****

高级程序设计语言课程设计

设计题目 \_\_\_火车票管理/购票系统\_\_\_\_\_\_\_\_\_\_\_

专 业：\_\_\_\_\_\_计算机类\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

班 级：\_\_\_\_\_\_\_02\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

设 计 者：\_\_202200800534 陈硕成\_\_\_\_\_\_\_\_\_\_

\_\_202200800594 司居正\_\_\_\_\_\_\_\_\_\_

\_\_202200800627 袁恩晗\_\_\_\_\_\_\_\_\_\_

2023年 6 月 10 日

本系统实现了

A．火车时刻信息录入。

包括车次、日期、起点、终点、开车时间、到达时间、票价。

B．火车时刻信息查询。

按照车次查询

按终点查询

按起点查询

按终点和日期查询

C．统计

按终点统计每日的车次数

按起点统计每日的车次数

D.订票

订票、退票、改签

（2）录入火车时刻信息

统计火车车次

退出

等功能，各成员分工如下：

陈硕成：工作量**33%** 票据文件加载与存储，查找票据、按各标签排序、购票、改签、退票用户已购买票据加载与存储、用户已购买票据查询等功能，函数为：

class widget\_user

{

public:

    Widget\_User(QWidget \*parent = nullptr);

    ~Widget\_User();

    void setUser(user t); //设置当前用户

private slots:

    void on\_pushButton\_clicked(); //查询按键

    void check(); //查询

    bool read(); //读取票据文件

    void on\_tableWidget\_cellDoubleClicked(int row, int column); //双击购买/改签单元格事件

    bool refund(ticket a); //退票

    void on\_pushButton\_3\_clicked(); //个人中心按钮

    bool readUsertickets(); //读取用户已购票据

    void on\_pushButton\_4\_clicked(); //退出登录按钮

    bool save(); //保存

    void get(bool m, ticket a); //窗口间传递changingTicket

    void timerUpdate(); //时钟

    void updateTicketData(); //更新票据文件

    void tableUpdate(); //更新列表

protected:

    void closeEvent(QCloseEvent \*event); //关闭窗口动作

};

class user{

    user();//默认构造函数

    user(QString a, QString b):account(a), password(b){} //合成构造函数

    bool buy(ticket a); //购票

    bool change(ticket a, ticket b); //改签

    bool refund(ticket a); //退票

    QString getid() const; //获取用户名

    QList<ticket> getTickets() const; //获取用户已购票据

    void setTickets(QList<ticket> t); //设置用户已购票据

};

class ticket

{

public:

    ticket();//构造函数

    bool operator ==(const ticket l)

    {

        /\*判断两票相同与否\*/

        if(id == l.id) return true;

        return false;

    }

};

class dialog1

{

    public:

    explicit dialog1(QWidget \*parent = nullptr);

    void setUser(user t); //设置当前用户

    ~dialog1();

signals:

    void send(bool mode, ticket a); //窗口间传递changingTicket

private slots:

    void on\_tableWidget\_2\_cellDoubleClicked(int row, int column); //双击改签/退票单元格事件

    void on\_pushButton\_clicked(); //已购票据查询按钮

    void on\_pushButton\_2\_clicked(); //退出按钮

}

司居正：工作量**33%** 用户登录、管理员登录、注册用户、修改用户密码、用户数据的读取、存入、修改，用户密码的加密、ui界面美化等功能，函数为：

```

chgdialog.h

#ifndef CHGPDDIALOG\_H

#define CHGPDDIALOG\_H

#include <QWidget>

namespace Ui {

class chgpdDialog;

}

class chgpdDialog : public QWidget

{

    Q\_OBJECT

public:

    explicit chgpdDialog(QWidget \*parent = nullptr);//构造函数

    ~chgpdDialog();//析构函数

    void paintEvent(QPaintEvent \*event);//边框加一条黑边以便识别

private slots:

    void on\_btn\_back\_clicked();// 返回到mainwindow

    void on\_btn\_chg\_clicked();// 确认修改密码

protected:

    //用于鼠标拖动窗口移动

    void mousePressEvent(QMouseEvent \*event) override;

    void mouseMoveEvent(QMouseEvent \*event) override;

    void mouseReleaseEvent(QMouseEvent \*event) override;

private:

    Ui::chgpdDialog \*ui;

    QPoint reltvPos; // 相对坐标

    bool m\_bMove; // 是否移动

};

#endif // CHGPDDIALOG\_H

```

```

logindialog.h

#ifndef LOGINDIALOG\_H

#define LOGINDIALOG\_H

#include <QString>

class LoginDialog

{

public:

    bool LoginUser(QString account , QString password);//用于进行登录的操作

public:

    LoginDialog();//默认构造函数

};

#endif // LOGINDIALOG\_H

```

```

md5.h

#ifndef MD5\_H

#define MD5\_H

#include <string>

using std::string;

class md5

{

public:

    //构造函数，用于初始化类的成员变量

    md5();

    /\*MD5主循环函数，用于处理每个512位的分组。

    根据索引 i 的不同值，选择不同的运算方式，并更新临时变量。\*/

    void mainLoop(unsigned int M[]);

    unsigned int\* add(string str);//填充函数，将输入的字符串填充为64字节的整数倍，并添加长度信息。

    string changeHex(int a);//将整数转换为16进制字符串表示。

    /\*计算给定字符串的MD5哈希值。初始化临时变量并调用 add 函数对输入字符串进行填充，

    然后通过 mainLoop 处理每个512位分组，最后将结果转换为16进制字符串形式并返回。\*/

    string getMD5(string source);

};

#endif // MD5\_H

```

```

regdialog.h

#ifndef REGDIALOG\_H

#define REGDIALOG\_H

#include <QWidget>

namespace Ui {

class RegDialog;

}

class RegDialog : public QWidget

{

    Q\_OBJECT

public:

    explicit RegDialog(QWidget \*parent = nullptr);//默认构造函数

    ~RegDialog();//默认析构函数

    bool Gender\_Choose();//用于性别选择，其中男性返回1，女性返回0，默认是男性

    void paintEvent(QPaintEvent \*event);//边框加一条黑边以便识别

private slots:

    void on\_Btn\_Back\_clicked();//返回mainwindow

    void on\_Btn\_Reg\_clicked();//注册用户

//用于鼠标拖动无边框窗口移动

protected:

    void mousePressEvent(QMouseEvent \*event) override;// 鼠标按下事件处理函数

    void mouseMoveEvent(QMouseEvent \*event) override; // 鼠标移动事件处理函数

    void mouseReleaseEvent(QMouseEvent \*event) override;// 鼠标释放事件处理函数

private:

    Ui::RegDialog \*ui;

    QPoint reltvPos; // 相对坐标

    bool m\_bMove; // 是否移动

};

#endif // REGDIALOG\_H

```

```

user\_crl.h

#ifndef USER\_CRL\_H

#define USER\_CRL\_H

#include <user.h>

#include <QList>

#include <QString>

class user\_Crl

{

    QList<user> userList;//创建一个用户链表，用于存储每一个用户的信息

public:

    user\_Crl();//默认构造函数，用于用户链表的初始化，将文件中的内容导入链表

    bool AddUser(QString ac , QString psd , bool gd , QString nm , bool op);//添加一个新用户，用于注册功能

    bool ChgUser(QString ac , QString oldpd , QString newpd, QString renewpd);//修改用户信息，用于修改密码

    bool LoginUser(QString ac , QString psd );//用于用户登录

    bool checkUser\_Password(QString account , QString password);//验证用户是否存在以及密码是否正确，用于登录功能和修改密码的检验功能

    bool checkUser\_Reg(QString ac , QString nm);//用于检查这个用户的名字和账号是不是已经存在,用于注册功能

    bool checkUser\_Name(QString name);//检查用户名是否已经存在

    bool checkUser\_Account(QString account);//检查账号是否已经存在

    bool checkUser\_OP(QString ac);//根据用户的账号返回该用户的权限

};

#endif // USER\_CRL\_H

```

```

mainwindow.h

#ifndef \_MAINWINDOWH\_H

#define \_MAINWINDOWH\_H

#include <QMainWindow>

#include <QTimer>

#include <QMouseEvent>

#include <QGraphicsDropShadowEffect>

#include <QDebug>

QT\_BEGIN\_NAMESPACE

namespace Ui { class framelessWidget; }

QT\_END\_NAMESPACE

class framelessWidget : public QWidget

{

    Q\_OBJECT

public:

    framelessWidget(QWidget \*parent = nullptr);

    ~framelessWidget();

private:

    Ui::framelessWidget \*ui;

    int cornerRadius = 20;// 窗口圆角半径

    QPoint lastPos; // 上次鼠标位置

    QWidget \*border = nullptr;// 窗口边框

    QGraphicsDropShadowEffect \*windowShadow; // 窗口阴影效果

    bool mousePressed = false;// 鼠标按下标志位

    //鼠标状态

    enum {AT\_LEFT = 1, AT\_TOP = 2,  AT\_RIGHT = 4, AT\_BOTTOM = 8,

          AT\_TOP\_LEFT = 3, AT\_TOP\_RIGHT = 6, AT\_BOTTOM\_LEFT = 9, AT\_BOTTOM\_RIGHT = 12};

    int mouseState;// 当前鼠标状态

    bool maximized = false;// 窗口最大化标志位

    void Init();// 给centralwidget添加一个mainwidget，设置遮罩及遮挡锯齿边缘的board

    void mousePressEvent(QMouseEvent \*event);// 鼠标按下事件(记录拉伸窗口或移动窗口时的起始坐标（左上角）)

    void mouseMoveEvent(QMouseEvent \*event);// 鼠标移动事件处理函数

    void mouseReleaseEvent(QMouseEvent \*event);// 鼠标释放事件处理函数

    void mouseDoubleClickEvent(QMouseEvent \*event);// 鼠标双击事件处理函数

    void resizeEvent(QResizeEvent \*event);// 窗口大小改变事件处理函数

    QRect lastGeometry;// 上次窗口位置信息

    void controlWindowScale();// 控制窗口缩放函数

private slots:

    void on\_btn\_login\_clicked();//登录按钮

    void on\_btn\_reg\_clicked();//注册按钮

    void on\_btn\_chg\_clicked();//修改密码按钮

};

#endif // \_MAINWINDOWH\_H

袁恩晗：工作量**33%** 增加车次、更改车次、添加用户、按车站、日期查询车票，按用户名查询车票、车票数据文件和用户数据文件的读取、修改和存入等功能，函数为：

class adduserdialog

{

public:

    ~adduserdialog();

    bool Gender\_Choose();//用于性别选择，其中男性返回1，女性返回0，默认是男性

    bool gender;//用户性别，1是男性，2是女性

    bool Over\_Power;//用户权限，1是管理员，2是普通用户

public slots:

    void btnclicked();//记录注册按钮的点击

};

class changeticketdialog

{

public:

    ~changeticketdialog();

    QString getiniid();//获取想要修改的车次的列车号

    QString getiniba();//获取想要修改的列车的现在的始发站

    QString getiniea();//获取想要修改的列车的现在的终点站

    QString getinibt();//获取想要修改的列车的现在的开车时间

    QString getiniet();//获取想要修改的列车的现在的到站时间

    QString getfinba();//获取想要修改的列车的改后的始发站

    QString getfinea();//获取想要修改的列车的改后的终点站

    QString getfinbt();//获取想要修改的列车的改后的开车时间

    QString getfinet();//获取想要修改的列车的改后的到站时间

    QString getfinnumber();//获取想要修改的列车的改后的票数

    QString getfinprice();//获取想要修改的列车的改后的车价

    void changebtnclicked();//修改按钮按下时触发

    int flap;//记录是否为修改按钮按下

};

class dialogaddticket

{

public:

    ~dialogaddticket();

    QString getid();//获取想要添加的列车的列车号

    QString getba();//获取想要添加的列车的始发站

    QString getea();//获取想要添加的列车的终点站

    QString getbt();//获取想要添加的列车的发车时间

    QString getet();//获取想要添加的列车的到站时间

    QString getticket();//获取想要添加的列车的车票数

    QString getprice();//获取想要添加的列车的票价

public slots:

    void saveclicked();//保存按钮按下式触发

};

class Widget

{

public:

    ~Widget();

    //将信息读入列表中

    void loadingticketdata();

    void loadinguserdata();

    //将列车信息显示在表格中

    void setticketdata(const QList<ticket>&);

    //将用户信息显示在表格中

    void setuserdata(const QList<user>&);

    //保存车票修改按钮

    void saveticket();

    //保存用户修改按钮

    void saveuser();

public slots:

    //添加用户信息按钮

    void on\_adduserButton\_clicked();

    //添加车次信息按钮

    void on\_addticketButton\_clicked();

    //列车信息菜单点击，获取当前位置

    void RightClickSlot(QPoint pos);

    //列车信息得知菜单当前的位置并删除

    void RightClickDelete(QAction \*act);

    //点击查询车票按钮

    void on\_searchticketbtn\_clicked();

    //点击查询用户按钮

    void on\_searchuserbtn\_clicked();

    //点击车票信息列表按钮

    void on\_ticketList\_clicked();

    //点击用户信息列表按钮

    void on\_userList\_clicked();

    //点击退出按钮

    void on\_Exit\_clicked();

    //点击修改列车信息是触发

    void on\_changeticketbtn\_clicked();

    //计时器

    void timerUpdate();

};

```

```

chgdialog.h

#ifndef CHGPDDIALOG\_H

#define CHGPDDIALOG\_H

#include <QWidget>

namespace Ui {

class chgpdDialog;

}

class chgpdDialog : public QWidget

{

    Q\_OBJECT

public:

    explicit chgpdDialog(QWidget \*parent = nullptr);//构造函数

    ~chgpdDialog();//析构函数

    void paintEvent(QPaintEvent \*event);//边框加一条黑边以便识别

private slots:

    void on\_btn\_back\_clicked();// 返回到mainwindow

    void on\_btn\_chg\_clicked();// 确认修改密码

protected:

    //用于鼠标拖动窗口移动

    void mousePressEvent(QMouseEvent \*event) override;

    void mouseMoveEvent(QMouseEvent \*event) override;

    void mouseReleaseEvent(QMouseEvent \*event) override;

private:

    Ui::chgpdDialog \*ui;

    QPoint reltvPos; // 相对坐标

    bool m\_bMove; // 是否移动

};

目录

[一、 系统介绍 1](#_Toc6988392)

[二、 概要设计 1](#_Toc6988393)

[三、 详细设计 1](#_Toc6988394)

[四、 总结 1](#_Toc6988395)

[1. 司居正 1](#_Toc6988396)

[2. 陈硕成 1](#_Toc6988397)

[3. 袁恩晗 2](#_Toc6988398)

（正文部分）

# 系统介绍

### 系统功能

登录界面：用户登录、管理员登录、注册用户、修改用户

用户界面：

购票查询界面：票据查询（按各标签排序）、购票、改签

个人中心界面：已购买票据查询、改签、退票

管理员界面：

车票列表、用户列表界面：按车站、日期查询车票，按用户名查询车票

增加车次、更改车次、添加用户窗口

### 2. 系统用户

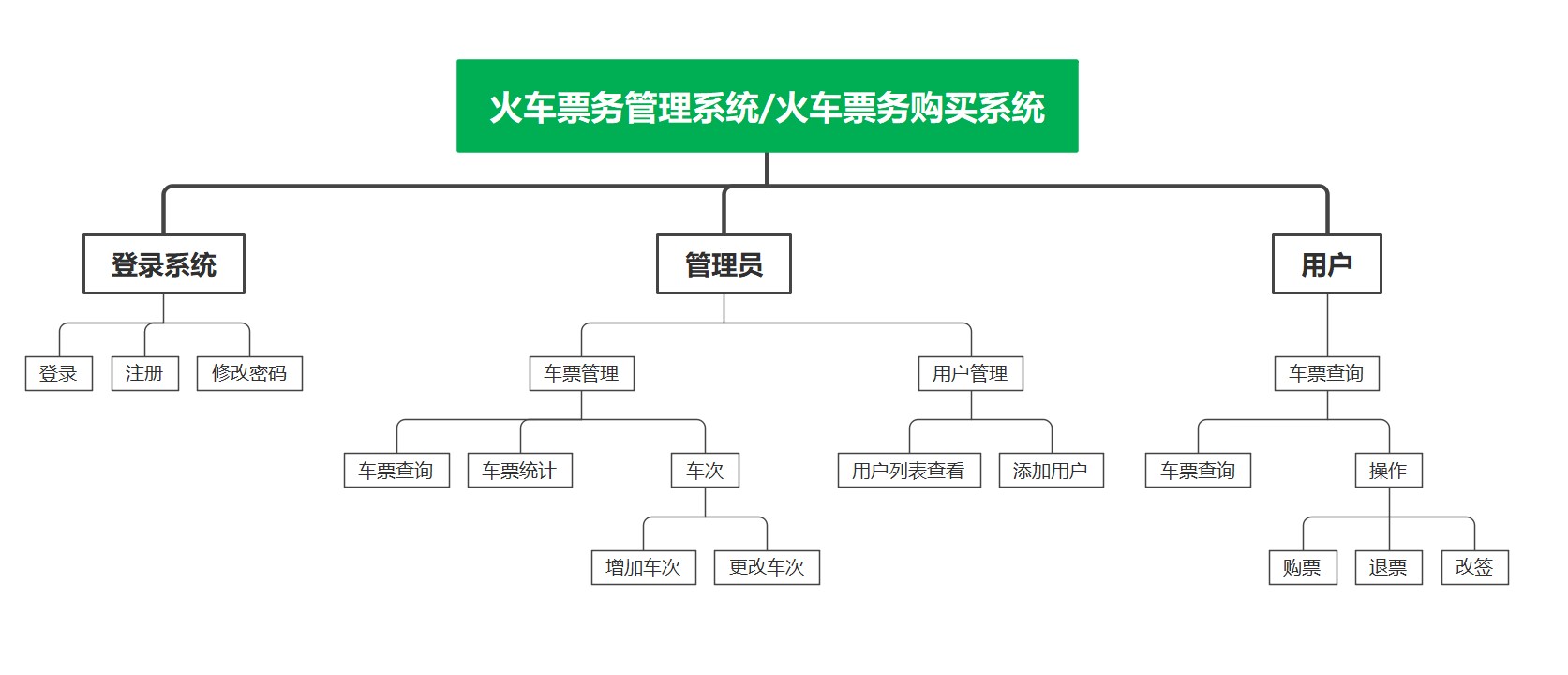
管理员、旅客

### 3. 使用场景

用于火车站购票、后台管理票据

# 概要设计

### 软件层次结构图



### 涉及数据类型

class ticket

{

public:

    ticket();

    QString id;//列车号

    QString beginpoint;//始发地

    QString endpoint;//终点站

    QString beginDay; //发车日期

    QString endDay; //到站日期

    QString begintime;//发车时间

    QString endtime;//到站时间

    int amount;//车票数

    double price;//车票价格

};

class user

{

public:

    QString account;//用户账号

    QString password;//用户密码

    bool gender;//用户性别，1是男性，2是女性

    QString name;//用户名

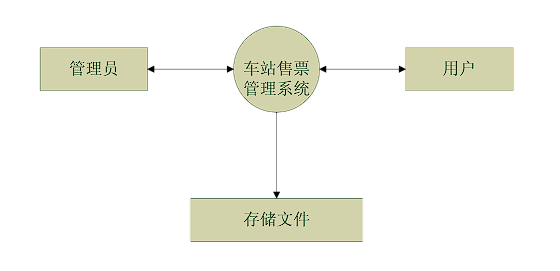
    bool Over\_Power;//用户权限，1是管理员，2是普通用户

    QList<ticket> tickets;

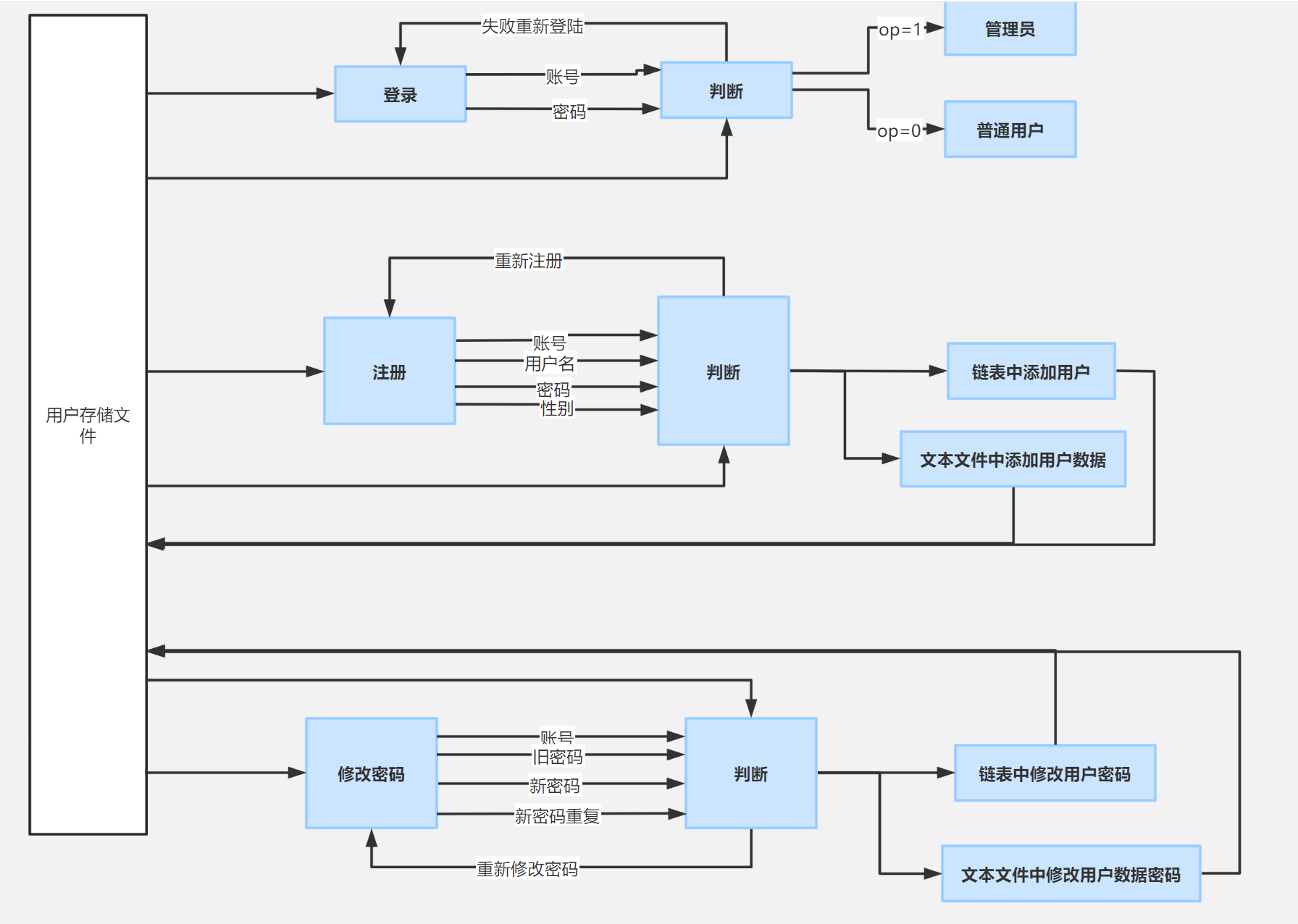
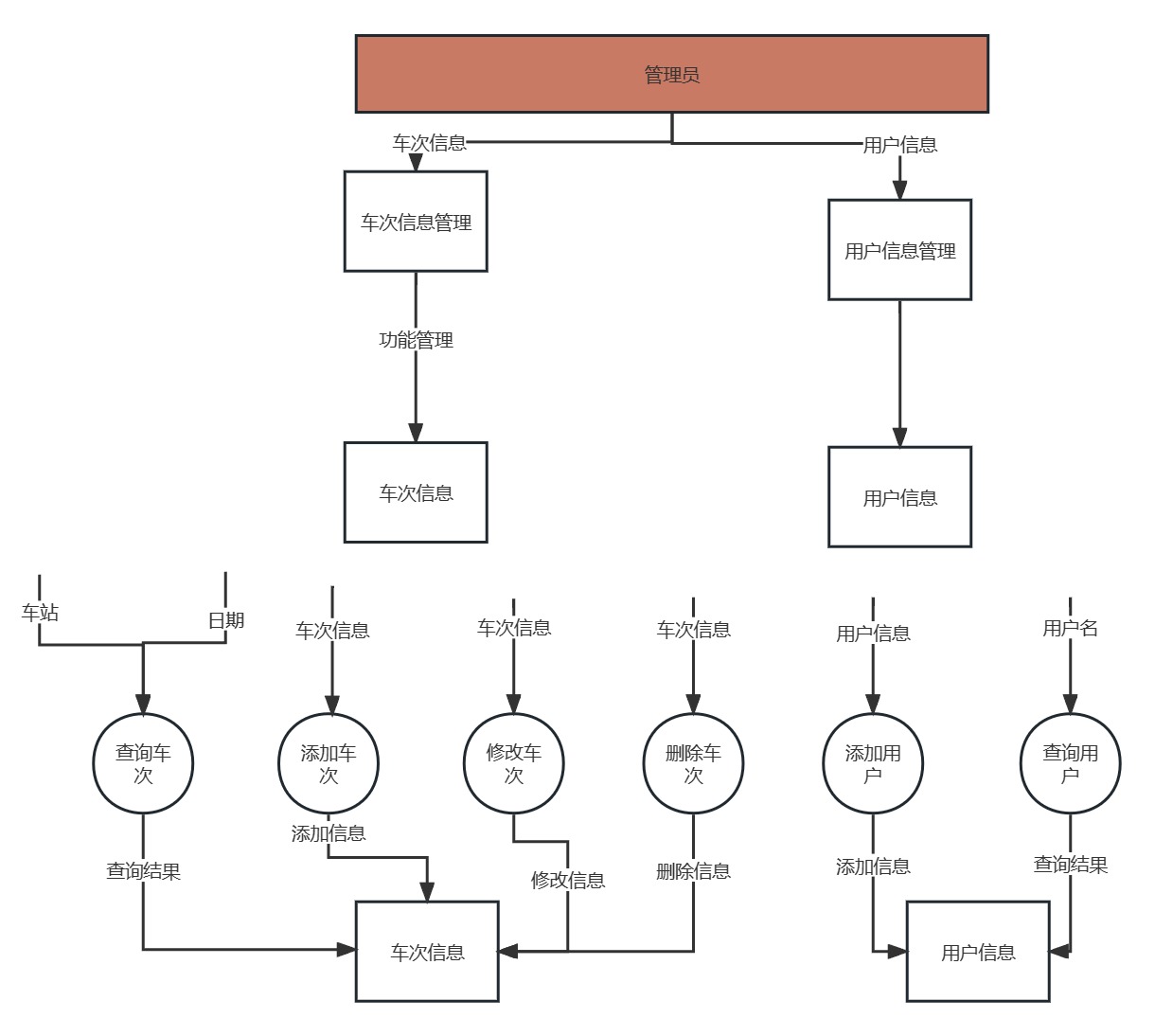
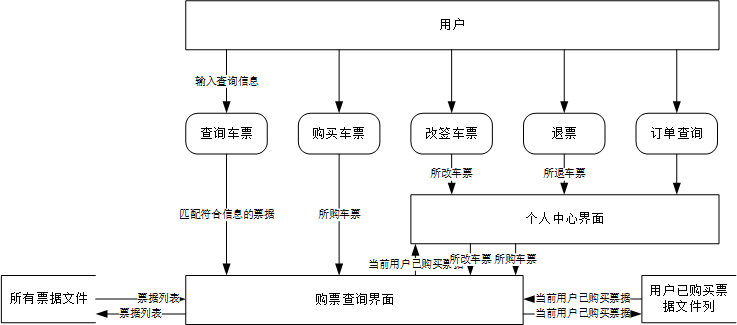
};

### 数据流图

#### 0层数据流图

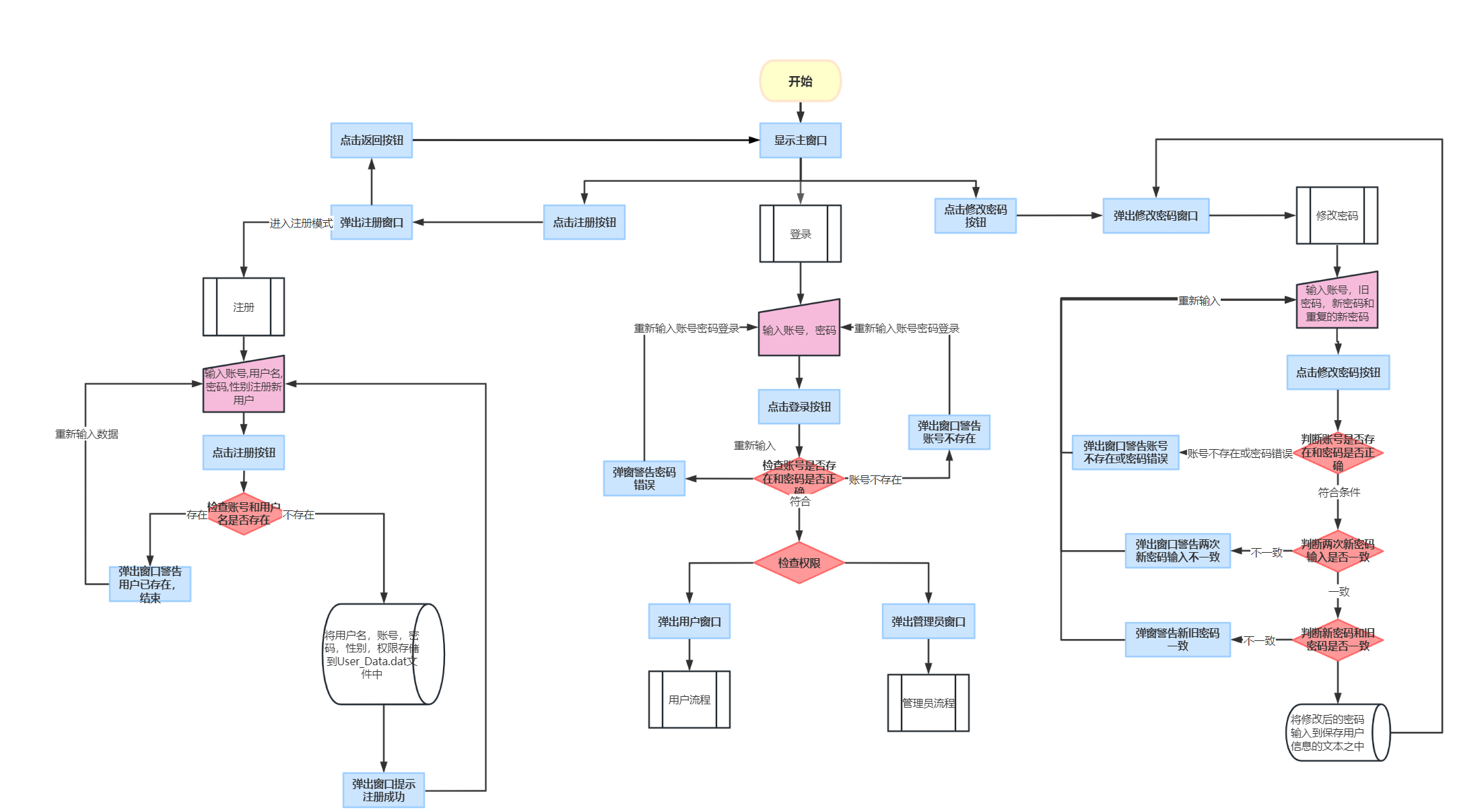


#### 1层数据流图

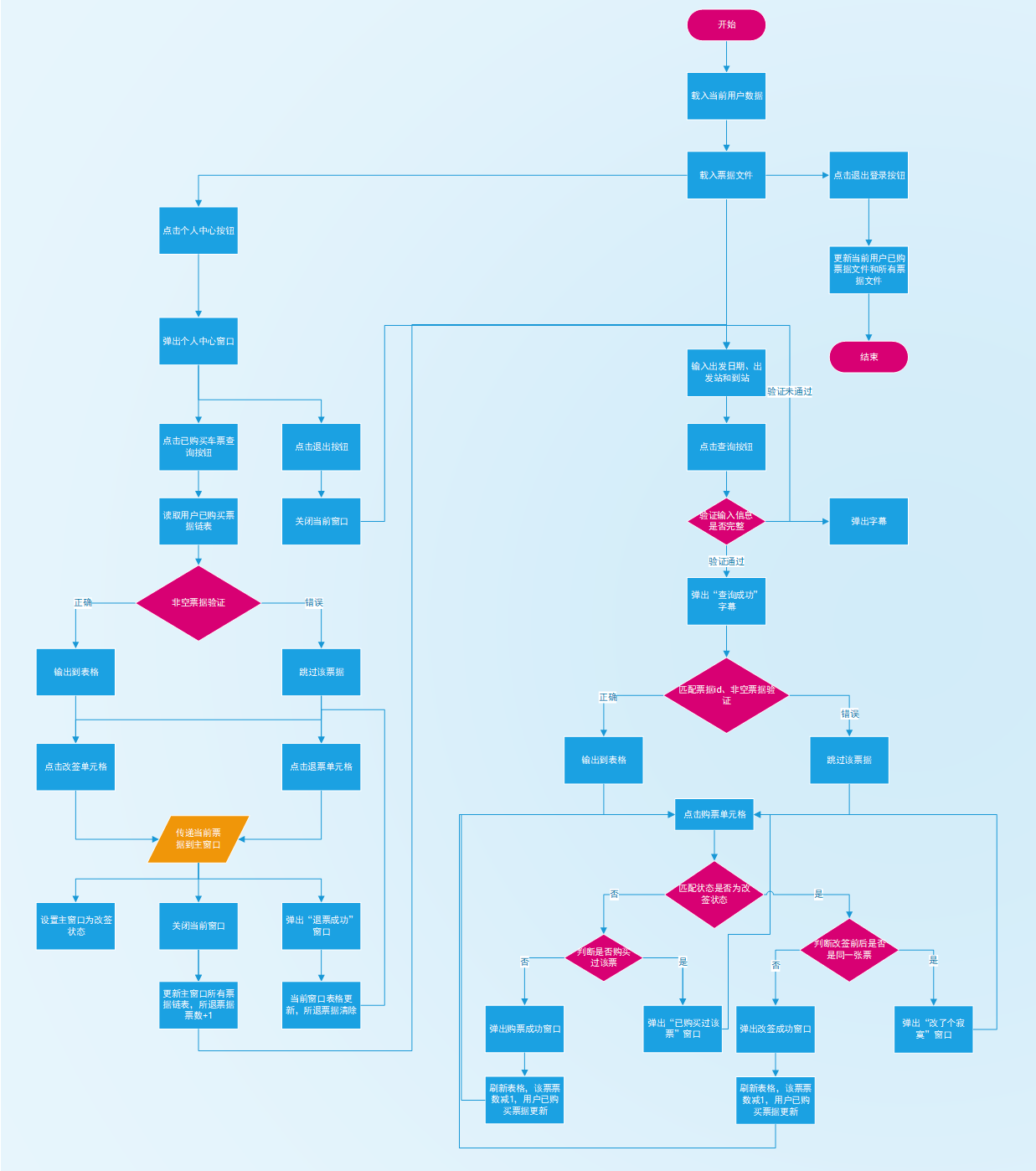


### 流程图

#### 登陆界面流程图

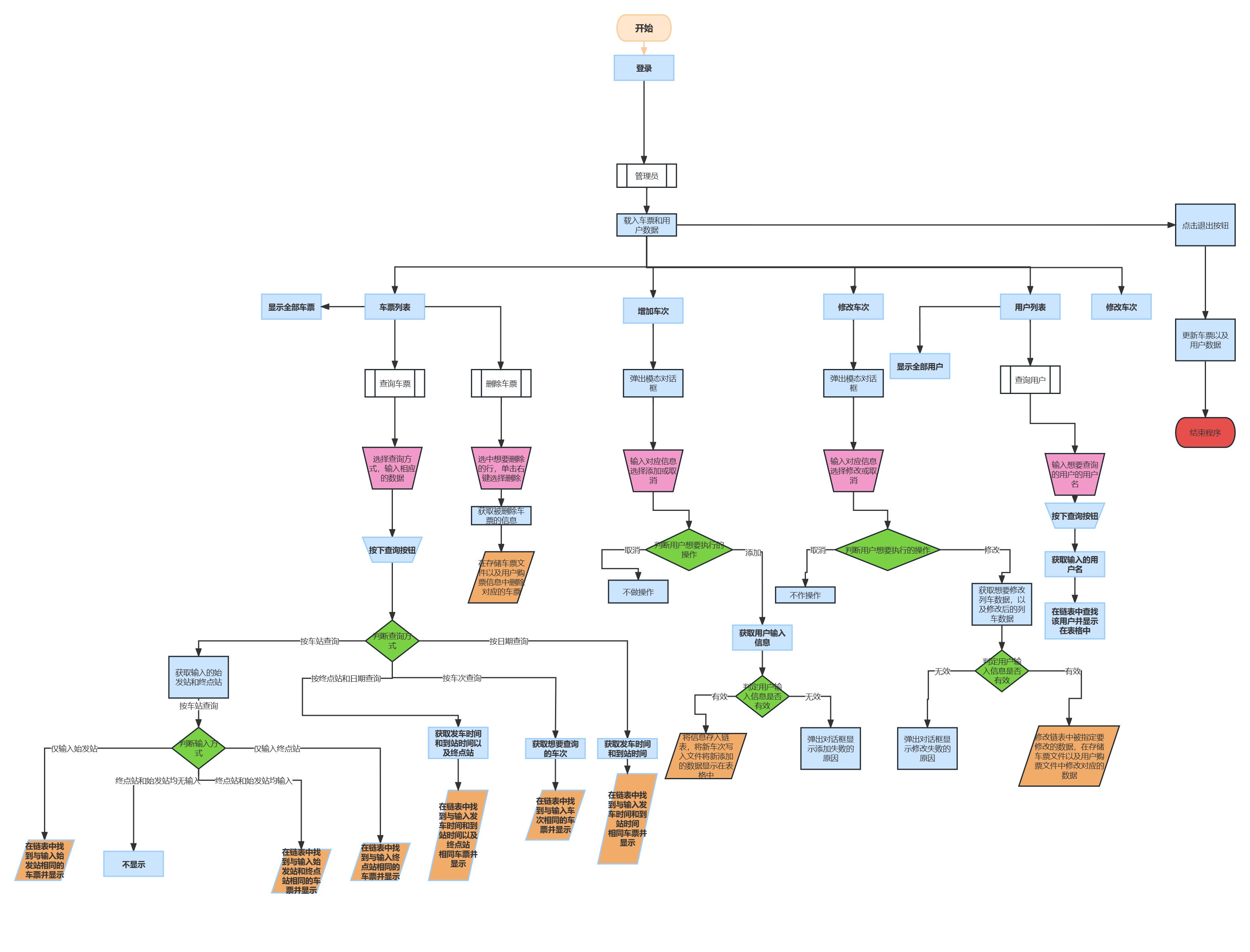


#### 用户界面流程图



#### 管理员界面流程图

# 详细设计



### 函数接口

class widget\_user

{

public:

    Widget\_User(QWidget \*parent = nullptr);

    ~Widget\_User();

    void setUser(user t); //设置当前用户

private slots:

    void on\_pushButton\_clicked(); //查询按键

    void check(); //查询

    bool read(); //读取票据文件

    void on\_tableWidget\_cellDoubleClicked(int row, int column); //双击购买/改签单元格事件

    bool refund(ticket a); //退票

    void on\_pushButton\_3\_clicked(); //个人中心按钮

    bool readUsertickets(); //读取用户已购票据

    void on\_pushButton\_4\_clicked(); //退出登录按钮

    bool save(); //保存

    void get(bool m, ticket a); //窗口间传递changingTicket

    void timerUpdate(); //时钟

    void updateTicketData(); //更新票据文件

    void tableUpdate(); //更新列表

protected:

    void closeEvent(QCloseEvent \*event); //关闭窗口动作

};

class user{

    user();//默认构造函数

    user(QString ac , QString psd , bool gd , QString nm , bool op );//带参构造函数，根据提供的数据来生成对应用户

    void setAccount(QString ac);//将用户账号设置为ac

    void setPassword(QString ps);//将用户密码设置为ps

    void setGender(bool gd);//将用户性别设置为gd

    void setName(QString nm);//将用户名设置为nm

    void setOP(bool op); //将用户权限设置为op

    user(QString a, QString b):account(a), password(b){} //合成构造函数

    bool buy(ticket a); //购票

    bool change(ticket a, ticket b); //改签

    bool refund(ticket a); //退票

    QString getid() const; //获取用户名

    QList<ticket> getTickets() const; //获取用户已购票据

    void setTickets(QList<ticket> t); //设置用户已购票据

};

class ticket

{

public:

    ticket();//构造函数

    bool operator ==(const ticket l)

    {

        /\*判断两票相同与否\*/

        if(id == l.id) return true;

        return false;

    }

};

class dialog1

{

    public:

    explicit dialog1(QWidget \*parent = nullptr);

    void setUser(user t); //设置当前用户

    ~dialog1();

signals:

    void send(bool mode, ticket a); //窗口间传递changingTicket

private slots:

    void on\_tableWidget\_2\_cellDoubleClicked(int row, int column); //双击改签/退票单元格事件

    void on\_pushButton\_clicked(); //已购票据查询按钮

    void on\_pushButton\_2\_clicked(); //退出按钮

}

class adduserdialog

{

public:

    ~adduserdialog();

    bool Gender\_Choose();//用于性别选择，其中男性返回1，女性返回0，默认是男性

    bool gender;//用户性别，1是男性，2是女性

    bool Over\_Power;//用户权限，1是管理员，2是普通用户

public slots:

    void btnclicked();//记录注册按钮的点击

};

class changeticketdialog

{

public:

    ~changeticketdialog();

    QString getiniid();//获取想要修改的车次的列车号

    QString getiniba();//获取想要修改的列车的现在的始发站

    QString getiniea();//获取想要修改的列车的现在的终点站

    QString getinibt();//获取想要修改的列车的现在的开车时间

    QString getiniet();//获取想要修改的列车的现在的到站时间

    QString getfinba();//获取想要修改的列车的改后的始发站

    QString getfinea();//获取想要修改的列车的改后的终点站

    QString getfinbt();//获取想要修改的列车的改后的开车时间

    QString getfinet();//获取想要修改的列车的改后的到站时间

    QString getfinnumber();//获取想要修改的列车的改后的票数

    QString getfinprice();//获取想要修改的列车的改后的车价

    void changebtnclicked();//修改按钮按下时触发

    int flap;//记录是否为修改按钮按下

};

class dialogaddticket

{

public:

    ~dialogaddticket();

    QString getid();//获取想要添加的列车的列车号

    QString getba();//获取想要添加的列车的始发站

    QString getea();//获取想要添加的列车的终点站

    QString getbt();//获取想要添加的列车的发车时间

    QString getet();//获取想要添加的列车的到站时间

    QString getticket();//获取想要添加的列车的车票数

    QString getprice();//获取想要添加的列车的票价

public slots:

    void saveclicked();//保存按钮按下式触发

};

class Widget

{

public:

    ~Widget();

    //将信息读入列表中

    void loadingticketdata();

    void loadinguserdata();

    //将列车信息显示在表格中

    void setticketdata(const QList<ticket>&);

    //将用户信息显示在表格中

    void setuserdata(const QList<user>&);

    //保存车票修改按钮

    void saveticket();

    //保存用户修改按钮

    void saveuser();

public slots:

    //添加用户信息按钮

    void on\_adduserButton\_clicked();

    //添加车次信息按钮

    void on\_addticketButton\_clicked();

    //列车信息菜单点击，获取当前位置

    void RightClickSlot(QPoint pos);

    //列车信息得知菜单当前的位置并删除

    void RightClickDelete(QAction \*act);

    //点击查询车票按钮

    void on\_searchticketbtn\_clicked();

    //点击查询用户按钮

    void on\_searchuserbtn\_clicked();

    //点击车票信息列表按钮

    void on\_ticketList\_clicked();

    //点击用户信息列表按钮

    void on\_userList\_clicked();

    //点击退出按钮

    void on\_Exit\_clicked();

    //点击修改列车信息是触发

    void on\_changeticketbtn\_clicked();

    //计时器

    void timerUpdate();

};

### 调用关系

chgdialog.h

#ifndef CHGPDDIALOG\_H

#define CHGPDDIALOG\_H

#include <QWidget>

namespace Ui {

class chgpdDialog;

}

class chgpdDialog : public QWidget

{

    Q\_OBJECT

public:

    explicit chgpdDialog(QWidget \*parent = nullptr);//构造函数

    ~chgpdDialog();//析构函数

    void paintEvent(QPaintEvent \*event);//边框加一条黑边以便识别

private slots:

    void on\_btn\_back\_clicked();// 返回到mainwindow

    void on\_btn\_chg\_clicked();// 确认修改密码

protected:

    //用于鼠标拖动窗口移动

    void mousePressEvent(QMouseEvent \*event) override;

    void mouseMoveEvent(QMouseEvent \*event) override;

    void mouseReleaseEvent(QMouseEvent \*event) override;

private:

    Ui::chgpdDialog \*ui;

    QPoint reltvPos; // 相对坐标

    bool m\_bMove; // 是否移动

};

#endif // CHGPDDIALOG\_H

```

```

logindialog.h

#ifndef LOGINDIALOG\_H

#define LOGINDIALOG\_H

#include <QString>

class LoginDialog

{

public:

    bool LoginUser(QString account , QString password);//用于进行登录的操作

public:

    LoginDialog();//默认构造函数

};

#endif // LOGINDIALOG\_H

```

```

md5.h

#ifndef MD5\_H

#define MD5\_H

#include <string>

using std::string;

class md5

{

public:

    //构造函数，用于初始化类的成员变量

    md5();

    /\*MD5主循环函数，用于处理每个512位的分组。

    根据索引 i 的不同值，选择不同的运算方式，并更新临时变量。\*/

    void mainLoop(unsigned int M[]);

    unsigned int\* add(string str);//填充函数，将输入的字符串填充为64字节的整数倍，并添加长度信息。

    string changeHex(int a);//将整数转换为16进制字符串表示。

    /\*计算给定字符串的MD5哈希值。初始化临时变量并调用 add 函数对输入字符串进行填充，

    然后通过 mainLoop 处理每个512位分组，最后将结果转换为16进制字符串形式并返回。\*/

    string getMD5(string source);

};

#endif // MD5\_H

```

```

regdialog.h

#ifndef REGDIALOG\_H

#define REGDIALOG\_H

#include <QWidget>

namespace Ui {

class RegDialog;

}

class RegDialog : public QWidget

{

    Q\_OBJECT

public:

    explicit RegDialog(QWidget \*parent = nullptr);//默认构造函数

    ~RegDialog();//默认析构函数

    bool Gender\_Choose();//用于性别选择，其中男性返回1，女性返回0，默认是男性

    void paintEvent(QPaintEvent \*event);//边框加一条黑边以便识别

private slots:

    void on\_Btn\_Back\_clicked();//返回mainwindow

    void on\_Btn\_Reg\_clicked();//注册用户

//用于鼠标拖动无边框窗口移动

protected:

    void mousePressEvent(QMouseEvent \*event) override;// 鼠标按下事件处理函数

    void mouseMoveEvent(QMouseEvent \*event) override; // 鼠标移动事件处理函数

    void mouseReleaseEvent(QMouseEvent \*event) override;// 鼠标释放事件处理函数

private:

    Ui::RegDialog \*ui;

    QPoint reltvPos; // 相对坐标

    bool m\_bMove; // 是否移动

};

#endif // REGDIALOG\_H

```

```

user\_crl.h

#ifndef USER\_CRL\_H

#define USER\_CRL\_H

#include <user.h>

#include <QList>

#include <QString>

class user\_Crl

{

    QList<user> userList;//创建一个用户链表，用于存储每一个用户的信息

public:

    user\_Crl();//默认构造函数，用于用户链表的初始化，将文件中的内容导入链表

    bool AddUser(QString ac , QString psd , bool gd , QString nm , bool op);//添加一个新用户，用于注册功能

    bool ChgUser(QString ac , QString oldpd , QString newpd, QString renewpd);//修改用户信息，用于修改密码

    bool LoginUser(QString ac , QString psd );//用于用户登录

    bool checkUser\_Password(QString account , QString password);//验证用户是否存在以及密码是否正确，用于登录功能和修改密码的检验功能

    bool checkUser\_Reg(QString ac , QString nm);//用于检查这个用户的名字和账号是不是已经存在,用于注册功能

    bool checkUser\_Name(QString name);//检查用户名是否已经存在

    bool checkUser\_Account(QString account);//检查账号是否已经存在

    bool checkUser\_OP(QString ac);//根据用户的账号返回该用户的权限

};

#endif // USER\_CRL\_H

```

```

mainwindow.h

#ifndef \_MAINWINDOWH\_H

#define \_MAINWINDOWH\_H

#include <QMainWindow>

#include <QTimer>

#include <QMouseEvent>

#include <QGraphicsDropShadowEffect>

#include <QDebug>

QT\_BEGIN\_NAMESPACE

namespace Ui { class framelessWidget; }

QT\_END\_NAMESPACE

class framelessWidget : public QWidget

{

    Q\_OBJECT

public:

    framelessWidget(QWidget \*parent = nullptr);

    ~framelessWidget();

private:

    Ui::framelessWidget \*ui;

    int cornerRadius = 20;// 窗口圆角半径

    QPoint lastPos; // 上次鼠标位置

    QWidget \*border = nullptr;// 窗口边框

    QGraphicsDropShadowEffect \*windowShadow; // 窗口阴影效果

    bool mousePressed = false;// 鼠标按下标志位

    //鼠标状态

    enum {AT\_LEFT = 1, AT\_TOP = 2,  AT\_RIGHT = 4, AT\_BOTTOM = 8,

          AT\_TOP\_LEFT = 3, AT\_TOP\_RIGHT = 6, AT\_BOTTOM\_LEFT = 9, AT\_BOTTOM\_RIGHT = 12};

    int mouseState;// 当前鼠标状态

    bool maximized = false;// 窗口最大化标志位

    void Init();// 给centralwidget添加一个mainwidget，设置遮罩及遮挡锯齿边缘的board

    void mousePressEvent(QMouseEvent \*event);// 鼠标按下事件(记录拉伸窗口或移动窗口时的起始坐标（左上角）)

    void mouseMoveEvent(QMouseEvent \*event);// 鼠标移动事件处理函数

    void mouseReleaseEvent(QMouseEvent \*event);// 鼠标释放事件处理函数

    void mouseDoubleClickEvent(QMouseEvent \*event);// 鼠标双击事件处理函数

    void resizeEvent(QResizeEvent \*event);// 窗口大小改变事件处理函数

    QRect lastGeometry;// 上次窗口位置信息

    void controlWindowScale();// 控制窗口缩放函数

private slots:

    void on\_btn\_login\_clicked();//登录按钮

    void on\_btn\_reg\_clicked();//注册按钮

    void on\_btn\_chg\_clicked();//修改密码按钮

};

#endif // \_MAINWINDOWH\_H

### 关键算法

1. 增

void Widget::on\_addticketButton\_clicked()

{

    //添加列车信息按钮功能的实现

    dialogaddticket\* da=new dialogaddticket;//创建一个新的添加列车对话框

    da->setModal(true);

    da->show();//显示对话框

    da->exec();//设置为模态

    //判断添加按钮是否被按下

    if(da->flap==1)

    {

        QString id=da->getid();//获取新车次的列车号

        QString ba=da->getba();//获取新车次的始发站

        QString ea=da->getea();//获取新车次的终点站

        QString bt=da->getbt();//获取新车次的发车时间

        QString et=da->getet();//获取新车次的到站时间

        QString t=da->getticket();//获取新车次的车票数

        QString pr=da->getprice();//获取新车次的票价

        delete da;//释放

        //判断是否有信息不全

        if(id.isEmpty()||ba.isEmpty()||ea.isEmpty()||bt.isEmpty()||et.isEmpty()||t.isEmpty())

        {

            QMessageBox::warning(this,"Warning","信息不全，无法添加！！！");//如果信息不全则警告，不予添加

        }

        else

        {

            //如果信息完全则尝试添加

            //判断有信息冲突

            QList<ticket>::iterator it=ticketlist.begin();

            for(;it!=ticketlist.end();it++)

            {

                if(it->id==id&&it->beginpoint==ba&&it->endpoint==ea&&it->begintime==bt)//这里仅判断是否有同一列车相同始发站和终点站同时开车的情况

                {

                    QMessageBox::warning(this,"Warning","信息冲突，无法添加！！！");//如果信息有冲突，则警告，不予添加

                    break;

                }

            }

            //如果信息完全无冲突，则添加该车次

            if(it==ticketlist.end())

            {

                ticket newt;//创建新的车票对象

                newt.id=id;//设置列车号

                newt.beginpoint=ba;//设置始发站

                newt.endpoint=ea;//设置终点站

                newt.begintime=bt;//设置开车时间

                newt.endtime=et;//设置到站时间

                newt.amount=t.toInt();//设置车票数

                newt.price=pr.toDouble();//设置票价

                ticketlist.push\_back(newt);//加入车次列表中

                //根据发车时间排序

                ticketlist\_sort(0,ticketlist.size()-1);

                setticketdata(ticketlist);

            }

        }

    }

    saveticket();

}

1. 删

//在列车信息列表右键单击时判断是否删除

void Widget::RightClickDelete(QAction \*act)

{

    //弹出提示框，看是否删除数据

    QMessageBox message(QMessageBox::NoIcon, QString::fromLocal8Bit("warning"),

                                QString::fromLocal8Bit("Whether to delete the data of this ticket"),

    QMessageBox::Yes | QMessageBox::No, NULL);

    //如确认删除

    if(message.exec() == QMessageBox::Yes)

    {

        if(act->text() == QString::fromLocal8Bit("delete"))   //看选中了删除这个菜单

        {

            QString id=ui->ticketWidget->model()->index(iDeletcRow,0).data().toString();//获取选中的车次的列车号

            QString ba=ui->ticketWidget->model()->index(iDeletcRow,1).data().toString();//获取选中的车次的始发站

            QString ea=ui->ticketWidget->model()->index(iDeletcRow,2).data().toString();//获取选中的车次的终点站

            QString bt=ui->ticketWidget->model()->index(iDeletcRow,3).data().toString();//获取选中的车次的发车时间

            QString et=ui->ticketWidget->model()->index(iDeletcRow,4).data().toString();//获取选中的车次的到站时间

            //在列车信息列表中查找该车票

            for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

            {

                //找到后删除

                if(it->id==id&&it->beginpoint==ba&&it->endpoint==ea&&it->begintime==bt&&it->endtime==et)

                {

                    ticketlist.erase(it);

                    break;

                }

            }

            //删除用户已经购买车票的信息

            for(QList<user>::iterator it=userlist.begin();it!=userlist.end();it++)

            {

                for(QList<ticket>::iterator it1=it->tickets.begin();it1!=it->tickets.end();it1++)

                {

                    //找到后删除

                    if(it1->id==id&&it1->beginpoint==ba&&it1->endpoint==ea&&it1->begintime==bt&&it1->endtime==et)

                    {

                        it->tickets.erase(it1);

                        break;

                    }

                }

            }

            ui->ticketWidget->removeRow(iDeletcRow);  //删除表格中信息

        }

    }

    saveticket();

    saveuser();

}

1. 改

void Widget::on\_changeticketbtn\_clicked()

{

    changeticketdialog\* ct=new changeticketdialog;//创建一个新的对话框

    ct->setModal(true);

    ct->show();//显示对话框

    ct->exec();//设置为模态

    //判断修改按钮是否被按下

    if(ct->flap==1)

    {

        QString iniid=ct->getiniid();//获取想要修改的车次的列车号

        QString iniba=ct->getiniba();//获取想要修改的列车的现在的始发站

        QString iniea=ct->getiniea();//获取想要修改的列车的现在的终点站

        QString inibt=ct->getinibt();//获取想要修改的列车的现在的开车时间

        QString iniet=ct->getiniet();//获取想要修改的列车的现在的到站时间

        QString finba=ct->getfinba();//获取想要修改的列车的改后的始发站

        QString finea=ct->getfinea();//获取想要修改的列车的改后的终点站

        QString finbt=ct->getfinbt();//获取想要修改的列车的改后的开车时间

        QString finet=ct->getfinet();//获取想要修改的列车的改后的到站时间

        QString finnumber=ct->getfinnumber();//获取想要修改的列车的改后的车票数

        QString finprice=ct->getfinprice();//获取想要修改的列车的改后的票价

        int temp=0;

        //遍历查找要修改的车次

        for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

        {

            //找到要修改的车票信息

            if(it->id==iniid&&it->beginpoint==iniba&&it->endpoint==iniea&&it->begintime==inibt&&it->endtime==iniet)

            {

                temp=1;

                int flap=0;//记录是否可以修改

                //检查信息是否有冲突

                for(QList<ticket>::iterator it1=ticketlist.begin();it1!=ticketlist.end();it1++)

                {

                    //判断车次信息是否有冲突

                    if(it1->id==iniid&&it1->beginpoint==finba&&it1->endpoint==finea&&it1->begintime==finbt&&it1->endpoint==finet&&QString::number(it1->amount)==finnumber&&QString::number(it1->price)==finprice)

                    {

                            QMessageBox::warning(this,"Warning","信息冲突，无法修改！！！");//有冲突则警告，不予修改

                            flap=1;

                            break;

                    }

                }

                //如果不能修改

                if(flap)

                {

                    break;

                }

                //可以修改

                else

                {

                    it->beginpoint=finba;//更改始发站

                    it->endpoint=finea;//更改终点站

                    it->begintime=finbt;//更改发车时间

                    it->endtime=finet;//更改到站时间

                    it->amount=finnumber.toDouble();//更改车票数

                    it->price=finprice.toDouble();//更改车票价格

                    //修改用户已经购买车票的信息

                    for(QList<user>::iterator it1=userlist.begin();it1!=userlist.end();it1++)

                    {

                        for(QList<ticket>::iterator it2=it1->tickets.begin();it2!=it1->tickets.end();it2++)

                        {

                            //找到后修改

                            if(it2->id==iniid&&it2->beginpoint==iniba&&it2->endpoint==iniea&&it2->begintime==inibt&&it2->endtime==iniet)

                            {

                                it2->beginpoint=finba;//更改始发站

                                it2->endpoint=finea;//更改终点站

                                it2->begintime=finbt;//更改发车时间

                                it2->endtime=finet;//更改到站时间

                                it2->price=finprice.toDouble();//更改车票价格

                                break;

                            }

                        }

                    }

                    QMessageBox::information(this,"提示","修改成功");//提示

                    //根据发车时间排序

                    ticketlist\_sort(0,ticketlist.size()-1);

                    setticketdata(ticketlist);//显示更改后车票的信息

                    setuserdata(userlist);//显示更改后的用户

                    //保存修改

                    saveticket();

                    saveuser();

                    break;

                }

            }

        }

        if(temp==0)

        {

            QMessageBox::warning(this,"Warning","查询不到想要修改的列车，无法修改！！！");//查询不到想要修改的列车则报警，不予修改

        }

    }

}

1. 查
2. void Widget\_User::check()
3. {
4. /\*查询票据实现\*/
5. //获取查询信息
6. QString b = ui->lineEdit->text();
7. QString c = ui->lineEdit\_2->text();
8. QString d = ui->dateEdit->date().toString("yyyy-MM-dd");
9. if(b == "" || c == "")
10. { //若有信息为空
11. ui->label\_5->setText("查询失败：信息不完整"); //报错
12. }
13. else
14. {
15. int i = 0;
16. int row = 0, column = 0;
17. while(i < tickets.size() - 1)
18. { //循环打印票据到表格
19. if(d == tickets[i].beginDay && b == tickets[i].beginpoint && c == tickets[i].endpoint)
20. {
21. ui->tableWidget->insertRow(0); //插入新行
22. column = 0;
23. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(tickets[i].id));
24. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(tickets[i].beginpoint));
25. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(tickets[i].endpoint));
26. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(tickets[i].begintime));
27. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(tickets[i].endtime));
28. QString t1; //临时字符串，存储历时
29. //字符串转化为QDate和QTime
30. QDate b = QDate::fromString(tickets[i].beginDay, "yyyy-MM-dd");
31. QDate e = QDate::fromString(tickets[i].endDay, "yyyy-MM-dd");
32. QTime bt = QTime::fromString(tickets[i].begintime, "hh:mm");
33. QTime et = QTime::fromString(tickets[i].endtime, "hh:mm");
34. t1 = QString::fromStdString(std::to\_string(b.daysTo(e)) +"天"+
35. std::to\_string(et.hour() - bt.hour()) +"时"+
36. std::to\_string(et.minute() - bt.minute()) +"分"); //历时计算
37. ui->tableWidget->setItem(row, column++, new QTableWidgetItem(t1));
38. ui->tableWidget->setItem(row, column++, new QTableWidgetItem
39. (QString::fromStdString(std::to\_string(tickets[i].amount))));
40. if(!mode)
41. {
42. ui->tableWidget->setItem(row, column, new QTableWidgetItem("购买"));
43. }
44. else
45. { //若为改签状态
46. ui->tableWidget->setItem(row, column, new QTableWidgetItem("改签")); //打印改签
47. }
48. ui->tableWidget->setItem(row, ++column, new QTableWidgetItem(QString::number(tickets[i].price, 'f', 2))); //保留两位小数显示票价
49. }
50. i++;
51. }
52. ui->label\_5->setText("查询成功!");
53. }
54. }

5. 筛选

//查询车票按钮功能的实现

void Widget::on\_searchticketbtn\_clicked()

{

    //清空查找列车表格数据

    for(int row = ui->searchticketWidget->rowCount() - 1;row >= 0; row--)

    {

        ui->searchticketWidget->removeRow(row);

    }

    //将表格转换到查找列车表格

    ui->stackedWidget2->setCurrentIndex(1);

    //判断查询方式

    //判断是否为按车站查询

    if(ui->addressButton->isChecked())

    {

        QString ba,ea;//创建存储始发站和终点站的字符串

        ba=ui->baEdit->text();//获取查询到始发站

        ea=ui->eaEdit->text();//获取查询的终点站

        //如果始发站和终点站均有输入

        if(!ba.isEmpty()&&!ea.isEmpty())

        {

            //在列表中查询是否有符合条件的车次信息

            for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

            {

                //找到后就显示该车次的信息

                if(it->beginpoint==ba&&it->endpoint==ea)

                {

                    print(it);

                }

            }

        }

        //如果仅有始发站输入

        else if(!ba.isEmpty()&&ea.isEmpty())

        {

            //在列表中查询是否有符合条件的车次信息

            for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

            {

                //找到后就显示该车次的信息

                if(it->beginpoint==ba)

                {

                    print(it);

                }

            }

        }

        //如果仅有终点站有输入

        else if(ba.isEmpty()&&!ea.isEmpty())

        {

            //在列表中查询是否有符合条件的车次信息

            for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

            {

                //找到后就显示该车次的信息

                if(it->endpoint==ea)

                {

                    print(it);

                }

            }

        }

        //如果始发站和终点站均无输入

        else

        {

            ;

        }

    }

    //判断是否为按时间查询

    if(ui->timeButton->isChecked())

    {

        QString strbt=ui->btEdit->dateTime().toString("yyyy-MM-dd");//创建存储开车时间的字符串

        QString stret=ui->etEdit->dateTime().toString("yyyy-MM-dd");//创建存储到站时间的字符串

        //遍历列表查找是否有符合条件的车次

        for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

        {

            QString rbt=it->begintime.mid(0,10);//存储被查找车次的开车时间

            QString ret=it->endtime.mid(0,10);//存储被查找车次的到站时间

            //如果符合条件

            if(rbt==strbt&&ret==stret)

            {

                print(it);

            }

         }

    }

    //判断是否为按车次查询

    if(ui->IdButton->isChecked())

    {

        QString searchid=ui->idEdit->text();//创建存储开车时间的字符串

        //遍历列表查找是否有符合条件的车次

        for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

        {

            //如果符合条件

            if(it->id==searchid)

            {

                print(it);

            }

         }

    }

    //判断是否为按终点和日期查询

    if(ui->EaAndTimeButton->isChecked())

    {

        QString ea=ui->eaEdit->text();//获取查询的终点站

        QString strbt=ui->btEdit->dateTime().toString("yyyy-MM-dd");//创建存储开车时间的字符串

        QString stret=ui->etEdit->dateTime().toString("yyyy-MM-dd");//创建存储到站时间的字符串

        //遍历列表查找是否有符合条件的车次

        for(QList<ticket>::iterator it=ticketlist.begin();it!=ticketlist.end();it++)

        {

            QString rbt=it->begintime.mid(0,10);//存储被查找车次的开车时间

            QString ret=it->endtime.mid(0,10);//存储被查找车次的到站时间

            //如果符合条件

            if(it->endpoint==ea&&rbt==strbt&&ret==stret)

            {

                print(it);

            }

         }

    }

}

### 程序亮点

1. 图形化ui界面
2. md5加密用户密码
3. 即时时钟
4. 通过单元格信号触发订票系列事件
5. 可通过日历选择日期

6. 用户可查看车次历时

7. 管理员右击删除票据，更加方便

8. 用户可单击表头排序票据

# 总结

### 登录主界面开发-司居正

1. 所做工作：
   * 实现了登录主界面
   * 实现了注册界面
   * 实现了修改密码界面
   * 实现了用户数据的读取、存入、修改
   * 用md5加密算法实现了用户密码的加密
   * 在文件中加入了qt-material-widgets-master静态库，使ui界面更简洁人性化
   * 视频录制
2. 开发中遇到的问题
   * 已解决
     + 窗口无边框
     + 点击窗口空白区域可以拖动窗口
     + 登录注册修改密码遇到的多种情况进行分情况讨论
     + 在注册时User\_Data.dat文件内的数据自动更新
     + 密码在User\_Data.dat中的密码加密，避免直接被人盗取
   * 未解决
     + 密码加密不可逆，如果使用非对称可逆加密SRA加密算法更具有科学性
     + 用户列表的查找方式时间复杂度太大，如果将用户列表排序后用二分查找会减少查找时间
3. 收获和想法

我的收获：通过本次开发，我学会了QT软件的基本使用，学会了静态库基本使用，熟悉了一个软件从设计，开发到测试和报告的整个流程，进一步掌握了C/C++语言基础语法的使用，同时也明白了一些前端设计的基础，对C/C++语言语法的掌握更进了一步。

在撰写实验报告的过程中，我了解了软件工程的一些基础知识，例如软件流程图，软件层次图，数据流图等概念，学会了从整体布局，局部分析完成的思维逻辑，受益匪浅。同时我也学会了markdown的基础语法，能够使文本的编 写更加规矩整洁，提高可读性。

在团队合作过程中，由于大家初次团队合作，对彼此的了解不算太深，一开始时问题百出，分工的不明确，功能漏洞百出。在不断的磨合中，我们对彼此的了解更加深刻，同时团队的合作也更加紧密。我与两位组员之间沟通到位， 遇到问题及时沟通解决，彼此主动承担任务，对自身的任务压力毫无推脱，大家一心为了好好完成和打磨大作业，给老师，队员和自己一个完美的答复。同时我们这次开发借助了github，对于初次接触github的我们来说，各种操作对大家来说都是从未接触的，导致前四五天操作github的时候总是操作失误，导致文件总是需要进行git reset --hard ...，导致现在看见这个代码就已经有反应了。但是经过几天的磨合后，我们团队就已经体会到在github合作的好处了。我认为github不仅为团队提高了效率，在点开主页看到一条一条的commit之后，内心的自豪感也是油然而生，这是我们一点一点从无到有搭建起来的一座小木屋，是我们近三个周共同努力的结果。

我的想法：非常感谢老师能够提供这次难得的机会，用一段比较长的时间来提高我自己的代码能力，让我提高自己的代码能力。在这次开发过程中，我对C/C++语言的熟练程度得到了提高，同时参与了整个管理系统的制作过程。同 时我也要感谢我的各位组员，在最开始我邀请两位一起成组的时候，大家在合作上还是很不熟悉彼 此，但是在几周的合作下，我们已经成功熟悉了彼此，遇到问题一起集思广益去解决问题，用集体的力量去解决问题。同时这次大作业也让我对自己认知有了更深的了解，一个人对于自己的能力应该有自信，要有面对困难，解决困难的勇气，如果不是经历过整个大作业的过程，过去的我也不会相信自己会在短短三周时间里学会qt、github、markdown、密码加密等基础知识，并有机会将这些知识转化为在大作业里面的应用。在经历了这次的大作业后，我会更加有勇气的去面对未来的困难，并且用乐观积极的态度解决问题，因为我相信我可以，我有能力去面对。最后，再次感谢老师和两位和我一起努力的队友。

### 用户界面开发-陈硕成

1. 所做工作：

* 实现了用户界面
* 实现了个人中心界面
* 实现了用户购买车票文件的读取和存入
* 实现了现有票据文件的读取和存入
* 报告排版、开发文档搭建、视频剪辑

1. 开发中遇到的问题：

* 已解决
  + 窗口间传值
  + 文本文件编码
  + 同步更新用户信息
  + 表头排序功能
  + 关闭窗口事件触发自动保存
* 未解决
  + 经停站显示
  + 票务转让

1. 收获和想法

我的收获：通过本次开发，我在步步摸索中学会了Qt5(C++)的开发模式，掌握了基本的ui设计，在应用中巩固了面向对象技术的要点，扎实了文件输入/输出、数据类型、函数调用等程序设计的基本功  
在撰写报告的过程中，我了解了软件工程中层次图、流程图、数据流图的画法与作用，体会到软件设计框架对于整体设计思路的益处

我的想法：对于程序设计大作业来说，管理系统可谓是老生常谈，实现其功能并不难，关键在于对于功能的封装，以及呈现在使用者面前的效果即对于使用者的友好性，以及操作的直观性、便捷性，这就需要一个良好的ui设计。另一方面，“金玉其外，败絮其中”的现象经常会发生在过于注重ui的程序，其经常会出现一些运行时错误，对于极端情况下的操作缺乏对策，因此需要全面考虑各种情况，分别给出提示信息。总的来说，这次的大作业开发周期不短，实现了良好的功能，同时我也学到了各方面的许多知识，对于我个人来说，是一次提升的过程。

### 管理员界面开发-袁恩晗

1. 所做工作

* 实现了管理员界面
* 实现了添加车票界面
* 实现了删除车票功能
* 实现了修改车票界面
* 实现了添加用户界面
* 实现了车票数据文件和用户数据文件的读取、修改和存入

1. 开发中遇到的问题

* 已解决的问题
  + QTableView排序功能
  + 用户、车票信息的及时更新功能实现
  + 不同窗口间的值传递问题、以及不同窗口间的调用问题
  + 对于用户不同行为的判断问题
  + 退出时保存所有修改
* 暂未解决的问题
  + 在QTableView中操作数据问题
  + 更为完善的车票信息冲突判断
  + ui界面控件的简化

1. 收获与想法

收获：在完成本次大作业的过程中，我熟悉了Qt的基本框架和模块,进一步掌握了C++编程技巧，学会了如何使用Qt Creator进行开发，特别是学会了ui界面的搭建，提高了自己解决实际问题能力。  
在撰写实验报告和README文档的过程中，我第一次接触了md文档的书写方法，更加具体的了解了一个较为正规的程序所应该具有的各项细节。学习了流程图，层次图，数据流图的绘制流程，帮助我更好地理解软件层次，从而更好地设计和实现软件。

想法：完成大作业的过程对于我来说应该是一个宝贵的财富。事实上完成大作业花费了我大量的时间和精力，同时也要克服许多困难和挑战。可以说经过了c语言和c++语言的学习，完成想要的功能并不困难。但是对于刚刚接触Qt的我来说，有太多的新知识需要去学习，去适应，去应用，经过了很长的磨合期，我才堪堪掌握Qt中各种功能的使用。但这不正是学习一项新知识该有的过程，我想这也是以后学习，工作时时常会遇到的情况，这给我打下了良好的思想基础。同时，在完成这次大作业的过程中，我也逐步的适应了团队开发的模式，学会在团队中交流，发表意见，提供想法，和队员协调各项事宜，共同完成程序的开发，测试。并且，我也学会了从不同角度审视一个软件。要考虑软件的美观性，实用性，尽可能的让软件适合更多的人使用，这给了我在程序设计方面很好的启发。总之，我十分感恩这次c语言程序设计大作业。