**面向对象程序课程设计**

**（2023/2024学年第二学期）**

**指导教师：陈燕兵**

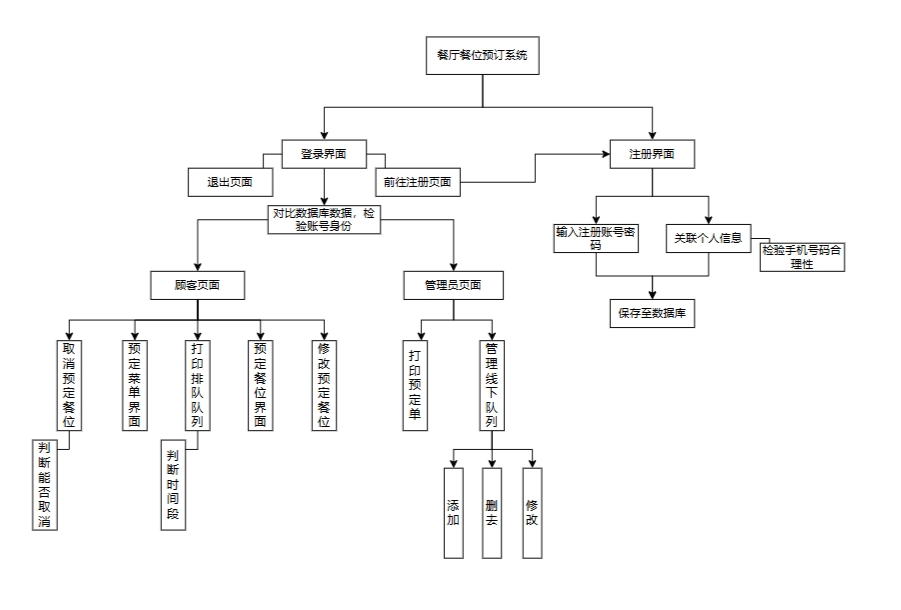
**班级：全英班**

**学号：2023337621159**

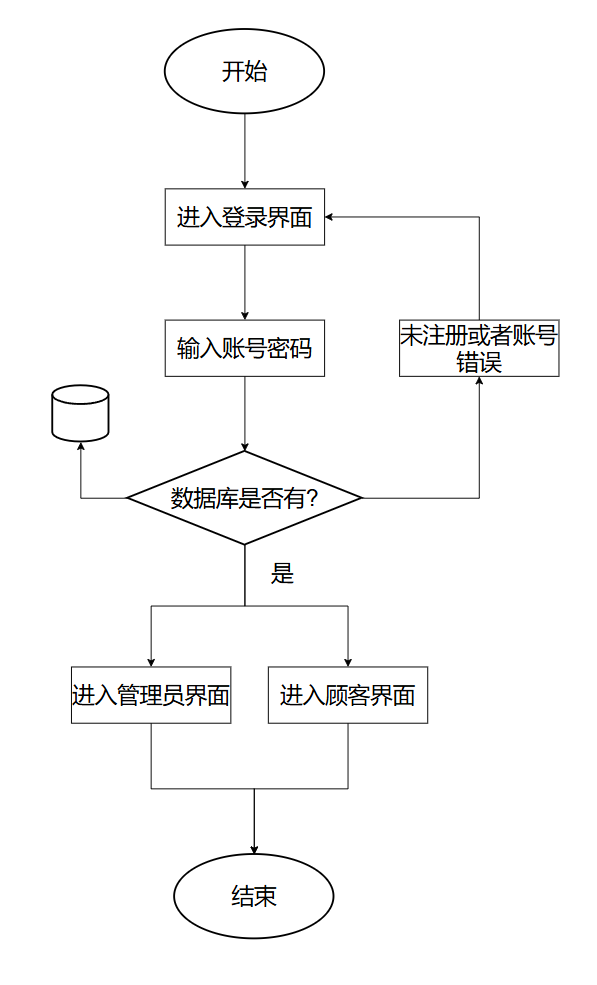
**姓名：代翔**

任务1 餐厅餐位预约系统

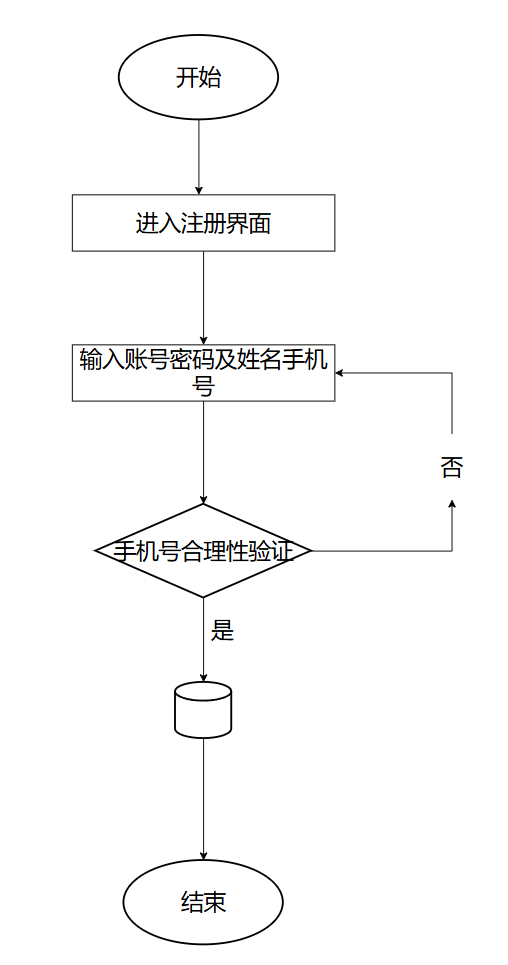
1. 系统功能结构图



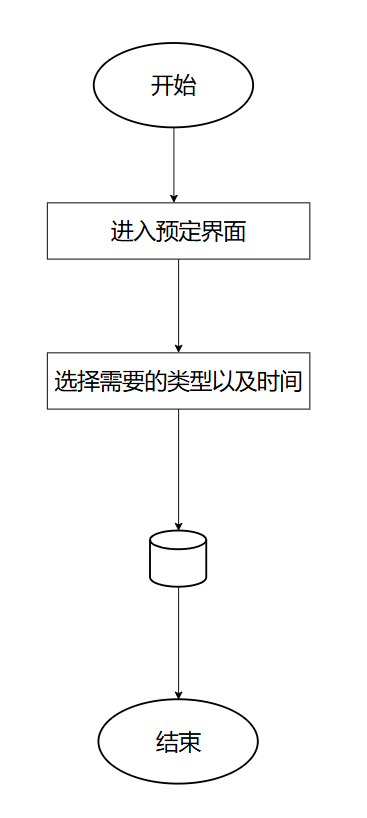
1. 业务流程图
2. 登录流程



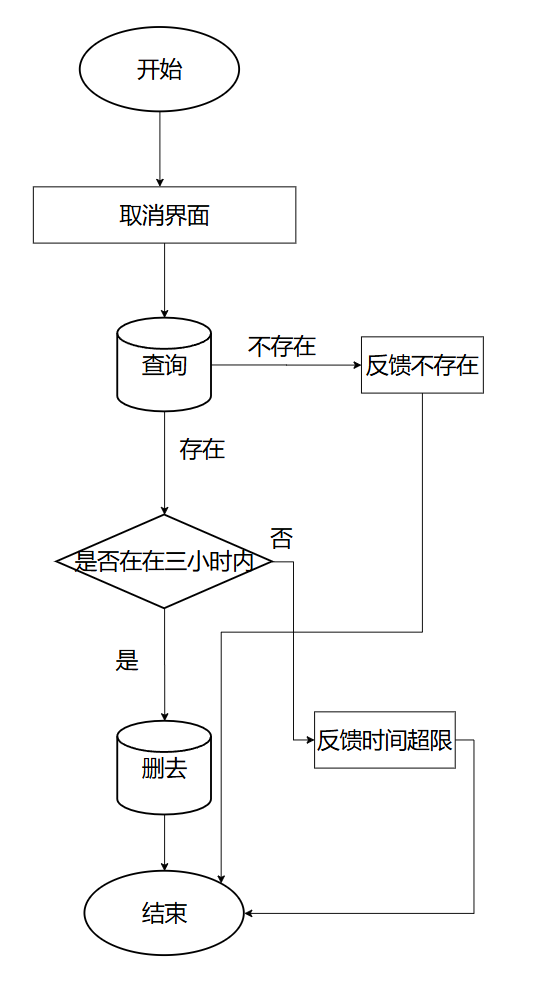
1. 注册流程



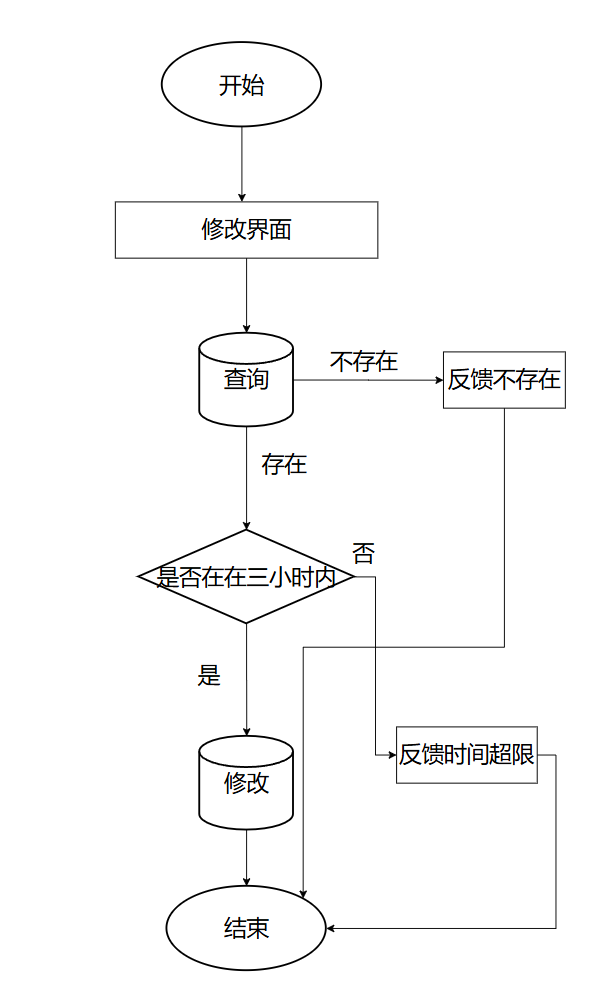
1. 预定餐位流程



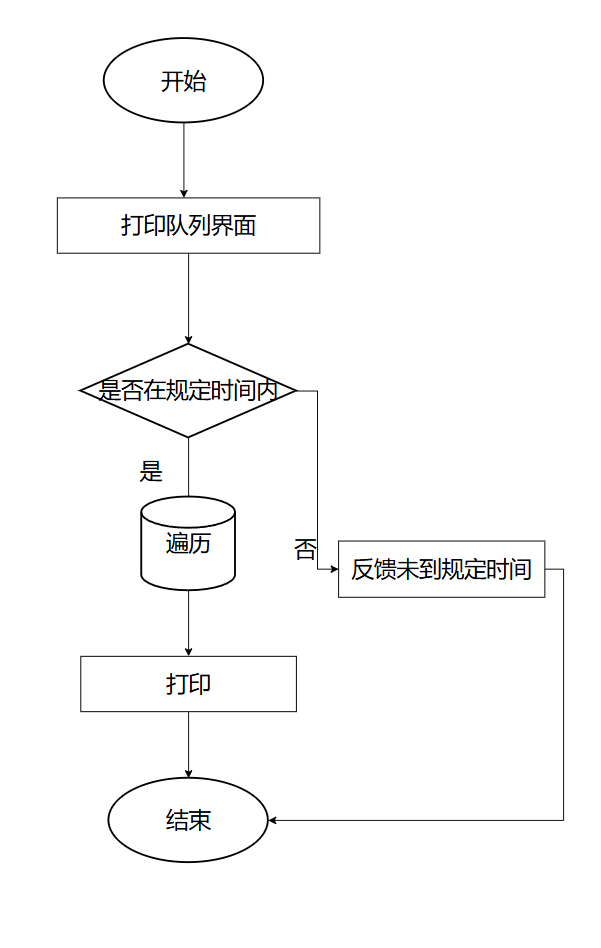
1. 取消餐位流程



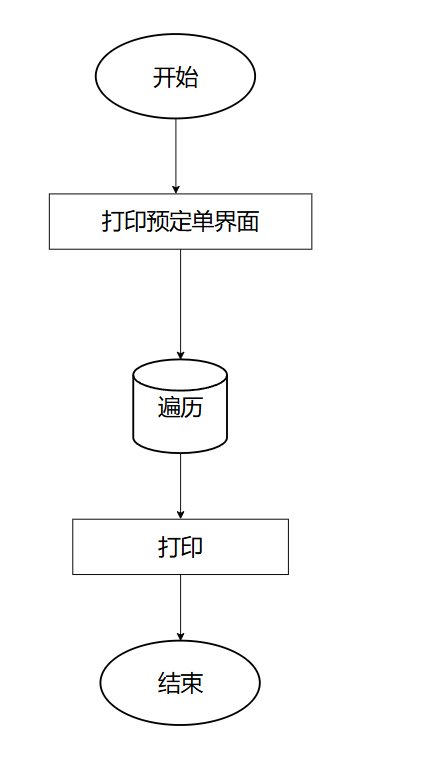
1. 修改餐位流程



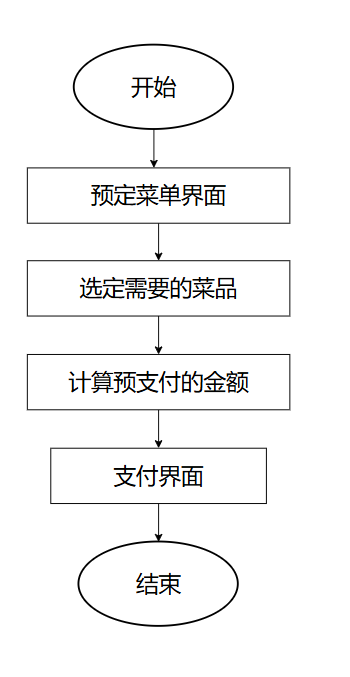
1. 打印排队队列流程



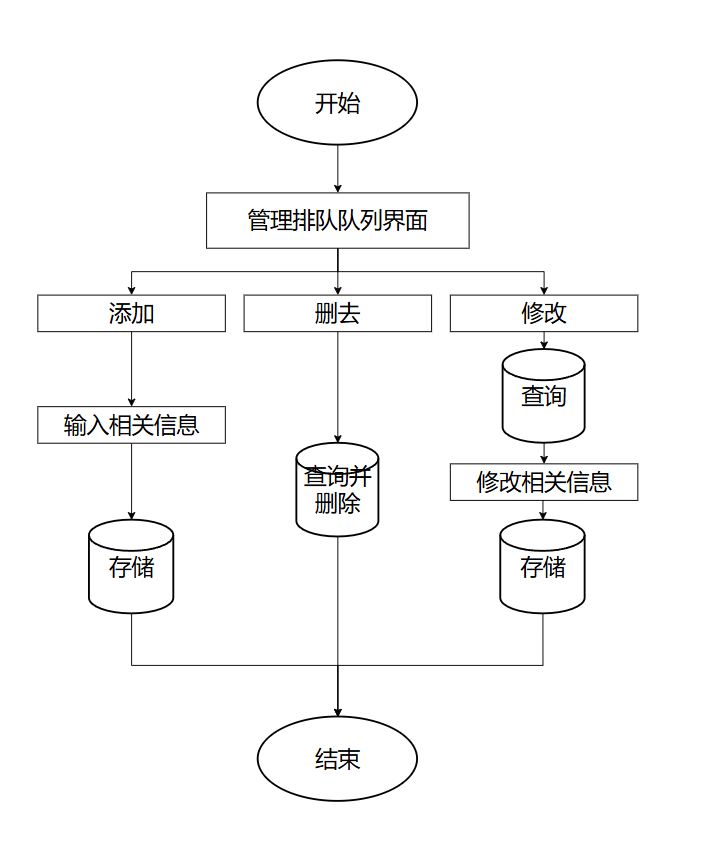
1. 打印预定菜单流程



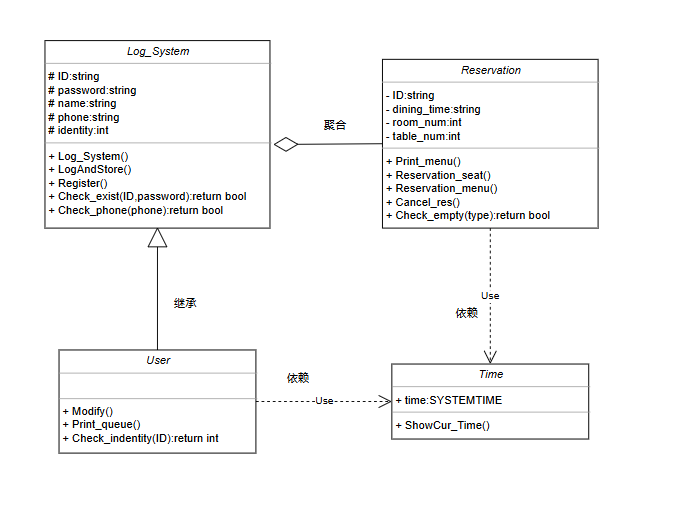
1. 预定菜单流程



1. 管理排队队列流程



1. 类的设计



1. 程序代码与说明

本系统源码共由10个源文件administor.cpp、book\_menu.cpp、form.cpp、login.cpp、queue\_manage.cpp、queue\_print.cpp、reservation\_print.cpp、reservation.cpp、widget.cpp、main.cpp和9个头文件administor.h、book\_menu.h、form.h、login.h、queue\_manage.h、queue\_print.h、reservation\_print.h、reservation.h、widget.h以及资源文件pic.qrc组成。

1. administor.h

|  |
| --- |
| #ifndef ADMINISTOR\_H  #define ADMINISTOR\_H  #include <QWidget>  namespace Ui {  class Administor;  }  class Administor : public QWidget  {  Q\_OBJECT  public:  explicit Administor(QWidget \*parent = nullptr);  ~Administor();  private slots:  void on\_pushButton\_4\_clicked();  void Current\_diasplay();  void on\_pushButton\_5\_clicked();  void on\_pushButton\_6\_clicked();  private:  Ui::Administor \*ui;  QTimer \*timer;  };  #endif // ADMINISTOR\_H |

1. book\_menu.h

|  |
| --- |
| #ifndef BOOK\_MENU\_H  #define BOOK\_MENU\_H  #include <QWidget>  namespace Ui {  class book\_menu;  }  class book\_menu : public QWidget  {  Q\_OBJECT  public:  explicit book\_menu(QWidget \*parent = nullptr);  ~book\_menu();  private slots:  void on\_pushButton\_clicked();  void on\_pushButton\_2\_clicked();  private:  Ui::book\_menu \*ui;  };  #endif // BOOK\_MENU\_H |

1. form.h

|  |
| --- |
| #ifndef FORM\_H  #define FORM\_H    #include <QWidget>  namespace Ui {  class Form;  }  class Form : public QWidget  {  Q\_OBJECT  public:  explicit Form(QWidget \*parent = nullptr);  ~Form();  private slots:  void on\_pushButton\_2\_clicked();  void on\_complete\_clicked();  private:  Ui::Form \*ui;  };  #endif // FORM\_H |

1. login.h

|  |
| --- |
| #ifndef LOGIN\_H  #define LOGIN\_H  #include <QWidget>  class QStandardItemModel;  namespace Ui {  class Login;  }  class Login : public QWidget  {  Q\_OBJECT  public:  explicit Login(QWidget \*parent = nullptr);  ~Login();  private slots:  void on\_pushButton\_5\_clicked();  void current\_display();  void on\_pushButton\_clicked();  void on\_pushButton\_3\_clicked();  void on\_pushButton\_6\_clicked();  void on\_pushButton\_2\_clicked();  void on\_pushButton\_4\_clicked();  private:  Ui::Login \*ui;  QTimer\* timer;  QStandardItemModel \*model;  };  #endif // LOGIN\_H |

1. queue\_manage.h

|  |
| --- |
| #ifndef QUEUE\_MANAGE\_H  #define QUEUE\_MANAGE\_H  #include <QWidget>  class QStandardItemModel;  namespace Ui {  class Queue\_Manage;  }  class Queue\_Manage : public QWidget  {  Q\_OBJECT  public:  explicit Queue\_Manage(QWidget \*parent = nullptr);  ~Queue\_Manage();  void read\_data();  private slots:  void on\_pushButton\_clicked();  void on\_pushButton\_4\_clicked();  void on\_pushButton\_2\_clicked();  void on\_pushButton\_5\_clicked();  private:  Ui::Queue\_Manage \*ui;  QStandardItemModel \*model;  };  #endif // QUEUE\_MANAGE\_H |

1. queue\_print.h

|  |
| --- |
| #ifndef QUEUE\_PRINT\_H  #define QUEUE\_PRINT\_H  #include <QWidget>  class QStandardItemModel;  namespace Ui {  class Queue\_print;  }  class Queue\_print : public QWidget  {  Q\_OBJECT  public:  explicit Queue\_print(QWidget \*parent = nullptr);  ~Queue\_print();  void read\_data();  private slots:  void on\_pushButton\_clicked();  private:  Ui::Queue\_print \*ui;  QStandardItemModel \*model;  };  #endif // QUEUE\_PRINT\_H |

1. reservation\_print.h

|  |
| --- |
| #ifndef RESERVATION\_PRINT\_H  #define RESERVATION\_PRINT\_H  #include <QWidget>  class QStandardItemModel;  namespace Ui {  class Reservation\_print;  }  class Reservation\_print : public QWidget  {  Q\_OBJECT  public:  explicit Reservation\_print(QWidget \*parent = nullptr);  ~Reservation\_print();  void read\_data();  private slots:  void on\_pushButton\_clicked();  private:  Ui::Reservation\_print \*ui;  QStandardItemModel \*model;  };  #endif // RESERVATION\_PRINT\_H |

1. reservationseat.h

|  |
| --- |
| #ifndef RESERVATIONSEAT\_H  #define RESERVATIONSEAT\_H  #include <QWidget>  #include <QRadioButton>  #include <QCheckBox>  #include <QGroupBox>  #include <QMessageBox>  namespace Ui {  class ReservationSeat;  }  class ReservationSeat : public QWidget  {  Q\_OBJECT  public:  explicit ReservationSeat(QWidget \*parent = nullptr);  ~ReservationSeat();  private slots:  void on\_radioButton\_2\_toggled(bool checked);  void on\_radioButton\_toggled(bool checked);  void on\_radioButton\_3\_toggled(bool checked);  void on\_pushButton\_2\_clicked();  void on\_pushButton\_clicked();  void current\_display();  private:  Ui::ReservationSeat \*ui;  QTimer \*timer;  };  #endif // RESERVATIONSEAT\_H |

1. widget.h

|  |
| --- |
| #ifndef WIDGET\_H  #define WIDGET\_H  #include <QWidget>  QT\_BEGIN\_NAMESPACE  namespace Ui {  class log\_page;  }  QT\_END\_NAMESPACE  class Widget : public QWidget  {  Q\_OBJECT  public:  Widget(QWidget \*parent = nullptr);  ~Widget();  private slots:  void on\_pushButton\_3\_clicked();  void on\_Exit\_clicked();  void on\_log\_in\_clicked();  public slots:  QString Get\_id();  private:  Ui::log\_page \*ui;  };  #endif // WIDGET\_H |

1. administor.cpp

|  |
| --- |
| #include "administor.h"  #include "ui\_administor.h"  #include "queue\_manage.h"  #include "reservation\_print.h"  #include <QLCDNumber>  #include <QTimer>  #include <QFile>  Administor::Administor(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Administor)  {  ui->setupUi(this);  ui->lcdNumber->setDigitCount(8);  //创建定时器  timer=new QTimer(this);  connect(timer,&QTimer::timeout,this,&Administor::Current\_diasplay);  timer->start(1000);  }  void Administor::Current\_diasplay()  {  QString cur\_time=QDateTime::currentDateTime().toString("hh:mm:ss");  ui->lcdNumber->display(cur\_time);  }  Administor::~Administor()  {  delete ui;  }  void Administor::on\_pushButton\_4\_clicked()  {  this->close();  }  void Administor::on\_pushButton\_5\_clicked()  {  Reservation\_print\* c=new Reservation\_print;  c->show();  }  void Administor::on\_pushButton\_6\_clicked()  {  Queue\_Manage \*a=new Queue\_Manage;  a->show();  } |

1. book\_menu.cpp

|  |
| --- |
| #include "book\_menu.h"  #include "ui\_book\_menu.h"  #include "widget.h"  #include <QFile>  #include <QTextStream>  #include <QMessageBox>  #include <QLabel>  int food\_number[8]={0,0,0,0,0,0,0,0};  const double price[8]={9.99,19.99,17.99,18.99,16.99,15.99,20.99,10.99};  const QString name[8]={"mapo tofu","beef","set meal 1","set meal 2","set meal 3","beef noodles","set meal 4","chicken"};  book\_menu::book\_menu(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::book\_menu)  {  ui->setupUi(this);  }  book\_menu::~book\_menu()  {  delete ui;  }  void book\_menu::on\_pushButton\_clicked()  {  this->close();  }  void book\_menu::on\_pushButton\_2\_clicked()  {  QMessageBox qmb;  qmb.setText("If you book now,you will get 90% discount,but you need to pay 10% advancely.");  qmb.exec();  QFile file("book\_menu.txt");  if(!file.open(QIODevice::Text|QIODevice::Append|QIODevice::WriteOnly))  {  qDebug()<<"open failed.";  }  QTextStream out(&file);  Widget \*a=new Widget;  QString id=a->Get\_id();  food\_number[0]=ui->spinBox->value();  food\_number[1]=ui->spinBox\_2->value();  food\_number[2]=ui->spinBox\_3->value();  food\_number[3]=ui->spinBox\_4->value();  food\_number[4]=ui->spinBox\_5->value();  food\_number[5]=ui->spinBox\_6->value();  food\_number[6]=ui->spinBox\_7->value();  food\_number[7]=ui->spinBox\_8->value();  double total\_pay=0;  for(int i=0;i<8;i++)  {  total\_pay+=price[i]\*food\_number[i]\*0.9;  }  double advance\_pay=total\_pay\*0.1;  total\_pay-=advance\_pay;  out<<id<<":";  for(int i=0;i<8;i++)  {  if(food\_number[i]!=0)  {  out<<name[i]<<":"<<food\_number[i]<<"|";  }  }  out<<"Remaining to pay:"<<total\_pay<<Qt::endl;  QString ad\_pay\_s=QString::number(advance\_pay);  QString ad="you need to pay "+ad\_pay\_s+"$ advancely";  QMessageBox qeb;  qeb.setText(ad);  qeb.exec();  QLabel \*label=new QLabel();  QPixmap pixmap("C:/Users/dai/Desktop/818107712257358933f54871ad21a65.jpg");  label->setPixmap(pixmap);  label->resize(1400,800);  label->show();  } |

1. form.cpp

|  |
| --- |
| #include "form.h"  #include "ui\_form.h"  #include"widget.h"  #include <QString>  #include <QFile>  #include <QTextStream>  #include <QLineEdit>  #include <QMessageBox>  #include <QDebug>  Form::Form(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Form)  {  ui->setupUi(this);  }  bool Check\_phone(const QString &phone)  {  if (phone.length() < 11)  {  return false;  }  if (phone[0] != '1')  {  return false;  }  return true;  }  Form::~Form()  {  delete ui;  }  void Form::on\_pushButton\_2\_clicked()  {  this->hide();  Widget \*b=new Widget;  b->show();  }  void Form::on\_complete\_clicked()  {  QFile file("Information2.txt");  if (!file.open(QIODevice::Append | QIODevice::Text|QIODevice::WriteOnly)) {  qDebug() << "failed to open.";  return;  }  QTextStream out(&file);  QString id = ui->ID->toPlainText();  QString password = ui->PASSWORD->toPlainText();  QString name = ui->NAME->toPlainText();  QString phone = ui->PHONE->toPlainText();  if (!Check\_phone(phone))  {  QMessageBox::warning(nullptr, "warniing", "please input the right phone number.");  }  else  {  out << id << "|" << password << "|" << name << "|" << phone << "|" << Qt::endl;  QMessageBox qmb;  qmb.addButton("I Get it.",QMessageBox::AcceptRole);  qmb.setText("Succeed!");  qmb.exec();  }  file.close();  } |

1. login.cpp

|  |
| --- |
| #include "login.h"  #include "ui\_login.h"  #include "queue\_print.h"  #include <book\_menu.h>  #include <widget.h>  #include <reservationseat.h>  #include <QTimer>  #include <QDateTime>  #include <QTime>  #include <QDebug>  #include <QCoreApplication>  #include <QDebug>  #include <QFile>  #include <QTextStream>  #include <QString>  #include <QStringList>  #include <QPixmap>  #include <QLabel>  Login::Login(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Login)  {  ui->setupUi(this);  ui->lcdNumber->setDigitCount(8);  //创建定时器  timer=new QTimer(this);  connect(timer,&QTimer::timeout,this,&Login::current\_display);//timeout超时信号  timer->start(1000);  }  void Login::current\_display()  {  QString cur\_time=QDateTime::currentDateTime().toString("hh:mm:ss");  ui->lcdNumber->display(cur\_time);  }  Login::~Login()  {  delete ui;  }  void Login::on\_pushButton\_5\_clicked()  {  this->close();  }  void Login::on\_pushButton\_clicked()  {  this->hide();  ReservationSeat \*a=new ReservationSeat;  a->show();  }  void Login::on\_pushButton\_6\_clicked()  {  book\_menu\* a=new book\_menu;  a->show();  }  bool deleteLineWithId(QStringList& lines, const QString& id)  {  bool found = false;  for (auto it = lines.begin(); it != lines.end(); )  {  if (it->contains(id))  {  it = lines.erase(it);  found = true;  }  else  {  ++it;  }  }  return found;  }  QDateTime Qs\_to\_QDate(QString &R\_time)  {  QTime time=QTime::fromString(R\_time);  QDateTime cur=QDateTime::currentDateTime();  QDateTime l\_time=QDateTime(cur.date(),time);  return l\_time;  }  void Cancel\_res()  {  QString R\_time;  QFile file("Reservation3.txt");  if (!file.open(QIODevice::ReadOnly))  {  qDebug() << "File open failed.";  }  QTextStream in(&file);  while (!in.atEnd())  {  QString line = in.readLine();  QStringList parts = line.split("|");  R\_time=parts.last();  }  file.close();  QDateTime l\_time;  QDateTime now = QDateTime::currentDateTime();  Widget a;  QString id=a.Get\_id();  l\_time=Qs\_to\_QDate(R\_time);  if (now.addSecs(3 \* 60 \* 60) >= l\_time)  {  QString filename = "Reservation3.txt";  QString id\_to\_delete = id;  QFile inputFile(filename);  if (inputFile.open(QIODevice::ReadOnly))  {  QTextStream in(&inputFile);  QStringList lines;  while (!in.atEnd()) {  lines.append(in.readLine());  }  inputFile.close();  bool deleted = deleteLineWithId(lines, id\_to\_delete);  QFile outputFile(filename);  if (outputFile.open(QIODevice::WriteOnly | QIODevice::Text))  {  QTextStream out(&outputFile);  for (const QString& line : lines)  {  out << line << Qt::endl;  }  outputFile.close();  if (deleted)  {  QMessageBox suc;  suc.setText("Success,but you need to pay 5% of the reservation cost.");  suc.addButton("OK.",QMessageBox::AcceptRole);  suc.exec();  QLabel \*label=new QLabel();  QPixmap pixmap("C:/Users/dai/Desktop/818107712257358933f54871ad21a65.jpg");  label->setPixmap(pixmap);  label->resize(1400,800);  label->show();  }  else  {  qDebug() << "Fail to find";  }  }  else  {  QMessageBox fal;  fal.setText("You haven't book a seat yet.");  fal.exec();  }  }  else  {  qDebug() << "Open failed";  }  }  else  {  QMessageBox win;  win.setText("Time out,you can't cancel it.");  win.exec();  }  }  void Login::on\_pushButton\_3\_clicked()  {  Cancel\_res();  }  void Login::on\_pushButton\_2\_clicked()  {  Cancel\_res();  this->hide();  ReservationSeat \*a=new ReservationSeat;  a->show();  }  bool Is\_Time\_Fair()  {  QTime now=QTime::currentTime();  int cur\_hour = now.hour();  int cur\_minute = now.minute();  int morning\_start\_hour = 10;  int morning\_start\_minute = 0;  int morning\_end\_hour = 12;  int morning\_end\_minute = 30;  int evening\_start\_hour = 17;  int evening\_start\_minute = 0;  int evening\_end\_hour = 19;  int evening\_end\_minute = 30;  if ((cur\_hour == morning\_start\_hour && cur\_minute > morning\_start\_minute) || (cur\_hour > morning\_start\_hour && cur\_hour < morning\_end\_hour) || (cur\_hour == morning\_end\_hour && cur\_minute < morning\_end\_minute))  {  return true;  }  if ((cur\_hour == evening\_start\_hour && cur\_minute > evening\_start\_minute) || (cur\_hour > evening\_start\_hour && cur\_hour < evening\_end\_hour) || (cur\_hour == evening\_end\_hour && cur\_minute < evening\_end\_minute))  {  return true;  }  return false;  }  void Login::on\_pushButton\_4\_clicked()  {  if(Is\_Time\_Fair())  {  Queue\_print\* c=new Queue\_print;  c->show();  }  else  {  QMessageBox \*qb=new QMessageBox;  qb->setText("Time unfair,please wait.");  qb->exec();  }  } |

1. main.cpp

|  |
| --- |
| #include "widget.h"  #include <QApplication>  int main(int argc, char \*argv[])  {  QApplication a(argc, argv);  Widget w;  w.show();  return a.exec();  } |

1. queue\_manage.cpp

|  |
| --- |
| #include "queue\_manage.h"  #include "ui\_queue\_manage.h"  #include <QFile>  #include <QDebug>  #include <QTextStream>  #include <QStandardItemModel>  Queue\_Manage::Queue\_Manage(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Queue\_Manage)  {  ui->setupUi(this);  }  Queue\_Manage::~Queue\_Manage()  {  delete ui;  delete model;  }  void Queue\_Manage::read\_data()  {  QFile file("Queue.txt");  if(!file.open(QIODevice::ReadOnly))  {  qDebug()<<"open failed.";  return;  }  //创立模型  model=new QStandardItemModel(this);  //把模型交给视图  ui->tableView->setModel(model);  //设立表头  QStringList header={"NAME","TIME","TABLE NUMBER"};  model->setHorizontalHeaderLabels(header);  //行存储  QTextStream in(&file);  QList<QStandardItem\*> items;  while(!in.atEnd())  {  items.clear();  QStringList queue =in.readLine().split('|');  for(auto data:queue)  {  items.push\_back(new QStandardItem(data));  }  model->appendRow(items);  }  file.close();  }  void Queue\_Manage::on\_pushButton\_clicked()  {  model->setRowCount(model->rowCount()+1);  }  void Queue\_Manage::on\_pushButton\_4\_clicked()  {  QFile file("Queue.txt");  if(!file.open(QIODevice::WriteOnly))  {  qDebug()<<"open failed.";  }  QTextStream in(&file);  QStringList line\_data;  for(int i=0;i<model->rowCount();i++)  {  for(int j=0;j<model->columnCount();j++)  {  if(!model->item(i,j))  {  goto end;  }  line\_data.append(model->item(i,j)->text());  }  QString line=line\_data.join("|")+"\n";  in<<line;  }  end:;  }  void Queue\_Manage::on\_pushButton\_2\_clicked()  {  QModelIndex index=ui->tableView->currentIndex();  model->removeRow(index.row());  }  void Queue\_Manage::on\_pushButton\_5\_clicked()  {  read\_data();  } |

1. queue\_print.cpp

|  |
| --- |
| #include "queue\_print.h"  #include "ui\_queue\_print.h"  #include <QFile>  #include <QStandardItemModel>  Queue\_print::Queue\_print(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Queue\_print)  {  ui->setupUi(this);  }  void Queue\_print::read\_data()  {  QFile file("Queue.txt");  if(file.open(QIODevice::ReadOnly))  {  qDebug()<<"open failed.";  }  //创立模型  model=new QStandardItemModel(this);  //把模型交给视图  ui->tableView->setModel(model);  //设立表头  QStringList header={"NAME","TIME","TABLE NUMBER"};  model->setHorizontalHeaderLabels(header);  //行存储  QTextStream in(&file);  QList<QStandardItem\*> items;  while(!in.atEnd())  {  items.clear();  QStringList queue =in.readLine().split('|');  for(auto data:queue)  {  items.push\_back(new QStandardItem(data));  }  model->appendRow(items);  }  file.close();  }  Queue\_print::~Queue\_print()  {  delete ui;  delete model;  }  void Queue\_print::on\_pushButton\_clicked()  {  QFile file("Queue.txt");  if(!file.open(QIODevice::ReadOnly))  {  qDebug()<<"open failed.";  return;  }  //创立模型  model=new QStandardItemModel(this);  //把模型交给视图  ui->tableView->setModel(model);  //设立表头  QStringList header={"NAME","TIME","TABLE NUMBER"};  model->setHorizontalHeaderLabels(header);  //行存储  QTextStream in(&file);  QList<QStandardItem\*> items;  while(!in.atEnd())  {  items.clear();  QStringList queue =in.readLine().split('|');  for(auto data:queue)  {  items.push\_back(new QStandardItem(data));  }  model->appendRow(items);  }  file.close();  } |

1. reservation\_print.cpp

|  |
| --- |
| #include "reservation\_print.h"  #include "ui\_reservation\_print.h"  #include <QFile>  #include <qstandarditemmodel.h>  class QStandardItemModel;  Reservation\_print::Reservation\_print(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::Reservation\_print)  {  ui->setupUi(this);  }  void Reservation\_print::read\_data()  {  QFile file("Reservation3.txt");  if(!file.open(QIODevice::ReadOnly))  {  qDebug()<<"open failed.";  }  model=new QStandardItemModel(this);  ui->tableView->setModel(model);  QStringList header={"NAME","TYPE","TABLE NUMBER","TIME"};  model->setHorizontalHeaderLabels(header);  QTextStream in(&file);  QList<QStandardItem\*> items;  while(!in.atEnd())  {  items.clear();  QStringList res =in.readLine().split('|');  for(auto data:res)  {  items.push\_back(new QStandardItem(data));  }  model->appendRow(items);  }  file.close();  }  Reservation\_print::~Reservation\_print()  {  delete ui;  delete model;  }  void Reservation\_print::on\_pushButton\_clicked()  {  read\_data();  } |

1. reservationseat.cpp

|  |
| --- |
| #include "reservationseat.h"  #include "ui\_reservationseat.h"  #include <widget.h>  #include <login.h>  #include <book\_menu.h>  #include <QList>  #include <QDebug>  #include <QFile>  #include <QTextStream>  #include <QDateTime>  #include <QTimer>  #include <QDateTimeEdit>  #include <QMessageBox>  ReservationSeat::ReservationSeat(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::ReservationSeat)  {  ui->setupUi(this);  ui->checkBox->setEnabled(false);  ui->checkBox\_2->setEnabled(false);  ui->checkBox\_3->setEnabled(false);  ui->checkBox\_4->setEnabled(false);  ui->checkBox\_5->setEnabled(false);  ui->checkBox\_6->setEnabled(false);  ui->checkBox\_7->setEnabled(false);  ui->checkBox\_8->setEnabled(false);  ui->checkBox\_9->setEnabled(false);  ui->checkBox\_10->setEnabled(false);  ui->checkBox\_11->setEnabled(false);  ui->checkBox\_12->setEnabled(false);  connect(ui->radioButton,&QRadioButton::toggled,this,&ReservationSeat::on\_radioButton\_toggled);  connect(ui->radioButton\_2,&QRadioButton::toggled,this,&ReservationSeat::on\_radioButton\_2\_toggled);  connect(ui->radioButton\_3,&QRadioButton::toggled,this,&ReservationSeat::on\_radioButton\_3\_toggled);  ui->lcdNumber->setDigitCount(8);  timer=new QTimer(this);  connect(timer,&QTimer::timeout,this,&ReservationSeat::current\_display);  timer->start(1000);  ui->dateTimeEdit->setDateTime(QDateTime::currentDateTime());  ui->dateTimeEdit->setMinimumDate(QDate(2024,5,28));  ui->dateTimeEdit->setDisplayFormat("yyyy-MM-dd-hh:mm");  ui->dateTimeEdit->setCalendarPopup(true);  }  void ReservationSeat::current\_display()  {  QString cur\_time=QDateTime::currentDateTime().toString("hh:mm:ss");  ui->lcdNumber->display(cur\_time);  }  ReservationSeat::~ReservationSeat()  {  delete ui;  }  void ReservationSeat::on\_radioButton\_2\_toggled(bool checked)  {  ui->checkBox->setEnabled(checked);  ui->checkBox\_2->setEnabled(checked);  ui->checkBox\_3->setEnabled(checked);  ui->checkBox\_4->setEnabled(checked);  }  void ReservationSeat::on\_radioButton\_toggled(bool checked)  {  ui->checkBox\_5->setEnabled(checked);  ui->checkBox\_6->setEnabled(checked);  ui->checkBox\_7->setEnabled(checked);  ui->checkBox\_8->setEnabled(checked);  }  void ReservationSeat::on\_radioButton\_3\_toggled(bool checked)  {  ui->checkBox\_9->setEnabled(checked);  ui->checkBox\_10->setEnabled(checked);  ui->checkBox\_11->setEnabled(checked);  ui->checkBox\_12->setEnabled(checked);  }  void ReservationSeat::on\_pushButton\_2\_clicked()  {  this->hide();  Login \*a=new Login;  a->show();  }  void ReservationSeat::on\_pushButton\_clicked()  {  QList<QCheckBox\*> \*CheckBoxList1=new QList<QCheckBox\*>;  CheckBoxList1->append(ui->checkBox);  CheckBoxList1->append(ui->checkBox\_2);  CheckBoxList1->append(ui->checkBox\_3);  CheckBoxList1->append(ui->checkBox\_4);  QList<QCheckBox\*> \*CheckBoxList2=new QList<QCheckBox\*>;  CheckBoxList2->append(ui->checkBox\_5);  CheckBoxList2->append(ui->checkBox\_6);  CheckBoxList2->append(ui->checkBox\_7);  CheckBoxList2->append(ui->checkBox\_8);  QList<QCheckBox\*> \*CheckBoxList3=new QList<QCheckBox\*>;  CheckBoxList3->append(ui->checkBox\_9);  CheckBoxList3->append(ui->checkBox\_10);  CheckBoxList3->append(ui->checkBox\_11);  CheckBoxList3->append(ui->checkBox\_12);  QFile file("Reservation3.txt");  if(!file.open(QIODevice::Append | QIODevice::Text|QIODevice::WriteOnly))  {  qDebug()<<"Fail to open";  return;  }  QString type="";  QString number="";  QTextStream out(&file);  if(ui->radioButton\_2->isChecked())  {  type=ui->radioButton\_2->text()+'|';  for(auto it:\*CheckBoxList1)  {  if(it->isChecked())  {  number=number+it->text()+'|';  }  }  }  if(ui->radioButton->isChecked())  {  type=ui->radioButton->text()+'|';  for(auto it:\*CheckBoxList2)  {  if(it->isChecked())  {  number=number+it->text()+'|';  }  }  }  if(ui->radioButton\_3->isChecked())  {  type=ui->radioButton\_3->text()+'|';  for(auto it:\*CheckBoxList3)  {  if(it->isChecked())  {  number=number+it->text()+'|';  }  }  }  Widget a;  QString id=a.Get\_id();  QString time=ui->dateTimeEdit->text();  out<<id<<'|'<<type<<number<<time<<Qt::endl;  file.close();  QMessageBox qmb;  qmb.setText("Success.");  qmb.setStandardButtons(QMessageBox::Ok);  qmb.exec();  } |

1. widget.cpp

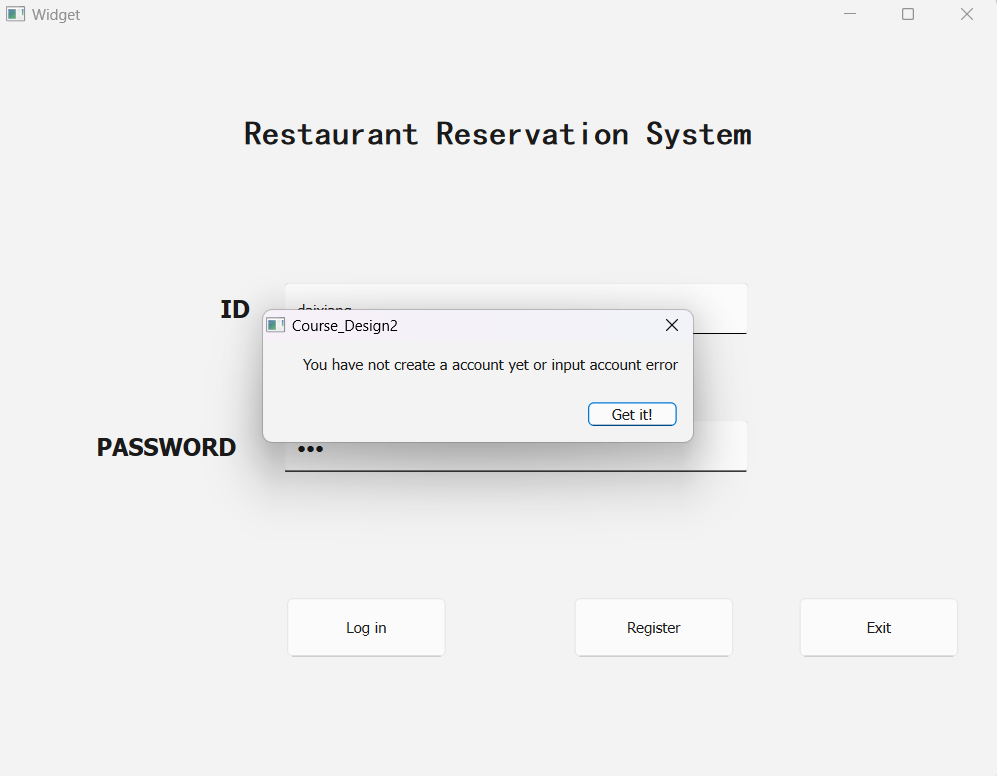
|  |
| --- |
| #include "widget.h"  #include "ui\_widget.h"  #include <QPushButton>  #include <form.h>  #include <login.h>  #include <administor.h>  #include <QFile>  #include <QTextStream>  #include <QString>  #include <QMessageBox>  QString id;  bool Check\_exist(const QString &ID, const QString &password)  {  QFile file("Information2.txt");  if (!file.open(QIODevice::ReadOnly | QIODevice::Text))  {  qDebug() << "File open failed.";  return false;  }  QTextStream in(&file);  while (!in.atEnd())  {  QString line = in.readLine();  QStringList parts = line.split("|");  if (parts.size() >= 2)  {  if (parts[0] == ID && parts[1] == password)  {  file.close();  return true;  }  }  }  file.close();  return false;  }  Widget::Widget(QWidget \*parent)  : QWidget(parent)  , ui(new Ui::log\_page)  {  ui->setupUi(this);  }  Widget::~Widget()  {  delete ui;  }  void Widget::on\_pushButton\_3\_clicked()  {  this->hide();  Form \*a=new Form;  a->show();  }  void Widget::on\_Exit\_clicked()  {  this->close();  }  void Widget::on\_log\_in\_clicked()  {  id=ui->lineEdit\_2->text();  QString password=ui->lineEdit->text();  if(id=="Administrator"&&password=="050728")  {  QMessageBox meg;  QString wel="welcome "+id;  meg.setText(wel);  meg.addButton("Get it!",QMessageBox::AcceptRole);  meg.exec();  this->hide();  Administor \*c=new Administor;  c->show();  }  else  {  if(Check\_exist(id,password))  {  QMessageBox megbox;  QString wel="welcome "+id;  megbox.setText(wel);  megbox.addButton("Get it!",QMessageBox::AcceptRole);  megbox.exec();  this->hide();  Login \*b=new Login;  b->show();  }  else  {  QMessageBox megbox;  megbox.setText("You have not create a account yet or input account error");  megbox.addButton("Get it!",QMessageBox::AcceptRole);  megbox.exec();  }  }  }  QString Widget::Get\_id()  {  return id;  } |

1. pic.qrc

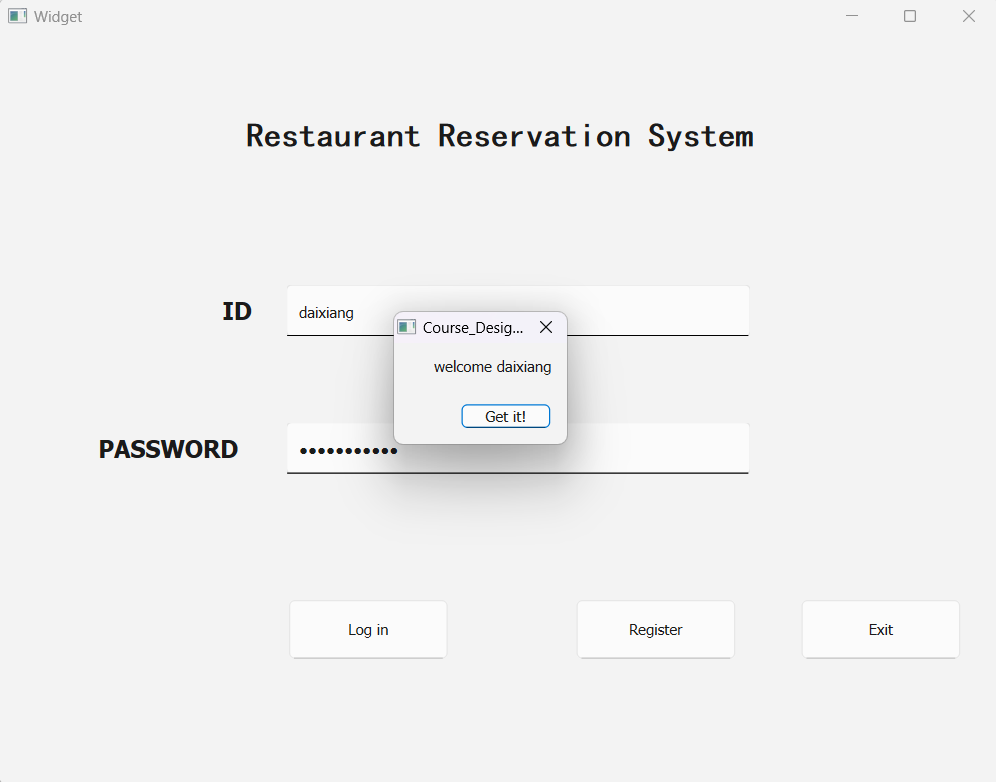
|  |
| --- |
| 图片 |

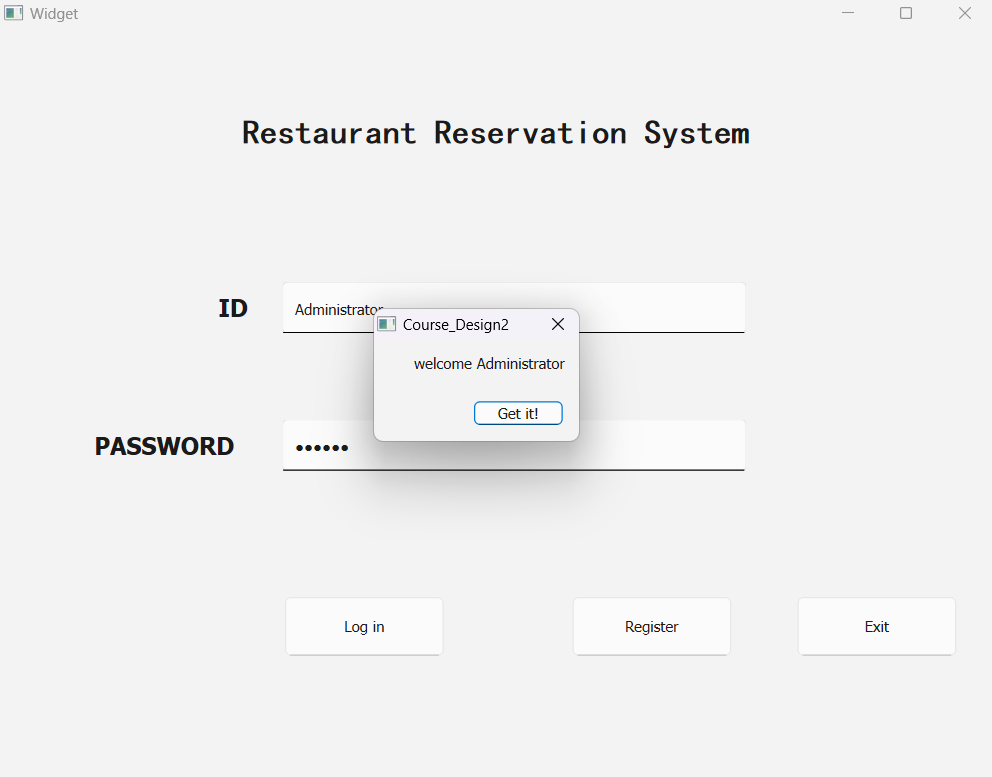
1. 运行结果与分析
2. 用户登录：

（一）如果还没注册或者密码错误会跳出提示窗口

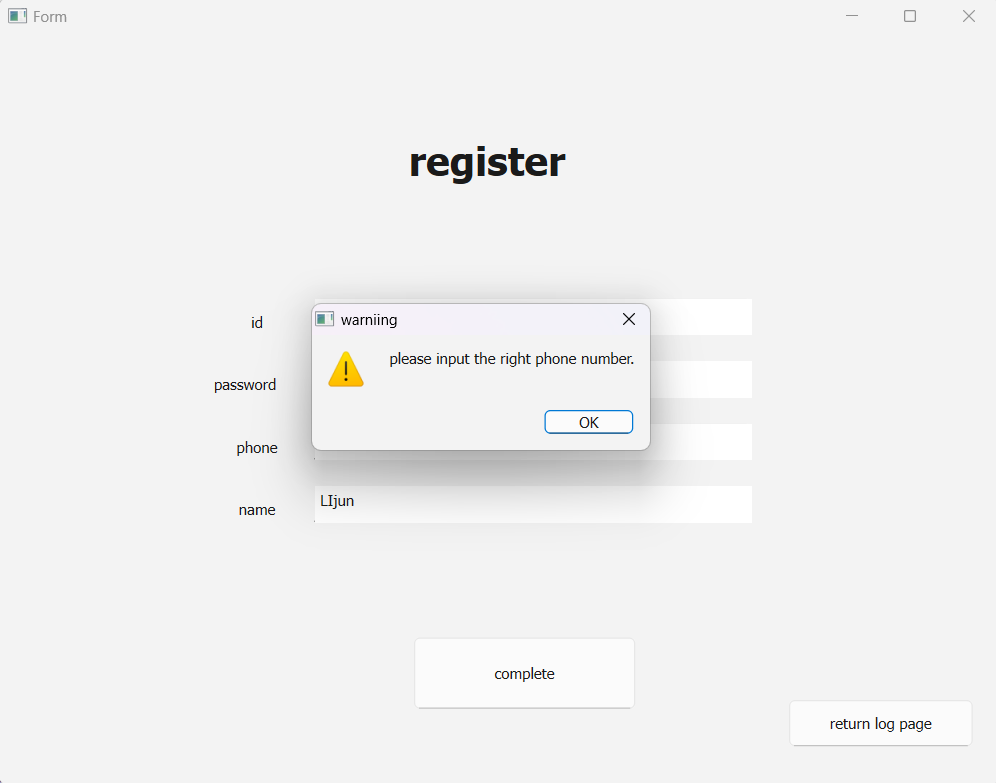


1. 如果登录成功会显示欢迎窗口：同时会根据账号性质分成管理员和用户

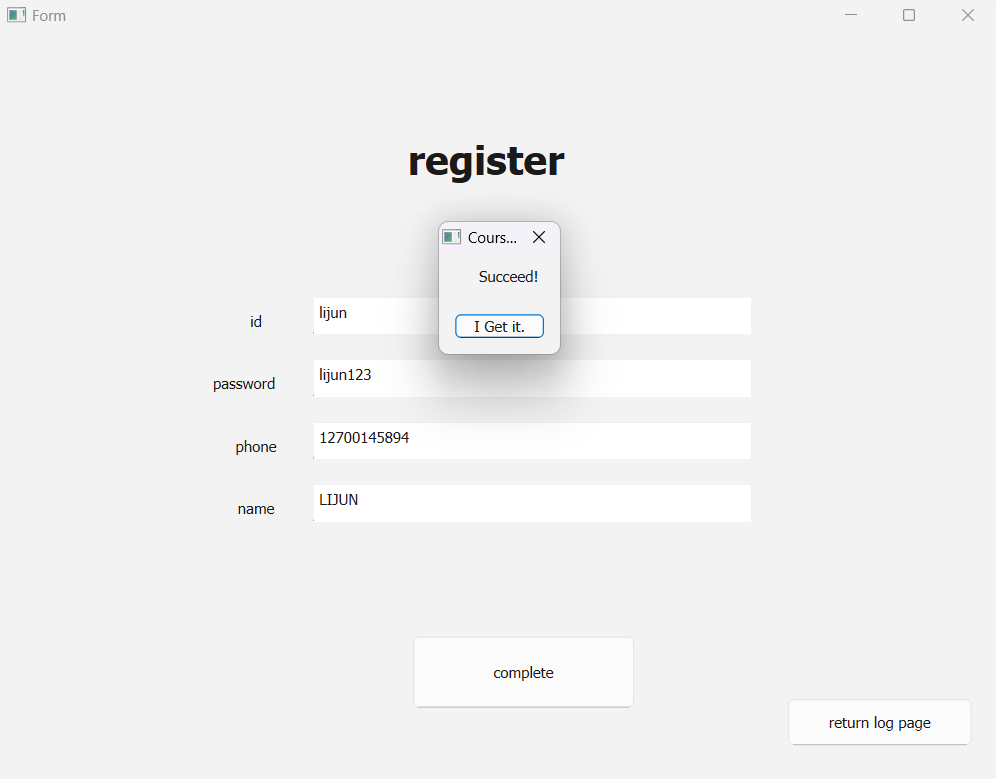


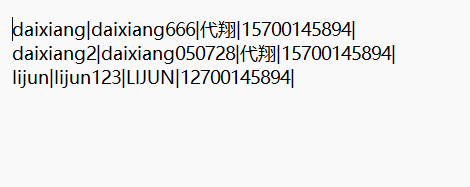


1. 游客注册：
2. 如果手机号码不符合规则会跳出提示窗口

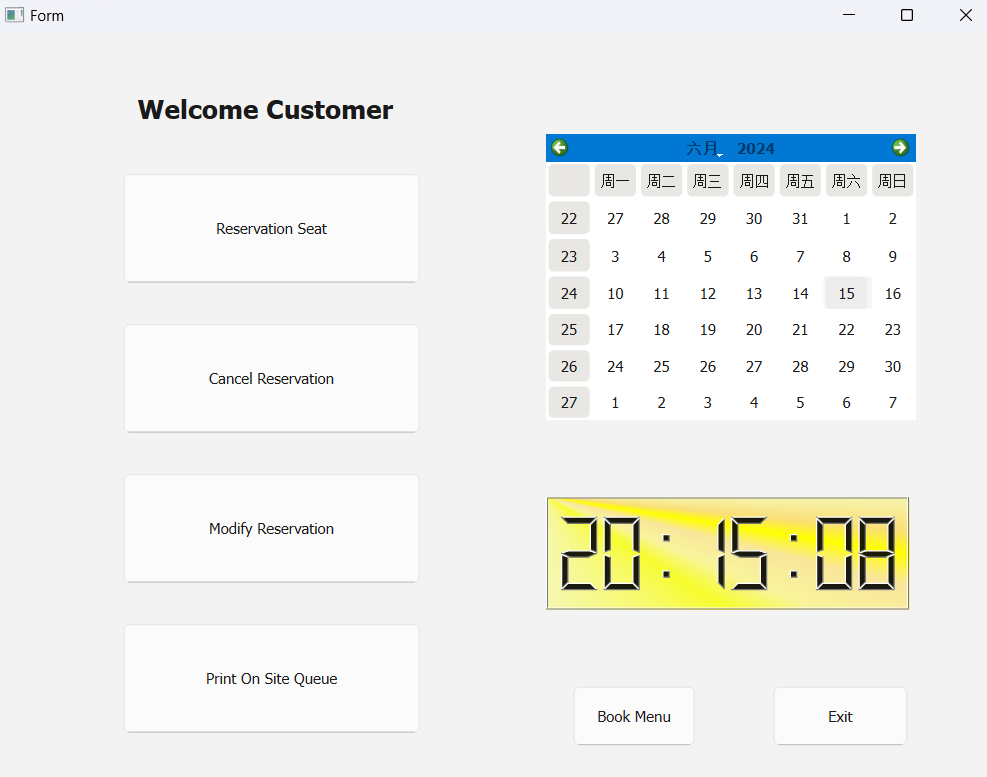


1. 注册成功会存入文本文档information并提示

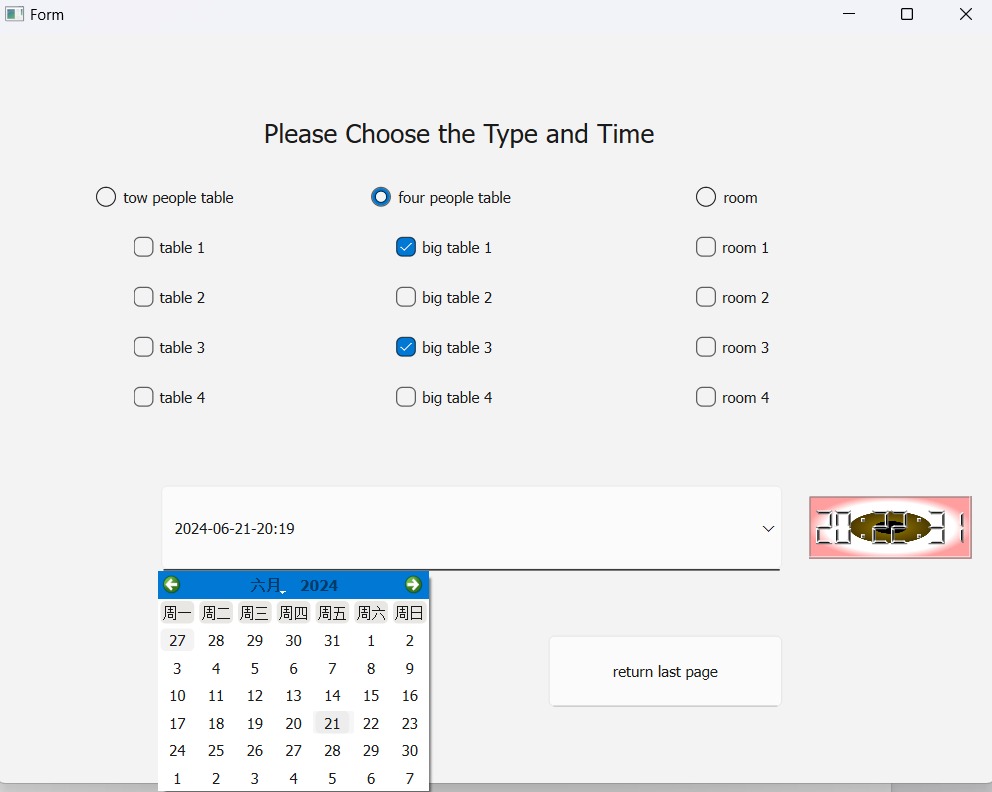




1. 顾客界面（主界面设计了一个日历和时间LED方便用户确定时间）

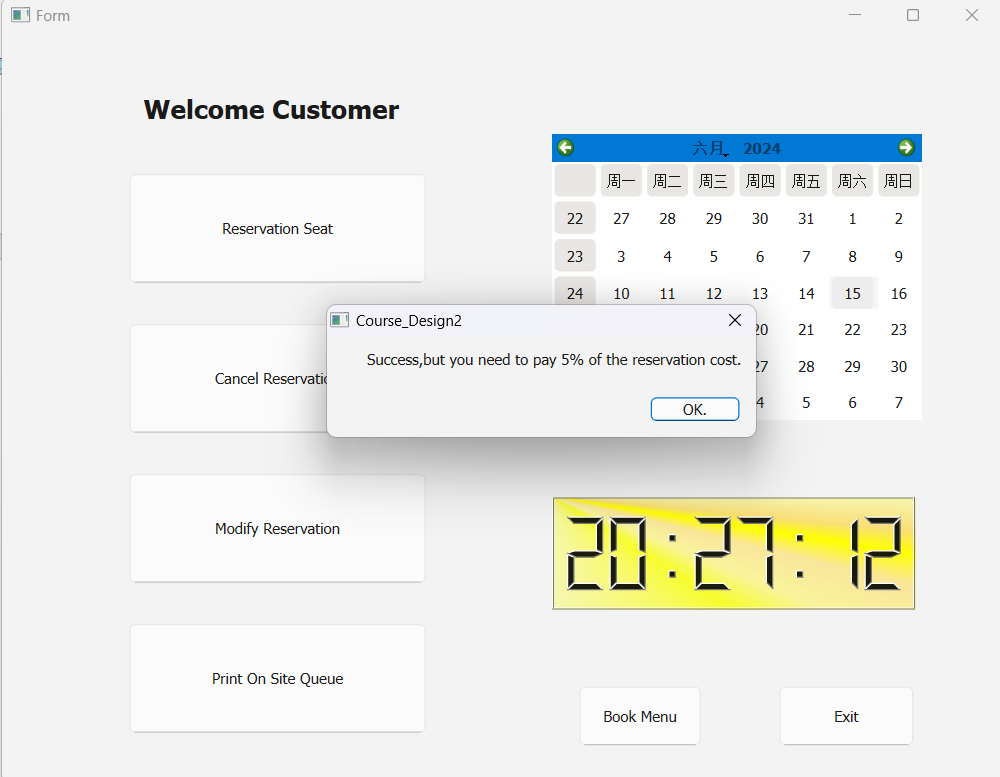


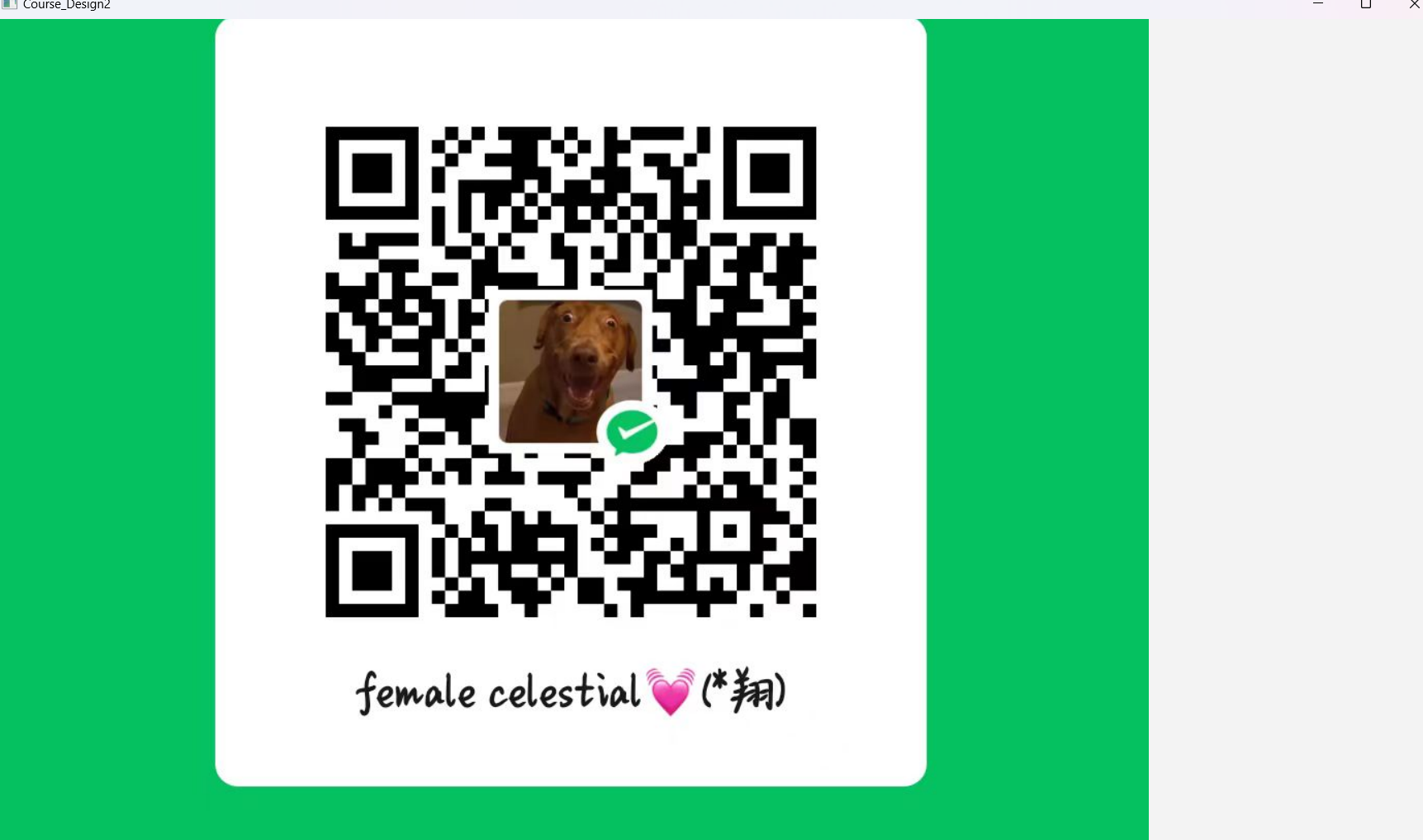
1. 预定餐桌和时间：预定无误后会将订单存入reservation文本文档，并且在设定下只有选择了是包间或者桌子类型才能解锁下面的桌号选择，以防出现错误；并且在选择时间时候可以打开日历进行选择。





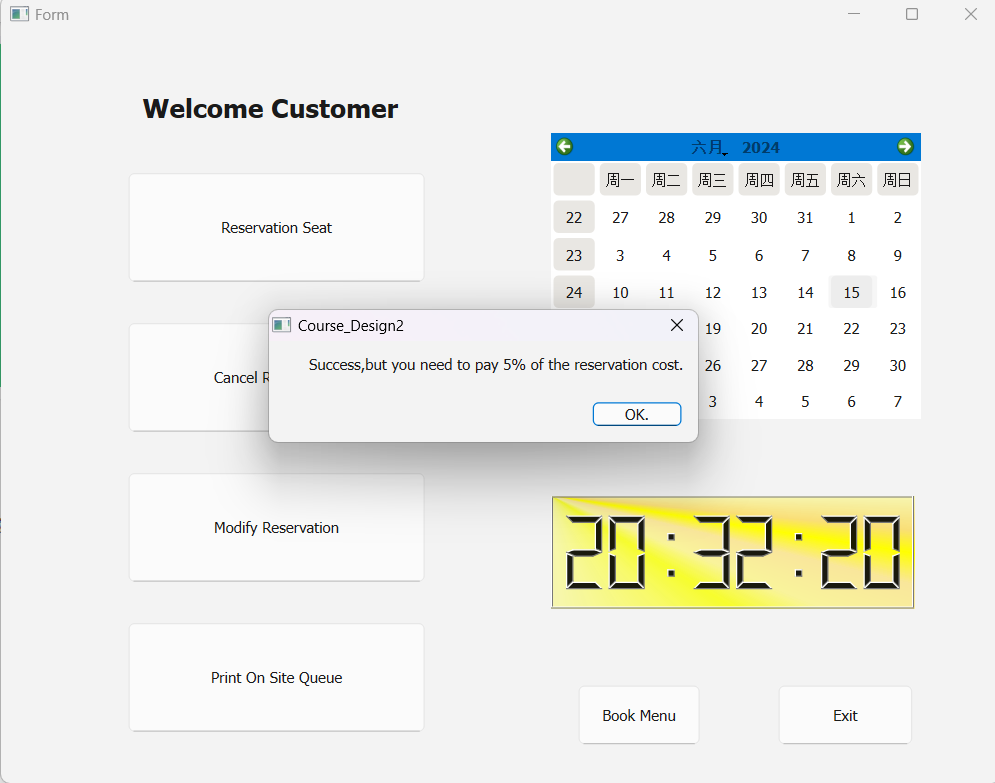
1. 取消订单：取消会有弹窗提醒成功，并修改文本文档reservation，同时需要支付一定的违约金，会跳出支付二维码。

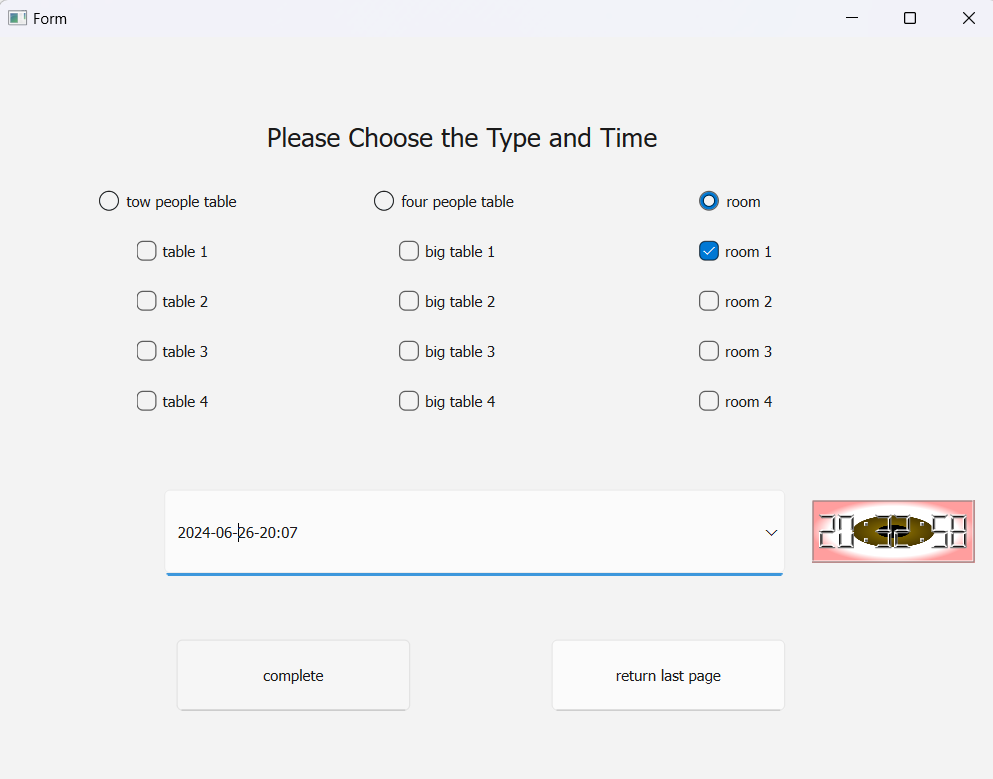






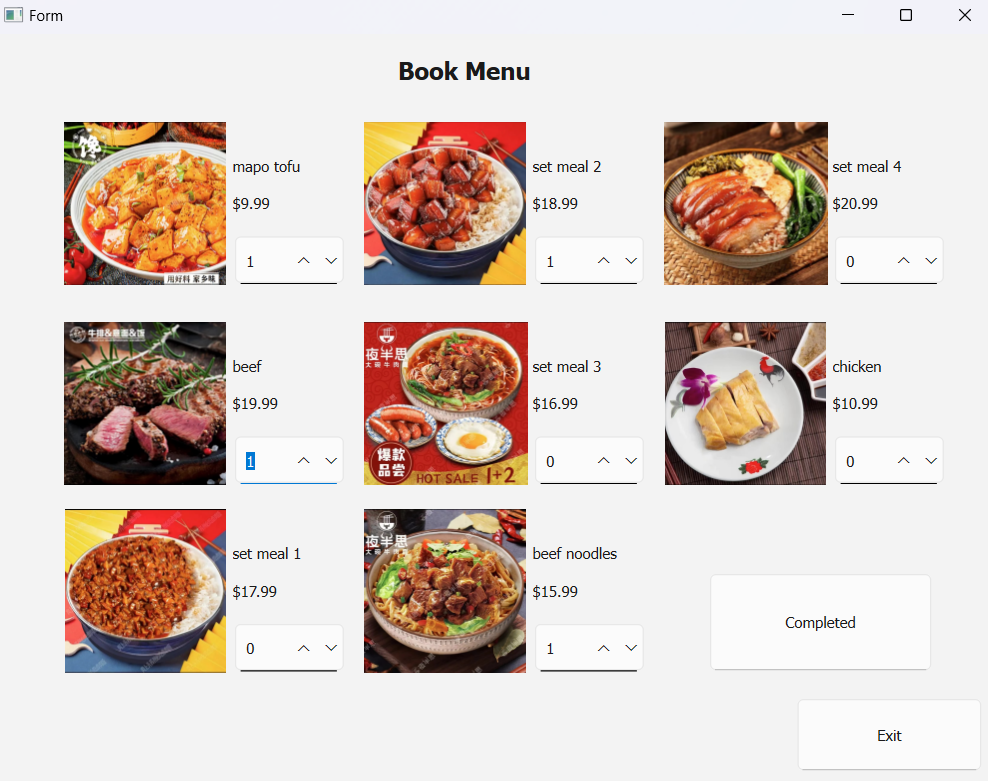
1. 修改订单：即预定单和取消订单的结合。

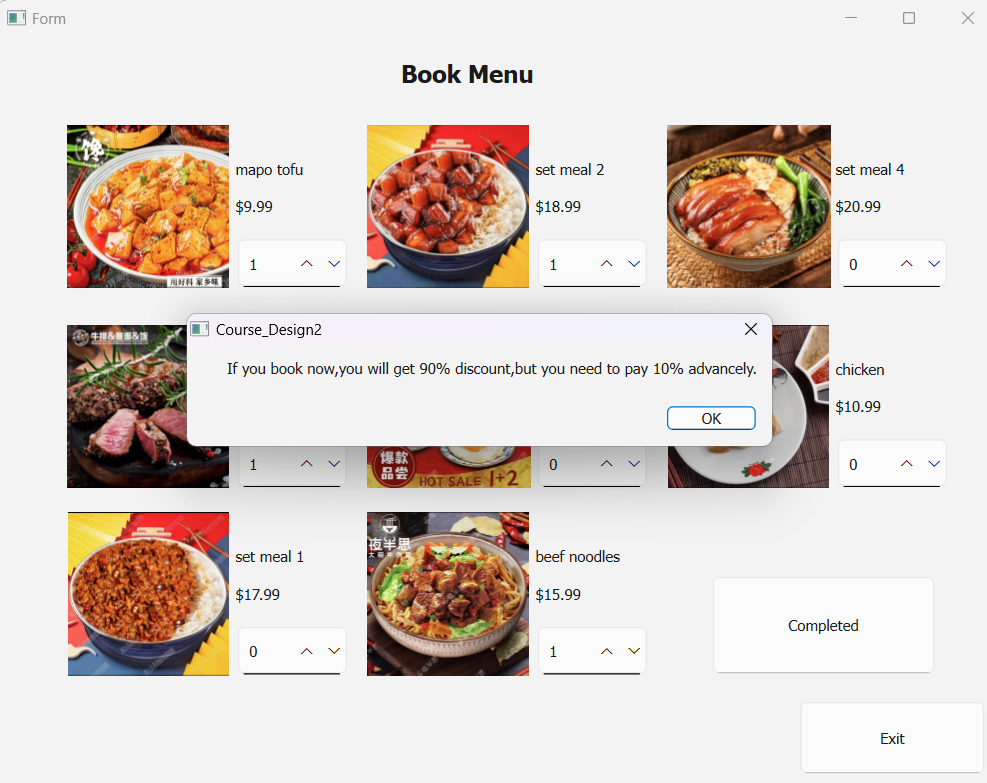


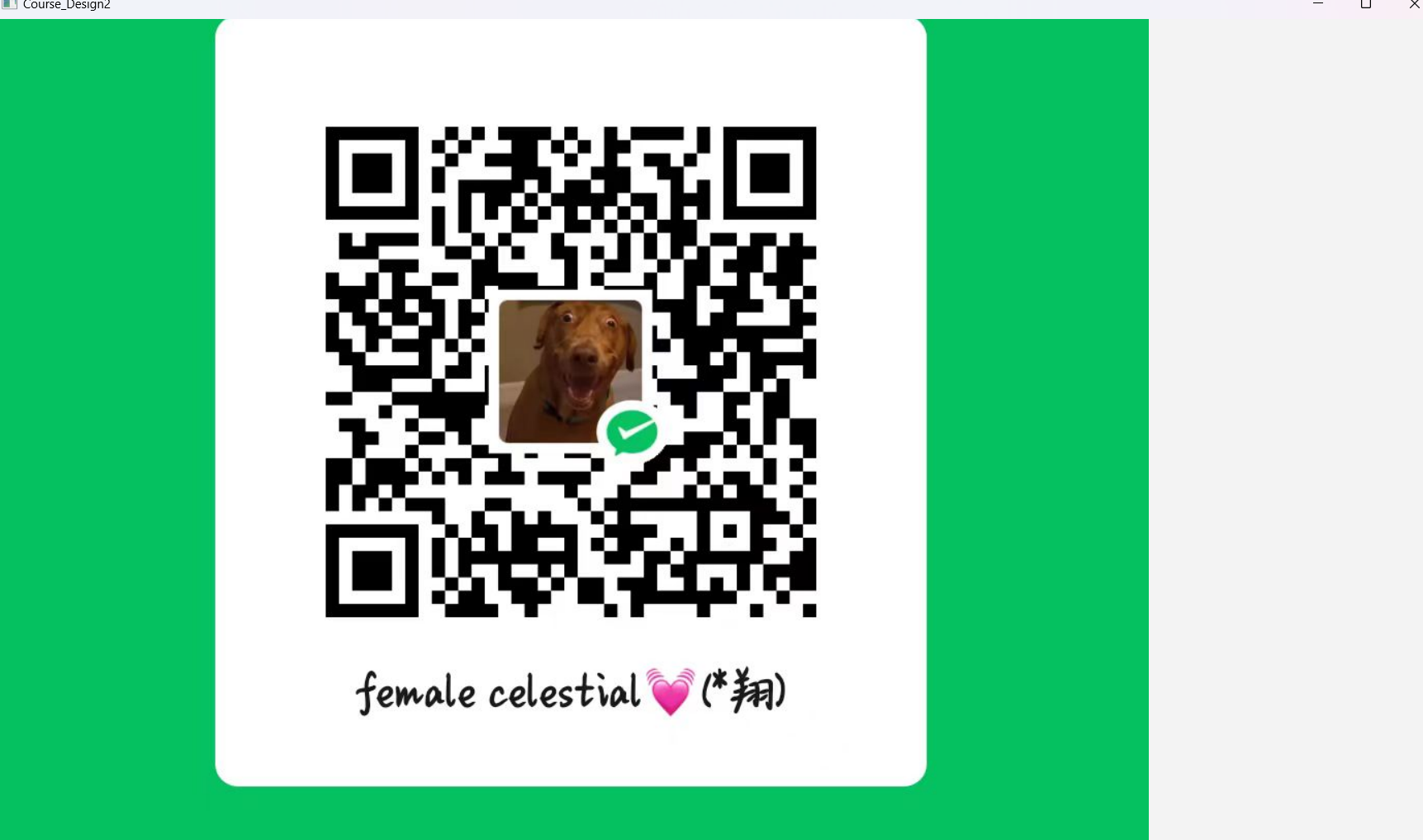


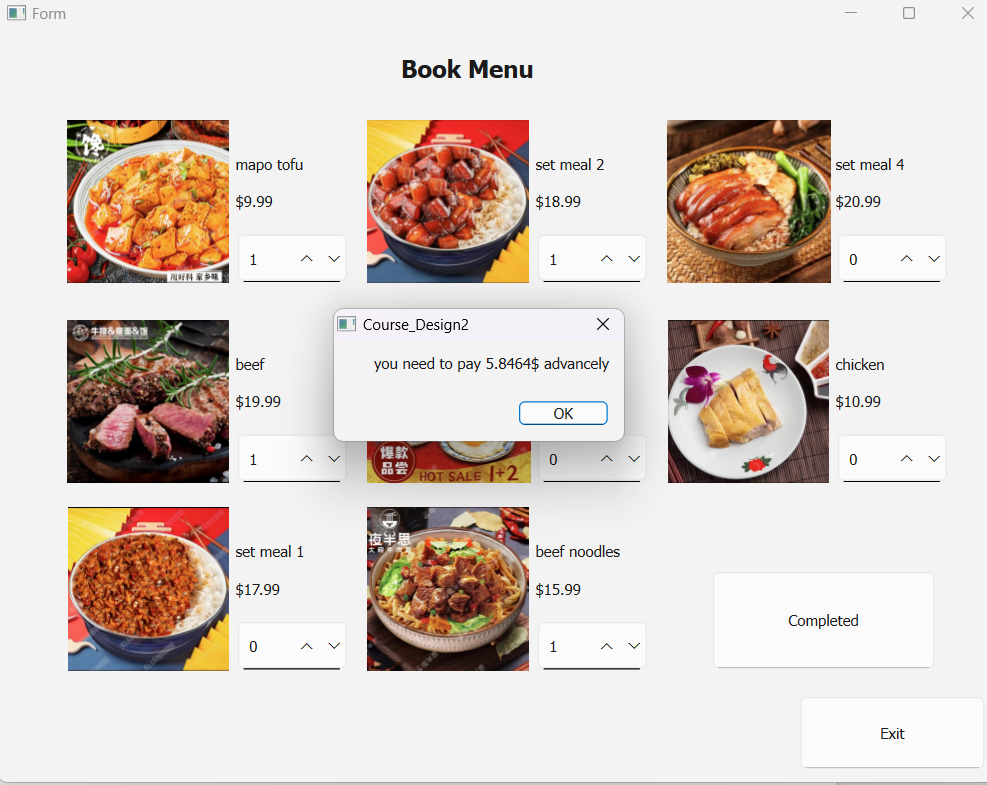
屏幕截图 2024-06-15 203342

1. 预定菜单：可以选择菜品，同时提醒顾客有折扣，计算出预定需要支付的价格，预支付价格，菜单，剩余支付价格保存至book\_menu文本文档，并跳出支付二维码。



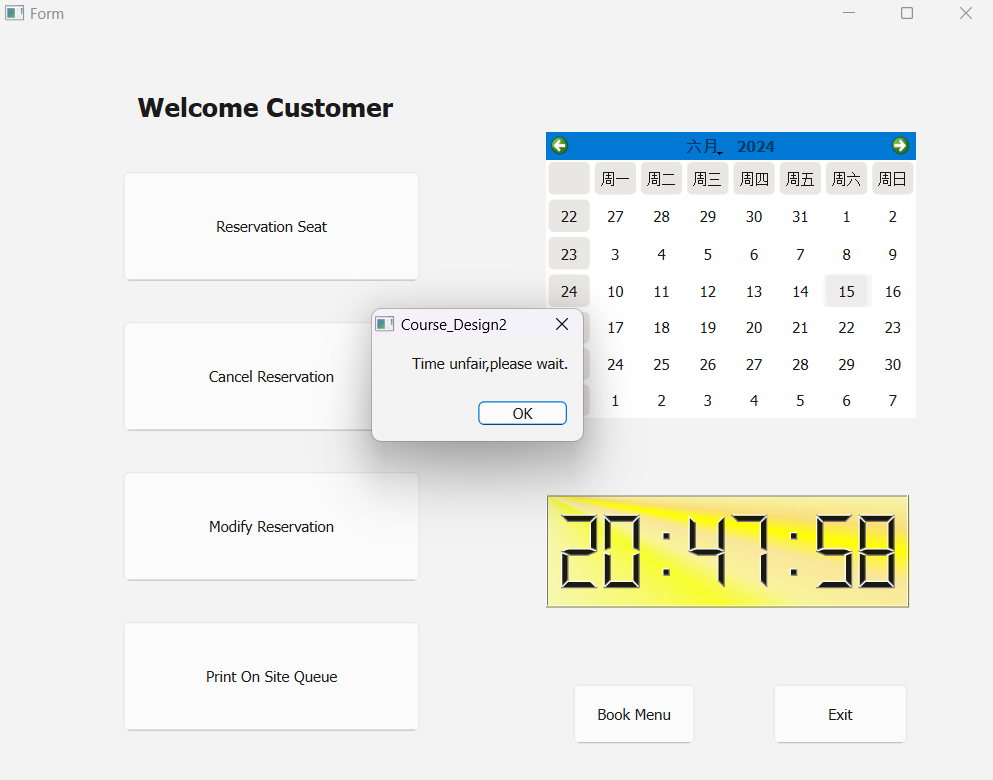


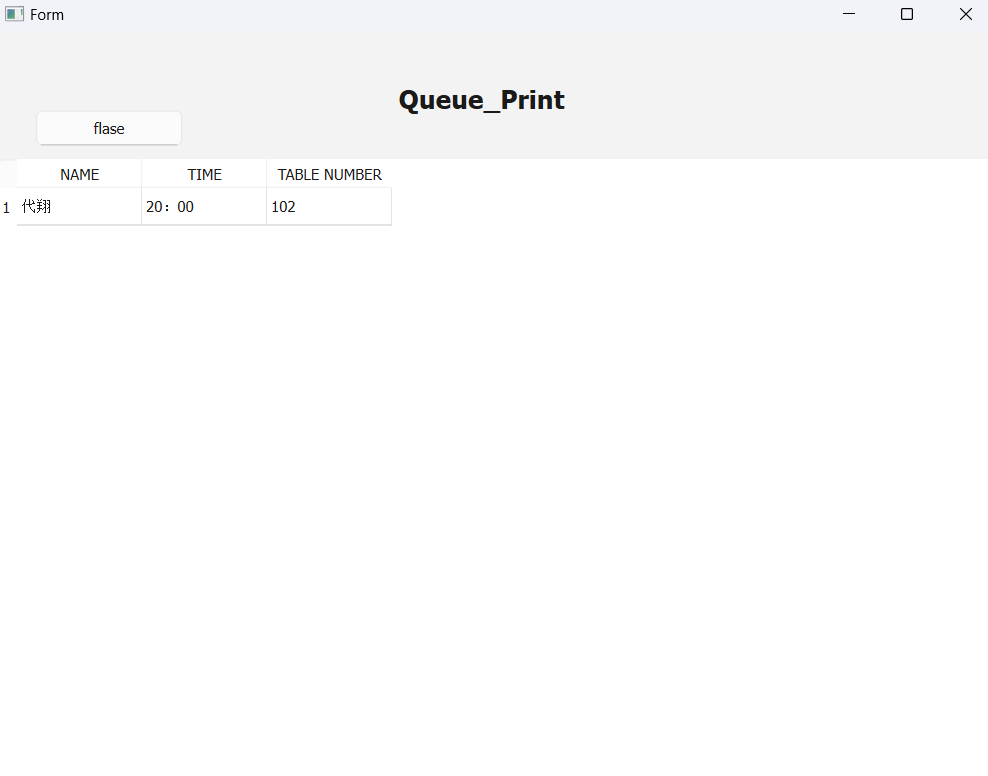




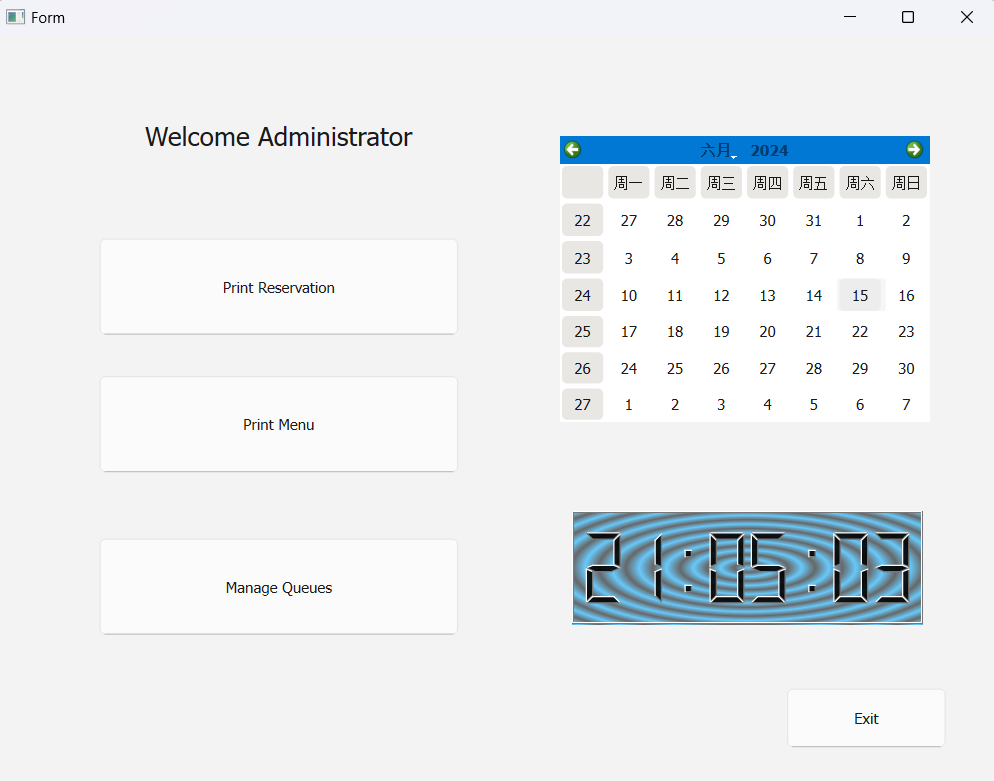
屏幕截图 2024-06-15 204226

1. 打印线下排队队列：只有在特定时间（10：00-12：30以及17：00-19：30）可以打印（可以刷新页面），否则会告诉你需要等待。

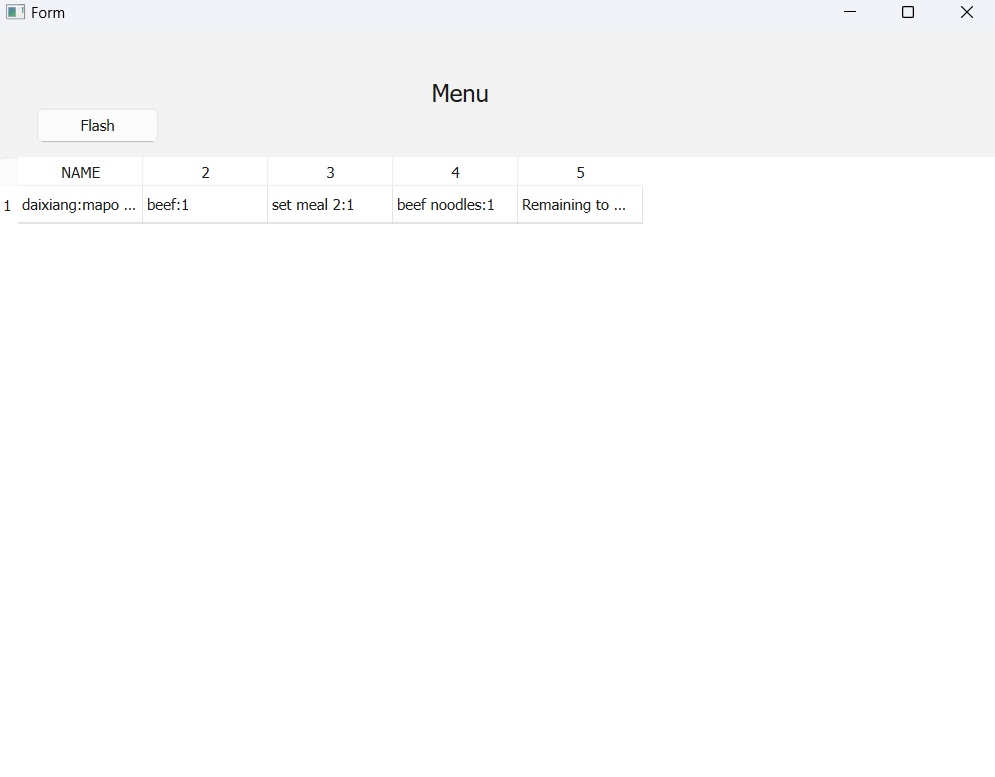


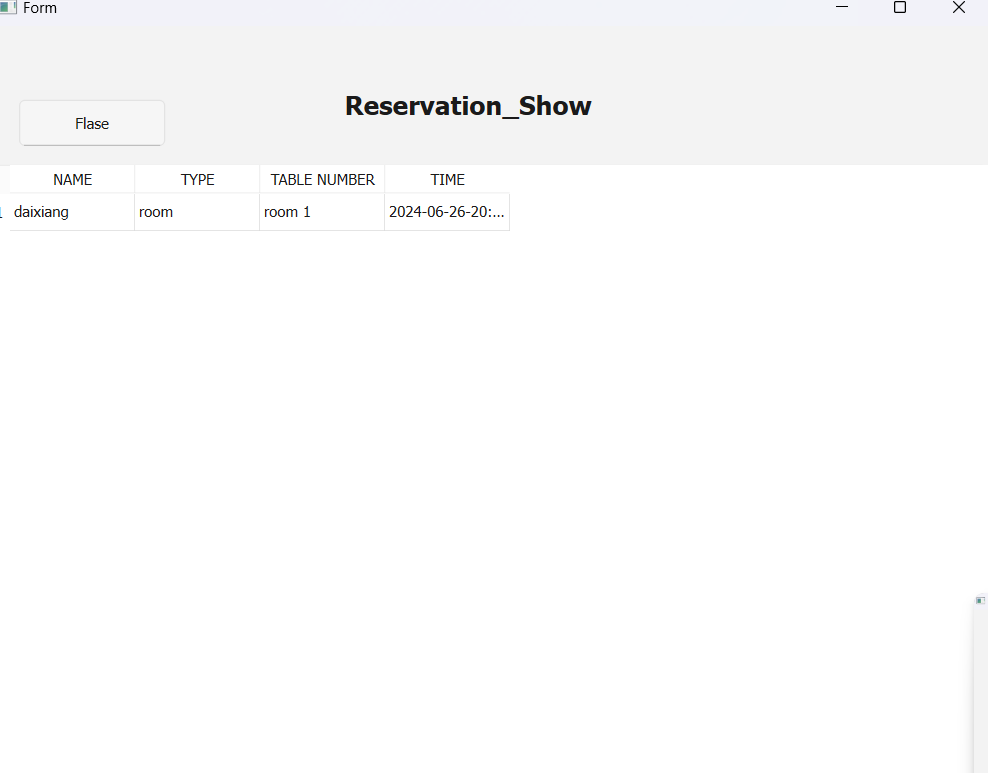


1. 管理员界面（主界面设计了一个日历和时间LED方便管理员确定时间）

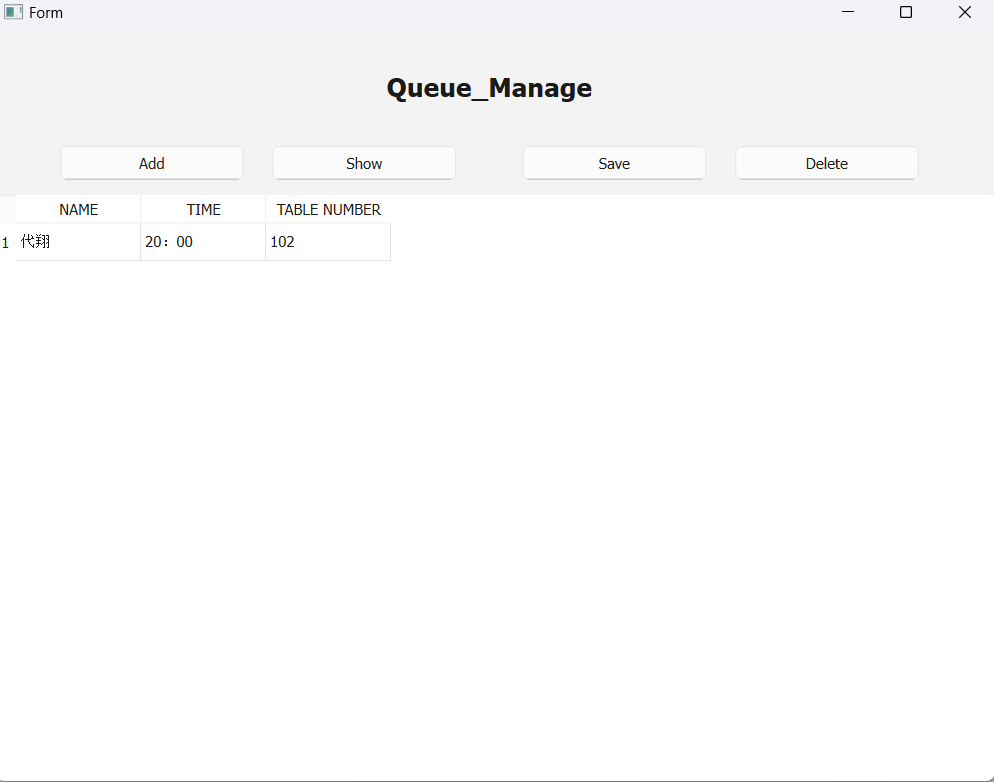


1. 打印预定单队列以及菜单队列：





1. 管理线下队列，增删改以及保存功能，并同步到Queue文本文档。



1. 心得与体会

在本代码的设计与实现过程中，第一个遇到的问题是不会使用数据库，在文本文档之中存入信息找不到方法进行合适的分割和读取，后面在不断的思考中，想到可以使用一些特定的符号作为分隔符，来将存入的信息进行分类，从而实现了对数据内容的读取与录入。

其次遇到的问题便是在预定桌号的时候，我对选择桌号设定的是checkbox类型，而选择类型设定的是radiobutton类型，由于checkbox的选择是自由的，从而会导致这种情况：假设你选择的类型为四人桌，但你可以去预定包厢或者二人桌号，这是一个bug。但我觉得如果加入先后页面，从而实现选择radiobutton再选择号码会使页面变得繁冗，所以我不选择这个方案。所以我去qt自带的帮助中查询checkbox，发现它可以通过setEnabled设置他无法被选中。然后我在界面初始化的时候将checkbox全部初始化为无法选中，并建立函数将部分的checkbox解锁，再通过槽函数将radiobutton的状态和函数以及checkbox连接起来，从而解决了这个问题。

紧接着，就是对于时间设定的问题，由于用惯了c++的windows.h中的SYSTEMTIME类型的时间展示，所以一开始就一直使用SYSTEMTIME类型的数据，后面在设计如何设定预定时间以及时间段的判定的时候出现了显示错误，qt报错发现不能LED输出，而且SYSTEMTIME里面的数据类型是特殊的DATE，他无法与数字进行比较，所以我提问了AI是否有可以吧DATE类型转为int类型的函数，AI给我编写了一个函数，这个函数很复杂而且在qt环境下运行不了，随后我再提问相同的问题，AI的回答前多了一点内容引起了我的注意：qt有自己表达时间的形式QDATETIME，所以我去qt的帮助中查询了这类形式，发现他有许多内置函数可以完成我对时间设定和显示的要求，并且其类型也为int类型，可以很好地完成我对时间段判定的要求。

再者就是我想在点菜单页面增加图片内容使点菜页面更优美，但我并不知道如何实现这个功能，于是我去B站搜索了相关教程，最后成功地在点菜页面插入了图片。

最后，在运行测试阶段，在输入手机号的检验过程的测试中发现一旦输入不合法的手机号，会一直弹出弹出弹窗，导致系统无法运行。结束运行，去检查代码发现在检查时候用了while循环，但在一个循环后没有加入去修改的代码，从而产生这样的问题，所以我加上了修改的代码，最后解决了这个bug。

参与这次课设，我深感编程的魅力和挑战。通过实际操作，我不仅加深了对QT框架的理解，还锻炼了解决实际问题的能力。面对复杂的界面设计和功能实现，我不断尝试、修正，最终成功完成了项目。这次经历让我认识到，耐心和坚持是编程路上不可或缺的品质。