团队成员各自承担了不同的任务，锻炼了我们的沟通和协作能力。通过定期召开会议（线上、线下）和有效的交流沟通，提高了工作效率和项目的推动进度。

4.2 反思与改进

用户界面的改进：由于时间限制，项目在 UI 设计上存在一些粗糙之处，例如搜索框和筛选条件的圆角设计、字体大小和自适应窗口的调整。未来将重点优化这些细节，提升用户界面的美观度和易用性。

菜品信息编辑：目前系统缺少对菜品信息的编辑功能。未来将增加修改菜品的选项，允许用户对菜品标签、名称或价格进行修改，以适应不断变化的菜品信息和用户需求。

GitHub 协作和版本控制：由于未能有效利用 GitHub 进行团队协作，导致开发过程中存在效率不高的问题。未来项目将从一开始就建立 git 仓库，利用分支管理、合并请求等工具来提高团队协作效率；缺乏有效的版本控制机制，使得代码管理和问题追踪变得困难。将通过 git 进行代码的版本控制，确保每次提交都是可追踪和可回滚的。

后续可以考虑加入智能推荐：引入机器学习算法，实现更智能的菜品推荐，根据用户的历史选择和偏好进行个性化推荐。