OBJECT ORIENTED PROGRAMMING

Final Project Report

Railway Ticketing System

Submitted to: Erum Ashraf

Submitted By:

Saad Ahmad (01-134222-130)

Sohaib Ahmed (01-134222-132)

Saaid Jameel (01-134222-134)

Contents

Project Description

The Railway ticketing System is a software application designed to efficiently manage and automate various operations related to railway services. This system aims to streamline ticket booking, passenger information management, train scheduling, and other related tasks to ensure smooth railway operations.

Project Functionalities

Number	***Description of Functional Requirement***
1	Automatic Routing
2	Ticket Generation
3	Travel Classes
4	Saving Train Logs on SQL
5	Menu Selection
6	Password Encryption using MD5 hashing

Significance of the Project

Our group is dedicated to developing robust Railway Ticketing System using Object-Oriented Programming (OOP) principles. This software application aims to revolutionize the management and operation of railway services. By leveraging OOP, we are creating an efficient and automated system that streamlines ticket booking, passenger information management, train scheduling, and other critical operations in the railway industry.

The Railway Ticketing System will serve as a comprehensive solution to enhance the overall efficiency and effectiveness of railway services. Through the use of modern software development practices and methodologies, our goal is to create a user-friendly and scalable system that caters to the diverse needs of railway authorities, staff, and passengers.

By automating manual processes and providing real-time information, our Railway Ticketing System empowers railway authorities to make informed decisions, optimize resource allocation, and improve service quality. Passengers will benefit from a seamless and hassle-free experience, from booking tickets to accessing timely information about train schedules and availability.

With a strong emphasis on OOP principles, we are creating a flexible and modular system that can easily adapt to future requirements and technological advancements. The Railway Ticketing System is poised to revolutionize railway operations, enhancing efficiency, reliability, and customer satisfaction in the ever-evolving transportation industry.

Header Files

#include “bill.h”

```
#ifndef BILL_H
#define BILL_H

#include <QDialog>
#include "database.h"
#include "traininfo.h"
namespace Ui {
class Bill;
}

class Bill : public QDialog
{
   Q_OBJECT

public:
   explicit Bill(QWidget *parent = nullptr,trainInfo a = trainInfo());
   ~Bill();
   void printContent();
   void setTotal(int a){
      total=a;
   }
   int getTotal(){
      return total;
   }
   // Getter method for 'name'
   QString getName() {
      return name;
   }

   // Setter method for 'name'
   void setName(QString newName) {
      name = newName;
   }

   // Getter method for 'ticket'
   int getTicket() {
      return ticketno;
   }

   // Setter method for 'ticket'
   void setTicketno(int newTicket) {
```

```
        ticketno = newTicket;
    }

private slots:
    void on_pushButton_clicked();

private:
    Ui::Bill *ui;
    trainInfo t1;
    int total;
    QString name;
    int ticketno;

};

#endif // BILL_H
```

#include "booking.h"

```
#ifndef BOOKING_H
#define BOOKING_H

#include <QDialog>
#include "train.h"
#include "traininfo.h"
namespace Ui {
class Booking;
}

class Booking : public QDialog,public trainInfo
{
    Q_OBJECT

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


private slots:
    void on_pushButton_clicked();

    void on_comboBox_currentTextChanged(const QString &arg1);

private:
    Ui::Booking *ui;
};
```

```
#endif // BOOKING_H
```

#include “captcha.h”

```
#ifndef CAPTCHA_H
#define CAPTCHA_H

#include <QObject>
#include <QFont>
#include <QImage>
#include <QTime>
#include <QVector>
#include <QStringList>
#include <QDebug>
#include <QFile>
#include <QTextStream>
#include <QPainter>
#include <QPainterPath>
#include <QtMath>

class Captcha : public QObject
{
   Q_OBJECT
   Q_ENUMS(DeformType)
   Q_ENUMS(TextGenerationMode)
   Q_PROPERTY(QFont font READ font WRITE setFont)
   Q_PROPERTY(QImage captchaImage READ captchaImage)
   Q_PROPERTY(QString captchaText READ captchaText WRITE setCaptchaText)
   Q_PROPERTY(DeformType deformationType READ deformationType WRITE
setDeformationType)
   Q_PROPERTY(TextGenerationMode textGeneration READ textGeneration WRITE
setTextGeneration)
   Q_PROPERTY(QStringList dictionary READ dictionary WRITE setDictionary)
   Q_PROPERTY(QColor fontColor READ fontColor WRITE setFontColor)
   Q_PROPERTY(QColor backColor READ backColor WRITE setBackColor)
   Q_PROPERTY(bool drawLines READ drawLines WRITE setDrawLines)
   Q_PROPERTY(bool drawEllipses READ drawEllipses WRITE setDrawEllipses)
   Q_PROPERTY(bool drawNoise READ drawNoise WRITE setDrawNoise)
   Q_PROPERTY(int noiseCount READ noiseCount WRITE setNoiseCount)
   Q_PROPERTY(int lineCount READ lineCount WRITE setLineCount)
   Q_PROPERTY(int ellipseCount READ ellipseCount WRITE setEllipseCount)
   Q_PROPERTY(int lineWidth READ lineWidth WRITE setLineWidth)
   Q_PROPERTY(int ellipseMinRadius READ ellipseMinRadius WRITE setEllipseMinRadius)
   Q_PROPERTY(int ellipseMaxRadius READ ellipseMaxRadius WRITE
setEllipseMaxRadius)
   Q_PROPERTY(int noisePointSize READ noisePointSize WRITE setNoisePointSize)
```

```
public:
    enum DeformType
    {
        Deform_SinCurve
    };

    enum TextGenerationMode
    {
        TextGeneration_Random,
        TextGeneration_Dictionary
    };

public:

    explicit Captcha(QObject *parent = 0);

    QFont font() const;
    QImage captchaImage() const;
    DeformType deformationType() const;
    QString captchaText() const;
    TextGenerationMode textGeneration() const;
    const QStringList &dictionary() const;
    QColor fontColor() const;
    QColor backColor() const;
    bool drawLines() const;
    bool drawEllipses() const;
    bool drawNoise() const;
    int noiseCount() const;
    int lineCount() const;
    int ellipseCount() const;
    int lineWidth() const;
    int ellipseMinRadius() const;
    int ellipseMaxRadius() const;
    int noisePointSize() const;

signals:
    void catpchaGenerated(const QImage& img, QString text);

public slots:
    void setFont(const QFont& arg);
    void setDeformationType(DeformType arg);
    void updateCaptcha();
    void randomize();
    void generateText(int noOfChars = 5, bool includeNumbers = false, bool includeSymbols =
false, bool allCapital = true);
```

```
    void setCaptchaText(QString arg);
    void setTextGeneration(TextGenerationMode arg);
    void setDictionary(const QStringList& arg);
    void loadDictionary(QString FileName);
    void setFontColor(QColor arg);
    void setBackColor(QColor arg);
    void setSinDeform(qreal hAmplitude, qreal hFrequency, qreal vAmplitude, qreal
vFrequency);
    QPair<QString, QImage> generateCaptcha();
    void setDrawLines(bool arg);
    void setDrawEllipses(bool arg);
    void setDrawNoise(bool arg);
    void setNoiseCount(int arg);
    void setLineCount(int arg);
    void setEllipseCount(int arg);
    void setLineWidth(int arg);
    void setEllipseMinRadius(int arg);
    void setEllipseMaxRadius(int arg);
    void setNoisePointSize(int arg);
    void setDifficulty(int val);

private:
    qreal m_hmod1;
    qreal m_hmod2;

    qreal m_vmod1;
    qreal m_vmod2;

    QFont m_font;
    QImage m_captchaImage;
    DeformType m_deformationType;
    QString m_captchaText;
    TextGenerationMode m_textGeneration;
    QStringList m_dictionary;
    QColor m_fontColor;
    QColor m_backColor;
    qreal m_padding;
    bool m_drawLines;
    bool m_drawEllipses;
    bool m_drawNoise;
    int m_noiseCount;
    int m_lineCount;
    int m_ellipseCount;
    int m_lineWidth;
    int m_ellipseMinRadius;
    int m_ellipseMaxRadius;
```

```
    int m_noisePointSize;
};

#endif // CAPTCHA_H
```

#include “database.h”

```
#ifndef DATABASE_H
#define DATABASE_H
#include <QtSql/QSqlDatabase>
#include <QDebug>
class Database
{
public:
    Database();
    QSqlDatabase mydb;
    void connClose(){

        mydb.close();
        mydb.removeDatabase(QSqlDatabase::defaultConnection);
    }
    bool connOpen()
{
        mydb = QSqlDatabase::addDatabase("QSQLITE");
        mydb.setDatabaseName("C:/Users/Saad/Desktop/Database/customer.db");
        if(!mydb.open()){
            qDebug() << "NOT OPEN";
            return false;
        }
        else{
            qDebug() << "OPEN";
            return true;
        }
    }
};

#endif // DATABASE_H
```

#include “dailog.h”

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QDialog>
#include "captcha.h"
#include <QPaintEvent>
```

```
namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();

private:
    Ui::Dialog *ui;

    // QWidget interface
protected:
    virtual void paintEvent(QPaintEvent *);
private slots:
    void on_pushButton_clicked();
    void on_lineEdit_textChanged(const QString &arg1);
};

#endif // DIALOG_H
```

#include "mainwindow.h"

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtSql/QSqlDatabase>
#include "signup.h"
#include "database.h"
#include "ticket.h"
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }

QT_END_NAMESPACE
extern QString Phoneno;
class Login{
private:
    QString Phone;
    QString Password;
    public:
    Login(QString a="",QString b=""){
```

```
        setPhone(a);
        setPassword(b);
    }
    void setPhone(QString a){
        Phone =a;
    }
    void setPassword(QString a){
        Password =a;
    }
    QString getPhone(){
        return Phone;
    }
    QString getPassword(){
        return Password;
    }
};

class MainWindow : public QMainWindow,public Login
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();


private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

#include "payment.h"

```
#ifndef PAYMENT_H
#define PAYMENT_H

#include <QDialog>
#include "traininfo.h"
namespace Ui {
class Payment;
}
```

```
class Payment : public QDialog
{
    Q_OBJECT

public:
    explicit Payment(QWidget *parent = nullptr,trainInfo t = trainInfo());
    ~Payment();

private slots:
    void on_radioButton_clicked();

    void on_radioButton_2_clicked();

    void on_radioButton_3_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_clicked();

private:
    Ui::Payment *ui;
    trainInfo t1;
};

#endif // PAYMENT_H
```

#include "seats.h"

```
#ifndef SEATS_H
#define SEATS_H

#include <QDialog>
#include "traininfo.h"
#include "database.h"
#include "payment.h"
namespace Ui {
class Seats;
}
class Seats : public QDialog
{
    Q_OBJECT

public:
    explicit Seats(QWidget *parent = nullptr,trainInfo t1= trainInfo());
    ~Seats();
```

```
private slots:
    void on_checkBox_stateChanged(int arg1);

    void on_pushButton_clicked();

    void on_checkBox_2_stateChanged(int arg1);

    void on_c1_stateChanged(int arg1);

    void on_c3_stateChanged(int arg1);

    void on_c2_stateChanged(int arg1);

    void on_c4_stateChanged(int arg1);

    void on_c5_stateChanged(int arg1);

    void on_c6_stateChanged(int arg1);

    void on_c7_stateChanged(int arg1);

    void on_c8_stateChanged(int arg1);

    void on_c9_stateChanged(int arg1);

    void on_c10_stateChanged(int arg1);

    void on_c11_stateChanged(int arg1);

    void on_c12_stateChanged(int arg1);

    void on_c13_stateChanged(int arg1);

    void on_c14_stateChanged(int arg1);

    void on_c15_stateChanged(int arg1);

    void on_c16_stateChanged(int arg1);

    void on_c20_stateChanged(int arg1);

    void on_c17_stateChanged(int arg1);

    void on_c18_stateChanged(int arg1);

    void on_c19_stateChanged(int arg1);
```

```
void on_c21_stateChanged(int arg1);

void on_c22_stateChanged(int arg1);

void on_c23_stateChanged(int arg1);

void on_c24_stateChanged(int arg1);

void on_c25_stateChanged(int arg1);

void on_c26_stateChanged(int arg1);

void on_c27_stateChanged(int arg1);

void on_c28_stateChanged(int arg1);

void on_c29_stateChanged(int arg1);

void on_c30_stateChanged(int arg1);

void on_c31_stateChanged(int arg1);

void on_c32_stateChanged(int arg1);

void on_c33_stateChanged(int arg1);

void on_c34_stateChanged(int arg1);

void on_c35_stateChanged(int arg1);

void on_c36_stateChanged(int arg1);

void on_c37_stateChanged(int arg1);

void on_c38_stateChanged(int arg1);

void on_c39_stateChanged(int arg1);

void on_c40_stateChanged(int arg1);
int getBill() const{
   return Bill;
}
void setBill(int bill){
   Bill=bill;
}
```

```
private:
    Ui::Seats *ui;
    trainInfo t;
int Bill;
};

#endif // SEATS_H
```

#include “singup.h”

```
#ifndef SIGNUP_H
#define SIGNUP_H

#include <QDialog>
#include "dialog.h"
#include "database.h"
namespace Ui {
class SignUp;
}
class Sign_UP{
private:
    QString Email;
    QString CheckPass;
    QString Name;
    QString Phone;
    QString Password;
    public:
    Sign_UP(QString a="",QString b="",QString c="",QString d="",QString e=""){

        setName(a);
        setPhone(b);
        setEmail(c);
        setPassword(d);
        setCheckpass(e);

    }
    void setEmail(QString a){
        Email =a;
    }
    void setCheckpass(QString a){
        CheckPass =a;
    }
    void setName(QString a){
        Name =a;
    }
    void setPhone(QString a){
        Phone =a;
```

```
    }
    void setPassword(QString a){
        Password =a;
    }
    QString getEmail(){
        return Email;
    }
    QString getCheckpass(){
        return CheckPass;
    }
    QString getName(){
        return Name;
    }
    QString getPhone(){
        return Phone;
    }
    QString getPassword(){
        return Password;
    }


};
class SignUp : public QDialog,public Sign_UP
{
    Q_OBJECT

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

private slots:
    void on_pushButton_clicked();

private:
    Ui::SignUp *ui;
};

#endif // SIGNUP_H
```

#include “ticket.h”

```
#ifndef TICKET_H
#define TICKET_H

#include <QDialog>
#include "booking.h"
namespace Ui {
```

```
class Ticket;
}

class Ticket : public QDialog
{
    Q_OBJECT

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

private slots:
    void on_pushButton_clicked();

private:
    Ui::Ticket *ui;
};

#endif // TICKET_H
```

#include “train.h”

```
#ifndef TRAIN_H
#define TRAIN_H

#include <QDialog>
#include "traininfo.h"
#include "database.h"

namespace Ui {
class Train;
}

class Train : public QDialog
{
    Q_OBJECT

public:
    explicit Train(QWidget *parent = nullptr,trainInfo a = trainInfo());
    ~Train();
    // Getter and Setter for time

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();
```

```
    void on_pushButton_3_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_5_clicked();

private:
    Ui::Train *ui;
trainInfo t1;

};

#endif // TRAIN_H
```

#include “traininfo.h”

```
#ifndef TRAININFO_H
#define TRAININFO_H
#include <QDialog>
class trainInfo {
private:
    QString Source;
    QString Destination;
    QString Date;
    QString Passengers;
    QString Time;
    QString Type;
    QString Price;
public:
    // Constructor
    trainInfo(QString source = "", QString destination = "", QString date = "", QString passengers
= "",QString time="",QString typ="",
            QString price="")
        : Source(source), Destination(destination), Date(date),
Passengers(passengers),Time(time),Type(typ), Price(price){
    }

    // Getter methods
    QString getSource() const {
        return Source;
    }

    QString getDestination() const {
        return Destination;
    }

    QString getDate() const {
```

```
    return Date;
}

QString getPassengers() const {
    return Passengers;
}

// Setter methods
void setSource(QString source) {
    Source = source;
}

void setDestination(QString destination) {
    Destination = destination;
}

void setDate(QString date) {
    Date = date;
}

void setPassengers(QString passengers) {
    Passengers = passengers;
}
QString getTime() const {
    return Time;
}

void setTime(QString newTime) {
    Time = newTime;
}

// Getter and Setter for t_Class
QString getType() const {
    return Type;
}

void setType( QString newType) {
    Type = newType;
}

// Getter and Setter for price
QString getPrice() const {
    return Price;
}

void setPrice(QString newPrice) {
```

```
        Price = newPrice;
    }

};

#endif // TRAININFO_H
```

Code

Booking.cpp

```
#include "booking.h"
#include "ui_booking.h"

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

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

void Booking::on_pushButton_clicked()
{
    QDate selectedDate = ui->calendarWidget->selectedDate();
    QString date = selectedDate.toString("dd-MM-yyyy");
    trainInfo t(ui->comboBox->currentText(),ui->comboBox_2->currentText(),date,ui->spinBox-
>text());
    this->hide();
    Train t1(nullptr,t);
    t1.setModal(true);
    t1.exec();
}

void Booking::on_comboBox_currentTextChanged(const QString &arg1)
{
    QString selectedCity = ui->comboBox->currentText();

    if (selectedCity == "Islamabad") {
        ui->comboBox_2->clear();
```

```
        ui->comboBox_2->addItem("Lahore");
        ui->comboBox_2->addItem("Karachi");
    } else if (selectedCity == "Karachi") {
        ui->comboBox_2->clear();
        ui->comboBox_2->addItem("Islamabad");
        ui->comboBox_2->addItem("Lahore");
    } else if (selectedCity == "Lahore") {
        ui->comboBox_2->clear();
        ui->comboBox_2->addItem("Islamabad");
        ui->comboBox_2->addItem("Karachi");
    }
}
```

Captcha.cpp

```
#include "captcha.h"
Captcha::Captcha(QObject *parent) :
    QObject(parent)
{
    m_hmod1 = 0.0;
    m_hmod2 = 0.0;

    m_vmod1 = 0.0;
    m_vmod2 = 0.0;

    m_font.setStyleStrategy(QFont::ForceOutline);
    m_font.setPointSize(30);
    m_font.setBold(true);
    m_font.setLetterSpacing(QFont::PercentageSpacing, QFont::SemiCondensed);

    m_captchaImage = QImage(200, 100, QImage::Format_RGB32);
    m_deformationType = Deform_SinCurve;
    m_captchaText = "Test";

    m_textGeneration = TextGeneration_Random;
    m_fontColor = Qt::black;
    m_backColor = Qt::white;
    m_padding = 5;

    m_drawLines = true;
    m_drawEllipses = true;
    m_drawNoise = true;
    m_noiseCount = 100;
    m_lineCount = 5;
    m_ellipseCount = 1;

    m_lineWidth = 2;
```

```
    m_ellipseMinRadius = 20;
    m_ellipseMaxRadius = 40;

    m_noisePointSize = 3;

    setSinDeform(8, 10, 5, 15);
}

void Captcha::setDifficulty(int val)
{
    if (val == 0)
    {
        m_drawLines = false;
        m_drawEllipses = false;
        m_drawNoise = false;
        setSinDeform(10, 10, 5, 20);
    }
    else if (val == 1)
    {
        m_drawLines = true;
        m_lineWidth = 3;
        m_lineCount = 5;
        m_drawEllipses = false;
        m_drawNoise = false;
        setSinDeform(10, 15, 5, 20);
    }
    else if (val == 2)
    {
        m_drawLines = true;
        m_lineWidth = 2;
        m_lineCount = 5;
        m_drawEllipses = true;
        m_ellipseCount = 1;
        m_ellipseMinRadius = 20;
        m_ellipseMaxRadius = 40;
        m_drawNoise = false;
        setSinDeform(10, 15, 5, 15);
    }
    else if (val == 3)
    {
        m_drawLines = true;
        m_lineWidth = 2;
        m_lineCount = 3;
        m_drawEllipses = true;
        m_ellipseCount = 1;
        m_ellipseMinRadius = 20;
```

```
        m_ellipseMaxRadius = 40;
        m_drawNoise = true;
        m_noiseCount = 100;
        m_noisePointSize = 3;
        setSinDeform(8, 13, 5, 15);
    }
    else if (val == 4)
    {
        m_drawLines = true;
        m_lineWidth = 3;
        m_lineCount = 5;
        m_drawEllipses = true;
        m_ellipseCount = 1;
        m_ellipseMinRadius = 20;
        m_ellipseMaxRadius = 40;
        m_drawNoise = true;
        m_noiseCount = 100;
        m_noisePointSize = 3;
        setSinDeform(8, 10, 5, 10);
    }
    else
    {
        m_drawLines = true;
        m_lineWidth = 4;
        m_lineCount = 7;
        m_drawEllipses = true;
        m_ellipseCount = 1;
        m_ellipseMinRadius = 20;
        m_ellipseMaxRadius = 40;
        m_drawNoise = true;
        m_noiseCount = 200;
        m_noisePointSize = 3;
        setSinDeform(8, 10, 5, 10);
    }
}

QFont Captcha::font() const
{
    return m_font;
}

QImage Captcha::captchaImage() const
{
    return m_captchaImage;
}
```

```
Captcha::DeformType Captcha::deformationType() const
{
    return m_deformationType;
}

QString Captcha::captchaText() const
{
    return m_captchaText;
}

Captcha::TextGenerationMode Captcha::textGeneration() const
{
    return m_textGeneration;
}

const QStringList &Captcha::dictionary() const
{
    return m_dictionary;
}
QColor Captcha::fontColor() const
{
    return m_fontColor;
}

QColor Captcha::backColor() const
{
    return m_backColor;
}

bool Captcha::drawLines() const
{
    return m_drawLines;
}

bool Captcha::drawEllipses() const
{
    return m_drawEllipses;
}

bool Captcha::drawNoise() const
{
    return m_drawNoise;
}

int Captcha::noiseCount() const
{
```

```
    return m_noiseCount;
}

int Captcha::lineCount() const
{
    return m_lineCount;
}

int Captcha::ellipseCount() const
{
    return m_ellipseCount;
}

int Captcha::lineWidth() const
{
    return m_lineWidth;
}

int Captcha::ellipseMinRadius() const
{
    return m_ellipseMinRadius;
}

int Captcha::ellipseMaxRadius() const
{
    return m_ellipseMaxRadius;
}

int Captcha::noisePointSize() const
{
    return m_noisePointSize;
}

void Captcha::setFont(const QFont &arg)
{
    m_font = arg;
}

void Captcha::setDeformationType(Captcha::DeformType arg)
{
    m_deformationType = arg;
}

void Captcha::updateCaptcha()
{
    QPainterPath path;
```

```
    QFontMetrics fm(m_font);

    if (m_deformationType == Deform_SinCurve)
    {

        path.addText(m_vmod2 + m_padding, m_hmod2 - m_padding + fm.height(), font(), captchaText());

        qreal sinrandomness = ((qreal) rand() / RAND_MAX) * 5.0;

        for (int i = 0; i < path.elementCount(); ++i)
        {
            const QPainterPath::Element& el = path.elementAt(i);
            qreal y = el.y + sin(el.x / m_hmod1 + sinrandomness) * m_hmod2;
            qreal x = el.x + sin(el.y / m_vmod1 + sinrandomness) * m_vmod2;
            path.setElementPositionAt(i, x, y);
        }

        m_captchaImage = QImage(fm.horizontalAdvance(m_captchaText) + m_vmod2 * 2 + m_padding * 2, fm.height() + m_hmod2 * 2 + m_padding * 2, QImage::Format_RGB32);
    }

    m_captchaImage.fill(backColor());
    QPainter painter;
    painter.begin(&m_captchaImage);
    painter.setPen(Qt::NoPen);
    painter.setBrush(fontColor());
    painter.setRenderHint(QPainter::Antialiasing);
    painter.drawPath(path);

    if (m_drawLines)
    {
        painter.setPen(QPen(Qt::black, m_lineWidth));
        for (int i = 0; i < m_lineCount; i++)
        {
            int x1 = ((qreal) rand() / RAND_MAX) * m_captchaImage.width();
            int y1 = ((qreal) rand() / RAND_MAX) * m_captchaImage.height();
            int x2 = ((qreal) rand() / RAND_MAX) * m_captchaImage.width();
            int y2 = ((qreal) rand() / RAND_MAX) * m_captchaImage.height();
            painter.drawLine(x1, y1, x2, y2);
        }
        painter.setPen(Qt::NoPen);
    }

    if (m_drawEllipses)
    {
```

```
        for (int i = 0; i < m_ellipseCount; i++)
        {
            int x1 = m_ellipseMaxRadius / 2.0 + ((qreal) rand() / RAND_MAX) * (m_captchaImage.width() - m_ellipseMaxRadius);
            int y1 = m_ellipseMaxRadius / 2.0 + ((qreal) rand() / RAND_MAX) * (m_captchaImage.height() - m_ellipseMaxRadius);
            int rad1 = m_ellipseMinRadius + ((qreal) rand() / RAND_MAX) * (m_ellipseMaxRadius - m_ellipseMinRadius);
            int rad2 = m_ellipseMinRadius + ((qreal) rand() / RAND_MAX) * (m_ellipseMaxRadius - m_ellipseMinRadius);
            painter.setBrush(backColor());
            painter.setCompositionMode(QPainter::CompositionMode_Difference);
            painter.drawEllipse(QPoint(x1, y1), rad1, rad2);
        }
    }

    if (m_drawNoise)
    {
        for (int i = 0; i < m_noiseCount; i++)
        {
            int x1 = ((qreal) rand() / RAND_MAX) * m_captchaImage.width();
            int y1 = ((qreal) rand() / RAND_MAX) * m_captchaImage.height();

            QColor col = QColor(((qreal) rand() / RAND_MAX) * 255, ((qreal) rand() / RAND_MAX) * 255, ((qreal) rand() / RAND_MAX) * 255);

            painter.setPen(QPen(col, m_noisePointSize));
            painter.setCompositionMode(QPainter::CompositionMode_SourceOver);
            painter.drawPoint(x1, y1);
        }
    }
    painter.end();
    emit catpchaGenerated(m_captchaImage, m_captchaText);
}

void Captcha::randomize()
{
    srand(QTime::currentTime().msec());
}

void Captcha::setCaptchaText(QString arg)
{
    m_captchaText = arg;
}

void Captcha::setTextGeneration(Captcha::TextGenerationMode arg)
```

```
{
    if (m_textGeneration != arg) generateText(m_captchaText.size());
    m_textGeneration = arg;
}

void Captcha::setDictionary(const QStringList &arg)
{
    m_dictionary = arg;
}

void Captcha::loadDictionary(QString FileName)
{
    QFile file(FileName);
    if (!file.open(QIODevice::ReadOnly))
    {
        qCritical() << "Unable to open dictionary file";
        return;
    }

    m_dictionary.clear();
    QTextStream text(&file);
    QString str = text.readLine();
    while (str.size() > 0)
    {
        m_dictionary.append(str);
        str = text.readLine();
    }

    if (m_dictionary.size() <= 0)
    {
        qWarning() << "No data loaded from dictionary file";
    }
}

void Captcha::setFontColor(QColor arg)
{
    m_fontColor = arg;
}

void Captcha::setBackColor(QColor arg)
{
    m_backColor = arg;
}

void Captcha::setSinDeform(qreal hAmplitude, qreal hFrequency, qreal vAmplitude, qreal
vFrequency)
```

```
{
    m_deformationType = Deform_SinCurve;
    m_hmod1 = hFrequency;
    m_hmod2 = hAmplitude;

    m_vmod1 = vFrequency;
    m_vmod2 = vAmplitude;
}

QPair<QString, QImage> Captcha::generateCaptcha()
{
    generateText(m_captchaText.size());
    return QPair<QString, QImage>(m_captchaText, m_captchaImage);
}

void Captcha::setDrawLines(bool arg)
{
    m_drawLines = arg;
}

void Captcha::setDrawEllipses(bool arg)
{
    m_drawEllipses = arg;
}

void Captcha::setDrawNoise(bool arg)
{
    m_drawNoise = arg;
}

void Captcha::setNoiseCount(int arg)
{
    m_noiseCount = arg;
}

void Captcha::setLineCount(int arg)
{
    m_lineCount = arg;
}

void Captcha::setEllipseCount(int arg)
{
    m_ellipseCount = arg;
}

void Captcha::setLineWidth(int arg)
```

```
{
   m_lineWidth = arg;
}

void Captcha::setEllipseMinRadius(int arg)
{
   m_ellipseMinRadius = arg;
}

void Captcha::setEllipseMaxRadius(int arg)
{
   m_ellipseMaxRadius = arg;
}

void Captcha::setNoisePointSize(int arg)
{
   m_noisePointSize = arg;
}

void Captcha::generateText(int noOfChars, bool includeNumbers, bool includeSymbols, bool allCapital)
{
   if (noOfChars <= 0)
   {
      qWarning() << "Unable to generate text : Invalid number of characters";
      return;
   }

   QString text;

   if (m_textGeneration == TextGeneration_Random)
   {
      QVector<unsigned char> chars;
      for (int i = 0; i < noOfChars * 2; i++)
      {
         chars.push_back(65 + ((qreal) rand() / RAND_MAX) * (90 - 65));
         if (!allCapital) chars.push_back(97 + ((qreal) rand() / RAND_MAX) * (122 - 97));
         if (includeNumbers) chars.push_back(48 + ((qreal) rand() / RAND_MAX) * (57 - 48));
         if (includeSymbols) chars.push_back(33 + ((qreal) rand() / RAND_MAX) * (47 - 33));
      }

      for (int i = 0; i < noOfChars; i++)
      {
         text = text + QChar(chars[rand() % chars.size()]);
      }
```

```
        m_captchaText = text;

    }
    else if (m_textGeneration == TextGeneration_Dictionary)
    {
        if (m_dictionary.size() <= 5)
        {
            qWarning() << "In text generation : Dictionary size is too small";
            return;
        }
        m_captchaText = m_dictionary[(rand() / (qreal) RAND_MAX) * (m_dictionary.size() -
1.0)];
    }
    else
    {
        qWarning() << "Unable to generate text : Invalid text generation mode";
    }
    updateCaptcha();
}
```

Dialog.cpp

```
#include "dialog.h"
#include "ui_dialog.h"
#include <QPainterPath>
#include <QPainter>
#include <QtMath>
#include <QMessageBox>
Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
}

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

QString check;
Captcha cp;
int count =0;
void Dialog::paintEvent(QPaintEvent *)
{
    if(count==0){
```

```
    QPainter painter(this);
    cp.randomize();
    cp.setDifficulty(1);
    cp.generateText();
    painter.drawImage(30, 30, cp.captchaImage());
    count++;
}

}

void Dialog::on_pushButton_clicked()
{
    check = cp.captchaText();
   qDebug() << check;
    QString check1 = ui->lineEdit->text();
    if(check==check1){
        this->hide();
    }
    else{
        QMessageBox::critical(this,"ERROR","Invalid Captcha! Please Try Again");
       count=0;
    }

}


void Dialog::on_lineEdit_textChanged(const QString &arg1)
{

}
```

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QVBoxLayout>
#include <QFrame>
#include <QLabel>
#include <QMessageBox>
#include <QSqlQuery>
#include <QCryptographicHash>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)

    , ui(new Ui::MainWindow)
```

```
{
    ui->setupUi(this);

}

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

void MainWindow::on_pushButton_clicked()
{
    Phoneno = ui->Phone->text();
    Login l1(ui->Phone->text(),ui->Password->text());
    QByteArray hashedPassword = QCryptographicHash::hash(l1.getPassword().toUtf8(),
QCryptographicHash::Md5);
    QString hashedPasswordStr = QString(hashedPassword.toHex());
    Database db;
    db.connOpen();
    QSqlQuery qry;
    qry.prepare("SELECT * FROM records WHERE Phone = '"+l1.getPhone()+"' AND Password
= '"+hashedPasswordStr+"'");
    if(qry.exec()){
        int count=0;
        while(qry.next()){
            count++;
        }
        if(count==1){
            QMessageBox::information(this,"Login Success", "LOGIN SUCCEFULL");
            this->hide();
            Ticket t1;
            t1.setModal(true);
            t1.exec();

        }
        else{
             QMessageBox::information(this,"Login Failed", "LOGIN Failed");
        }
    }
}


void MainWindow::on_pushButton_2_clicked()
{
    this->hide();
    SignUp s1;
```

```
    s1.setModal(true);
    s1.exec();
}
```

Payment.cpp

```
#include "payment.h"
#include "ui_payment.h"
#include "bill.h"
Payment::Payment(QWidget *parent,trainInfo t) :
    QDialog(parent),
    ui(new Ui::Payment),
    t1(t)

{
    ui->setupUi(this);
    ui->frame_2->setVisible(false);
    ui->radioButton->setChecked(true);
}

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

void Payment::on_radioButton_clicked()
{
    ui->frame->setVisible(true);
    ui->frame_2->setVisible(false);
}


void Payment::on_radioButton_2_clicked()
{
    ui->frame->setVisible(false);
    ui->frame_2->setVisible(true);
}


void Payment::on_radioButton_3_clicked()
{
    ui->frame->setVisible(false);
    ui->frame_2->setVisible(true);
}
```

```
void Payment::on_pushButton_2_clicked()
{
this->hide();
    Bill b1(nullptr,t1);
    b1.setModal(true);
    b1.exec();
}


void Payment::on_pushButton_clicked()
{

}
```

Seats.cpp

```
#include "seats.h"
#include "mainwindow.h"
#include "ui_seats.h"
#include <QSqlQuery>
#include "QMessageBox"
int count1=0;
Seats::Seats(QWidget *parent,trainInfo t1) :
    QDialog(parent),
    ui(new Ui::Seats),
    t(t1)
{
    int total = (t.getPassengers().toInt() * t.getPrice().toInt());
    int check;
    ui->setupUi(this);
    Database db;
    db.connOpen();
    QSqlQuery qry;
    qry.prepare("SELECT * FROM bookings WHERE Source = '"+t1.getSource()+"' AND
Destination = '"+t1.getDestination()+"' AND Time = '"+t1.getTime()+"' AND Date
='"+t1.getDate()+"' ");
    if(qry.exec()){
        qDebug() << "Sucess";
    }
    while(qry.next()){
        check = qry.value(5).toInt();

        if (check == 1) {
            ui->c1->setDisabled(true);
        }
        else if (check == 2) {
```

```
    ui->c2->setDisabled(true);
}
else if (check == 3) {
    ui->c3->setDisabled(true);
}
else if (check == 4) {
    ui->c4->setDisabled(true);
}
else if (check == 5) {
    ui->c5->setDisabled(true);
}
else if (check == 6) {
    ui->c6->setDisabled(true);
}
else if (check == 7) {
    ui->c7->setDisabled(true);
}
else if (check == 8) {
    ui->c8->setDisabled(true);
}
else if (check == 9) {
    ui->c9->setDisabled(true);
}
else if (check == 10) {
    ui->c10->setDisabled(true);
}
else if (check == 11) {
    ui->c11->setDisabled(true);
}
else if (check == 12) {
    ui->c12->setDisabled(true);
}
else if (check == 13) {
    ui->c13->setDisabled(true);
}
else if (check == 14) {
    ui->c14->setDisabled(true);
}
else if (check == 15) {
    ui->c15->setDisabled(true);
}
else if (check == 16) {
    ui->c16->setDisabled(true);
}
else if (check == 17) {
    ui->c17->setDisabled(true);
```

```
}
else if (check == 18) {
   ui->c18->setDisabled(true);
}
else if (check == 19) {
   ui->c19->setDisabled(true);
}
else if (check == 20) {
   ui->c20->setDisabled(true);
}
else if (check == 21) {
   ui->c21->setDisabled(true);
}
else if (check == 22) {
   ui->c22->setDisabled(true);
}
else if (check == 23) {
   ui->c23->setDisabled(true);
}
else if (check == 24) {
   ui->c24->setDisabled(true);
}
else if (check == 25) {
   ui->c25->setDisabled(true);
}
else if (check == 26) {
   ui->c26->setDisabled(true);
}
else if (check == 27) {
   ui->c27->setDisabled(true);
}
else if (check == 28) {
   ui->c28->setDisabled(true);
}
else if (check == 29) {
   ui->c29->setDisabled(true);
}
else if (check == 30) {
   ui->c30->setDisabled(true);
}
else if (check == 31) {
   ui->c31->setDisabled(true);
}
else if (check == 32) {
   ui->c32->setDisabled(true);
}
```

```
        else if (check == 33) {
            ui->c33->setDisabled(true);
        }
        else if (check == 34) {
            ui->c34->setDisabled(true);
        }
        else if (check == 35) {
            ui->c35->setDisabled(true);
        }
        else if (check == 36) {
            ui->c36->setDisabled(true);
        }
        else if (check == 37) {
            ui->c37->setDisabled(true);
        }
        else if (check == 38) {
            ui->c38->setDisabled(true);
        }
        else if (check == 39) {
            ui->c39->setDisabled(true);
        }
        else if (check == 40) {
            ui->c40->setDisabled(true);
        }

    }
    db.connClose();
    setBill(total);
    ui->label->setText(QString("Your total bill is: %1 Click Below to Pay").arg(getBill()));

}
QString Phoneno;
Seats::~Seats()
{
    delete ui;
}
void insertdata(int seat,trainInfo t1){
    qDebug() << Phoneno;
    QString name;
    Database db;
    db.connOpen();

    QSqlQuery query;
    query.prepare("SELECT * FROM records WHERE Phone = '"+Phoneno+"'");
    query.exec();
    int count=0;
```

```
    while(query.next()){
        count++;
    }
    if(count==1){
        query.first();
        name =query.value(0).toString();
        qDebug() << name;
    }
    db.connClose();
    db.connOpen();
    QSqlQuery qry;
    qry.prepare("INSERT INTO bookings(Name,Phone,Source,Destination,Passengers,Seatno,Time,Type,Price,Date)
Values(:Name,:Phone,:Source,:Destination,:Passengers,:Seatno,:Time,:Type,:Price,:Date)");
    qry.bindValue(":Name",name);
    qry.bindValue(":Phone",Phoneno);
    qry.bindValue(":Source",t1.getSource());
    qry.bindValue(":Destination",t1.getDestination());
    qry.bindValue(":Passengers",t1.getPassengers());
    qry.bindValue(":Seatno",seat);
    qry.bindValue(":Time",t1.getTime());
    qry.bindValue(":Type",t1.getType());
    qry.bindValue(":Price",t1.getPrice());
    qry.bindValue(":Date",t1.getDate());
    if(qry.exec()){
        qDebug() << "Sucessful";
    }
    db.connClose();
}
void Seats::on_checkBox_stateChanged(int arg1)
{

}


void Seats::on_pushButton_clicked()
{
    if(count1 ==t.getPassengers().toInt()){
        int seat;
        if(ui->c1->isChecked()){
            seat =1;
            insertdata(seat,t);
        }
        if(ui->c2->isChecked()){
            seat =2;
            insertdata(seat,t);
```

```
}
if(ui->c3->isChecked()){
    seat =3;
    insertdata(seat,t);
}
if(ui->c4->isChecked()){
    seat =4;
    insertdata(seat,t);
}
if(ui->c5->isChecked()){
    seat =5;
    insertdata(seat,t);
}
if(ui->c6->isChecked()){
    seat =6;
    insertdata(seat,t);
}
if(ui->c7->isChecked()){
    seat =7;
    insertdata(seat,t);
}
if(ui->c8->isChecked()){
    seat =8;
    insertdata(seat,t);
}
if(ui->c9->isChecked()){
    seat =9;
    insertdata(seat,t);
}
if(ui->c10->isChecked()){
    seat =10;
    insertdata(seat,t);
}
if(ui->c11->isChecked()){
    seat =11;
    insertdata(seat,t);
}
if(ui->c12->isChecked()){
    seat =12;
    insertdata(seat,t);
}
if(ui->c13->isChecked()){
    seat =13;
    insertdata(seat,t);
}
if(ui->c14->isChecked()){
```

```
    seat =14;
    insertdata(seat,t);
}
if(ui->c15->isChecked()){
    seat =15;
    insertdata(seat,t);
}
if(ui->c16->isChecked()){
    seat =16;
    insertdata(seat,t);
}
if (ui->c17->isChecked()){
    seat =17;
    insertdata(seat,t);
}
if(ui->c18->isChecked()){
    seat =18;
    insertdata(seat,t);
}
if(ui->c19->isChecked()){
    seat =19;
    insertdata(seat,t);
}
if(ui->c20->isChecked()){
    seat =20;
    insertdata(seat,t);
}
if(ui->c21->isChecked()){
    seat =21;
    insertdata(seat,t);
}
if(ui->c22->isChecked()){
    seat =22;
    insertdata(seat,t);
}
if(ui->c23->isChecked()){
    seat =23;
    insertdata(seat,t);
}
if(ui->c24->isChecked()){
    seat =24;
    insertdata(seat,t);
}
if(ui->c25->isChecked()){
    seat =25;
    insertdata(seat,t);
```

```
}
if(ui->c26->isChecked()){
    seat =26;
    insertdata(seat,t);
}
if(ui->c27->isChecked()){
    seat =27;
    insertdata(seat,t);
}
if(ui->c28->isChecked()){
    seat =28;
    insertdata(seat,t);
}
if(ui->c29->isChecked()){
    seat =29;
    insertdata(seat,t);
}
if(ui->c30->isChecked()){
    seat =30;
    insertdata(seat,t);
}
if(ui->c31->isChecked()){
    seat =31;
    insertdata(seat,t);
}
if(ui->c32->isChecked()){
    seat =32;
    insertdata(seat,t);
}
if(ui->c33->isChecked()){
    seat =33;
    insertdata(seat,t);
}
if(ui->c34->isChecked()){
    seat =34;
    insertdata(seat,t);
}
if(ui->c35->isChecked()){
    seat =35;
    insertdata(seat,t);
}
if(ui->c36->isChecked()){
    seat =36;
    insertdata(seat,t);
}
if(ui->c37->isChecked()){
```

```
            seat =37;
            insertdata(seat,t);
        }
        if(ui->c38->isChecked()){
            seat =38;
            insertdata(seat,t);
        }
        if(ui->c39->isChecked()){
            seat =39;
            insertdata(seat,t);
        }
        if(ui->c40->isChecked()){
            seat =40;
            insertdata(seat,t);
        }
        this->hide();
        Payment p(nullptr,t);
        p.setModal(true);
        p.exec();
    }
    else{
        QMessageBox::critical(this,"Error","Invalid number of seats selected");
    }
}


void Seats::on_checkBox_2_stateChanged(int arg1)
{

}


void Seats::on_c1_stateChanged(int arg1)
{
    if(ui->c1->isChecked()){
        count1++;}
    if(ui->c1->checkState()== Qt::Unchecked){
        count1--;
    }

}
void Seats::on_c2_stateChanged(int arg1)
{
    if(ui->c2->isChecked()){
        count1++;}
    if(ui->c2->checkState()== Qt::Unchecked){
```

```
        count1--;
    }
}
void Seats::on_c3_stateChanged(int arg1)
{
    if(ui->c3->isChecked()){
        count1++;}
    if(ui->c3->checkState()== Qt::Unchecked){
        count1--;
    }
}
void Seats::on_c4_stateChanged(int arg1)
{
    if(ui->c4->isChecked()){
        count1++;}
    if(ui->c4->checkState()== Qt::Unchecked){
        count1--;
    }
}
void Seats::on_c5_stateChanged(int arg1)
{
    if(ui->c5->isChecked()){
        count1++;}
    if(ui->c5->checkState()== Qt::Unchecked){
        count1--;
    }
}
void Seats::on_c6_stateChanged(int arg1)
{
    if(ui->c6->isChecked()){
        count1++;}
    if(ui->c6->checkState()== Qt::Unchecked){
        count1--;
    }
}
void Seats::on_c7_stateChanged(int arg1)
{
    if(ui->c7->isChecked()){
        count1++;}
    if(ui->c7->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c8_stateChanged(int arg1)
{
    if(ui->c8->isChecked()){
        count1++;}
```

```
    if(ui->c8->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c9_stateChanged(int arg1)
{
    if(ui->c9->isChecked()){
        count1++;}
    if(ui->c9->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c10_stateChanged(int arg1)
{
    if(ui->c10->isChecked()){
        count1++;}
    if(ui->c10->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c11_stateChanged(int arg1)
{
    if(ui->c11->isChecked()){
        count1++;}
    if(ui->c11->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c12_stateChanged(int arg1)
{
    if(ui->c12->isChecked()){
        count1++;}
    if(ui->c12->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c13_stateChanged(int arg1)
{
    if(ui->c13->isChecked()){
        count1++;}
    if(ui->c13->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c14_stateChanged(int arg1)
{
    if(ui->c14->isChecked()){
        count1++;}
    if(ui->c14->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c15_stateChanged(int arg1)
```

```
{
   if(ui->c15->isChecked()){
      count1++;}
   if(ui->c15->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c16_stateChanged(int arg1)
{
   if(ui->c16->isChecked()){
      count1++;}
   if(ui->c16->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c17_stateChanged(int arg1)
{
   if(ui->c17->isChecked()){
      count1++;}
   if(ui->c17->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c18_stateChanged(int arg1)
{
   if(ui->c18->isChecked()){
      count1++;}
   if(ui->c18->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c19_stateChanged(int arg1)
{
   if(ui->c19->isChecked()){
      count1++;}
   if(ui->c19->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c20_stateChanged(int arg1)
{
   if(ui->c20->isChecked()){
      count1++;}
   if(ui->c20->checkState()== Qt::Unchecked){
      count1--;
   }
}void Seats::on_c21_stateChanged(int arg1)
{
   if(ui->c21->isChecked()){
      count1++;}
   if(ui->c21->checkState()== Qt::Unchecked){
```

```
        count1--;
    }
}void Seats::on_c22_stateChanged(int arg1)
{
    if(ui->c22->isChecked()){
        count1++;}
    if(ui->c22->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c23_stateChanged(int arg1)
{
    if(ui->c23->isChecked()){
        count1++;}
    if(ui->c23->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c24_stateChanged(int arg1)
{
    if(ui->c24->isChecked()){
        count1++;}
    if(ui->c24->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c25_stateChanged(int arg1)
{
    if(ui->c25->isChecked()){
        count1++;}
    if(ui->c25->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c26_stateChanged(int arg1)
{
    if(ui->c26->isChecked()){
        count1++;}
    if(ui->c26->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c27_stateChanged(int arg1)
{
    if(ui->c27->isChecked()){
        count1++;}
    if(ui->c27->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c28_stateChanged(int arg1)
{
```

```
    if(ui->c28->isChecked()){
        count1++;}
    if(ui->c28->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c29_stateChanged(int arg1)
{
    if(ui->c29->isChecked()){
        count1++;}
    if(ui->c29->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c30_stateChanged(int arg1)
{
    if(ui->c30->isChecked()){
        count1++;}
    if(ui->c30->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c31_stateChanged(int arg1)
{
    if(ui->c31->isChecked()){
        count1++;}
    if(ui->c31->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c32_stateChanged(int arg1)
{
    if(ui->c32->isChecked()){
        count1++;}
    if(ui->c32->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c33_stateChanged(int arg1)
{
    if(ui->c33->isChecked()){
        count1++;}
    if(ui->c33->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c34_stateChanged(int arg1)
{
    if(ui->c34->isChecked()){
        count1++;}
    if(ui->c34->checkState()== Qt::Unchecked){
        count1--;
```

```
    }
}void Seats::on_c35_stateChanged(int arg1)
{
    if(ui->c35->isChecked()){
        count1++;}
    if(ui->c35->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c36_stateChanged(int arg1)
{
    if(ui->c36->isChecked()){
        count1++;}
    if(ui->c36->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c37_stateChanged(int arg1)
{
    if(ui->c37->isChecked()){
        count1++;}
    if(ui->c37->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c38_stateChanged(int arg1)
{
    if(ui->c38->isChecked()){
        count1++;}
    if(ui->c38->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c39_stateChanged(int arg1)
{
    if(ui->c39->isChecked()){
        count1++;}
    if(ui->c39->checkState()== Qt::Unchecked){
        count1--;
    }
}void Seats::on_c40_stateChanged(int arg1)
{
    if(ui->c40->isChecked()){
        count1++;}
    if(ui->c40->checkState()== Qt::Unchecked){
        count1--;
    }
}
```

Signup.cpp

```
#include "signup.h"
#include "ui_signup.h"
#include <QSqlQuery>
#include <QMessageBox>
#include <QDebug>
#include <QCryptographicHash> // Include the cryptographic hash header

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

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

void SignUp::on_pushButton_clicked()
{
    Dialog d1;
    d1.setModal(true);
    d1.exec();
    int count = 0;
    bool check = false;
    Database db;
    db.connOpen();
    QSqlQuery qry;
    Sign_UP s1(ui->Name->text(), ui->Phone->text(), ui->Email->text(), ui->Password->text(),
ui->RePassword->text());
    // Hash the password
    QByteArray hashedPassword = QCryptographicHash::hash(s1.getPassword().toUtf8(),
QCryptographicHash::Md5);
    QString hashedPasswordStr = QString(hashedPassword.toHex());

    if (s1.getPhone().length() == 11) {
        count++;
        qDebug() << "count";
    } else {
        QMessageBox::critical(this, "Invalid Phone", "Invalid Phone");
    }

    QString checkEmail = s1.getEmail();
```

```
    for (int i = 0; i < s1.getEmail().length(); i++) {
        if (checkEmail[i] == '@') {
            qDebug() << "HEHE";
            for (int j = 0; j < s1.getEmail().length(); j++) {
                if (checkEmail[j] == '.') {
                    count++;
                    qDebug() << "HEHE2";
                    check = true;
                    break;
                }
            }
        }
    }

    if (!check) {
        QMessageBox::critical(this, "Invalid Email", "Invalid Email Format");
    }

    if (s1.getCheckpass() == s1.getPassword()) {
        count++;
    } else {
        QMessageBox::critical(this, "Invalid Password", "Passwords dont match");
    }

    if (count >= 3) {

        qry.prepare("INSERT INTO records(Name,Phone,Email,Password)
Values(:Name,:Phone,:Email,:Password)");
        qry.bindValue(":Name", s1.getName());
        qry.bindValue(":Phone", s1.getPhone());
        qry.bindValue(":Email", s1.getEmail());
        qry.bindValue(":Password", hashedPasswordStr); // Store the hashed password
        if (qry.exec()) {
            QMessageBox::information(this, "Success", "Registration successful!");
        } else {
            QMessageBox::critical(this, "Error", "Registration failed. User Already exists");
        }
        db.connClose();
    }
}
```

Ticket.cpp

```
#include "ticket.h"
#include "ui_ticket.h"
extern QString Phoneno;
Ticket::Ticket(QWidget *parent) :
```

```
    QDialog(parent),
    ui(new Ui::Ticket)
{
    ui->setupUi(this);
}

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

void Ticket::on_pushButton_clicked()
{
    this->hide();
    Booking b1;
    b1.setModal(true);
    b1.exec();
}
```

Train.cpp

```
#include "train.h"
#include "ui_train.h"
#include <QSqlQuery>
#include "seats.h"
Train::Train(QWidget *parent,trainInfo t2) :
    QDialog(parent),
    ui(new Ui::Train),

    t1(t2)
{
Database db;
db.connOpen();
    ui->setupUi(this);
    ui->label1->setText(t1.getSource());
    ui->label1_2->setText(t1.getSource());
    ui->label1_3->setText(t1.getSource());
    ui->label1_4->setText(t1.getSource());
    ui->label1_5->setText(t1.getSource());
    ui->label2->setText(t1.getDestination());
    ui->label2_2->setText(t1.getDestination());
    ui->label2_3->setText(t1.getDestination());
    ui->label2_4->setText(t1.getDestination());
    ui->label2_5->setText(t1.getDestination());
    ui->label5->setText(t1.getDate());
    ui->label5_2->setText(t1.getDate());
```

```
    ui->label5_3->setText(t1.getDate());
    ui->label5_4->setText(t1.getDate());
    ui->label5_5->setText(t1.getDate());
    QSqlQuery qry;
    qry.prepare("SELECT * FROM Trains WHERE Source = '"+t1.getSource()+"' AND
Destination = '"+t1.getDestination()+"'");
    if (qry.exec()) {
        int count = 0;
        while (count < 5 && qry.next()) {
            t1.setTime(qry.value(2).toString());
            t1.setType(qry.value(4).toString());
            t1.setPrice(qry.value(5).toString());
            if (count == 0) {
                ui->label6->setText(t1.getTime());
                ui->label3->setText(t1.getType());
                ui->label4->setText(t1.getPrice());
            } else if (count == 1) {
                ui->label6_2->setText(t1.getTime());
                ui->label3_2->setText(t1.getType());
                ui->label4_2->setText(t1.getPrice());
            }else if (count == 2) {
                ui->label6_3->setText(t1.getTime());
                ui->label3_3->setText(t1.getType());
                ui->label4_3->setText(t1.getPrice());
            }else if (count == 3) {
                ui->label6_4->setText(t1.getTime());
                ui->label3_4->setText(t1.getType());
                ui->label4_4->setText(t1.getPrice());
            }else if (count == 4) {
                ui->label6_5->setText(t1.getTime());
                ui->label3_5->setText(t1.getType());
                ui->label4_5->setText(t1.getPrice());
            }
            count++;
        }
    }

}

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

void Train::on_pushButton_clicked()
{
```

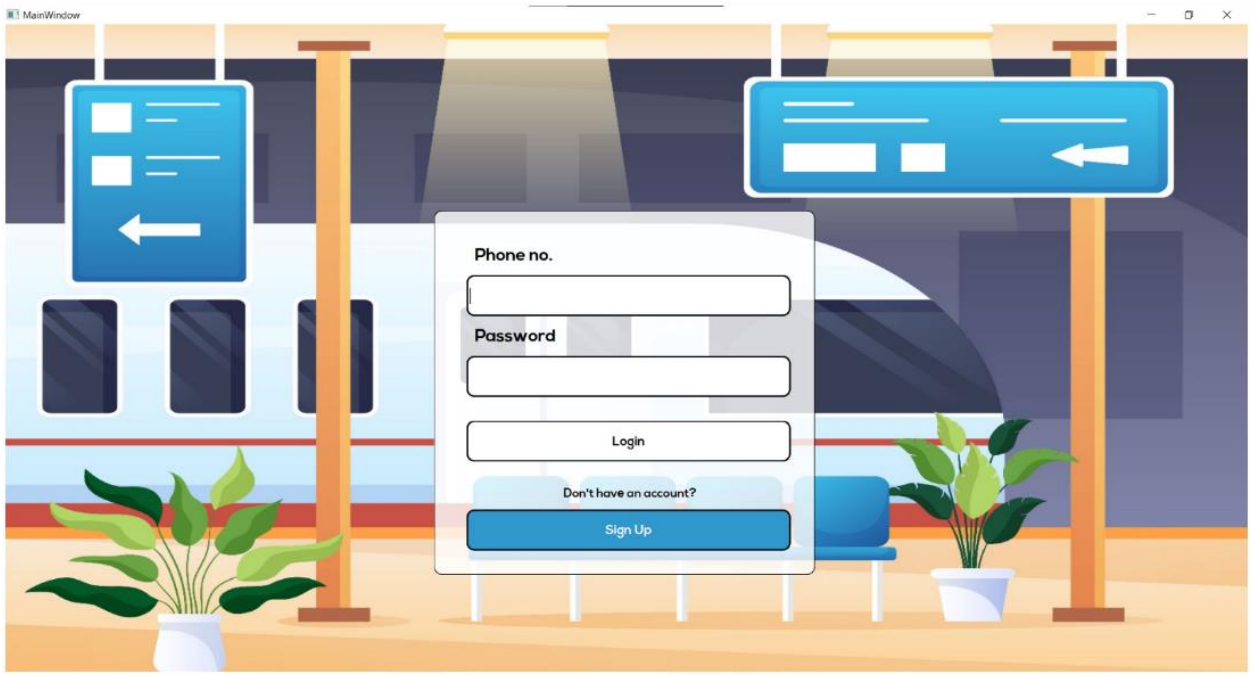
```
    t1.setTime(ui->label6->text());
    t1.setType(ui->label3->text());
    t1.setPrice(ui->label4->text());
    this->hide();
    Seats s1(nullptr,t1);
    s1.setModal(true);
    s1.exec();
}


void Train::on_pushButton_2_clicked()
{
    t1.setTime(ui->label6_2->text());
    t1.setType(ui->label3_2->text());
    t1.setPrice(ui->label4_2->text());
    this->hide();
    Seats s1(nullptr,t1);
    s1.setModal(true);
    s1.exec();
}


void Train::on_pushButton_3_clicked()
{

        t1.setTime(ui->label6_3->text());
        t1.setType(ui->label3_3->text());
        t1.setPrice(ui->label4_3->text());
        this->hide();
        Seats s1(nullptr,t1);
        s1.setModal(true);
        s1.exec();

}


void Train::on_pushButton_4_clicked()
{
    t1.setTime(ui->label6_4->text());
    t1.setType(ui->label3_4->text());
    t1.setPrice(ui->label4_4->text());
    this->hide();
    Seats s1(nullptr,t1);
    s1.setModal(true);
    s1.exec();
}
```

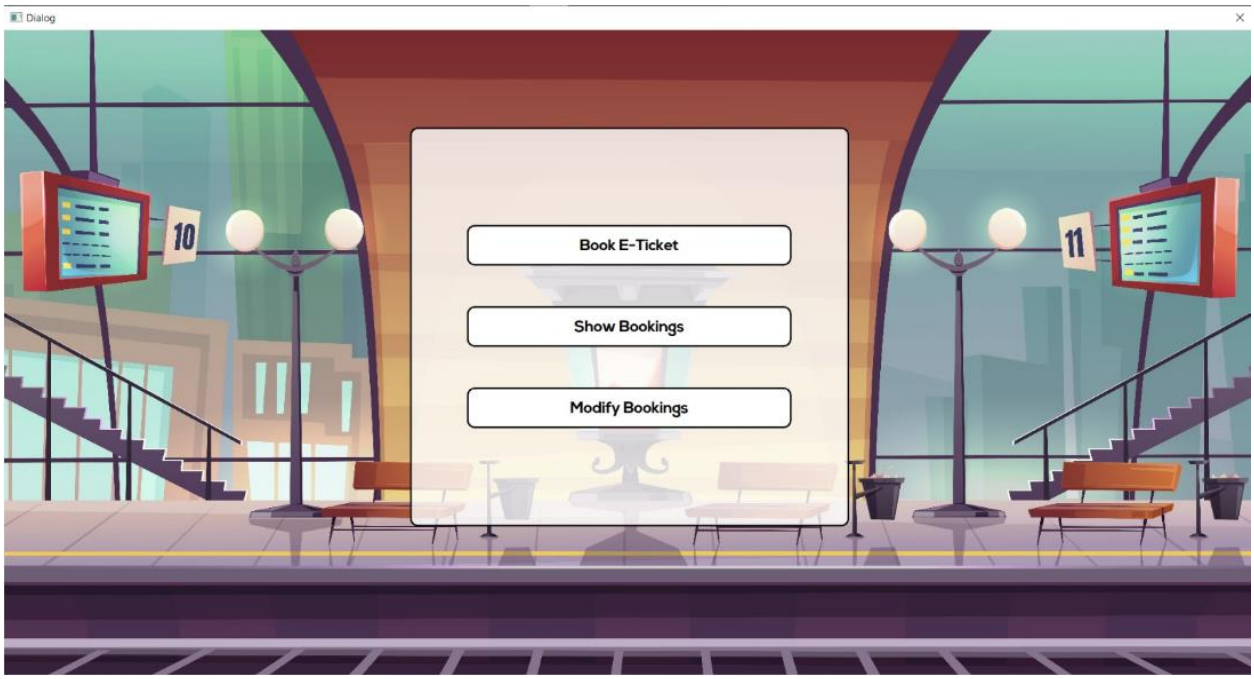
```
void Train::on_pushButton_5_clicked()
{
   t1.setTime(ui->label6_5->text());
   t1.setType(ui->label3_5->text());
   t1.setPrice(ui->label4_5->text());
   this->hide();
   Seats s1(nullptr,t1);
   s1.setModal(true);
   s1.exec();
}
```

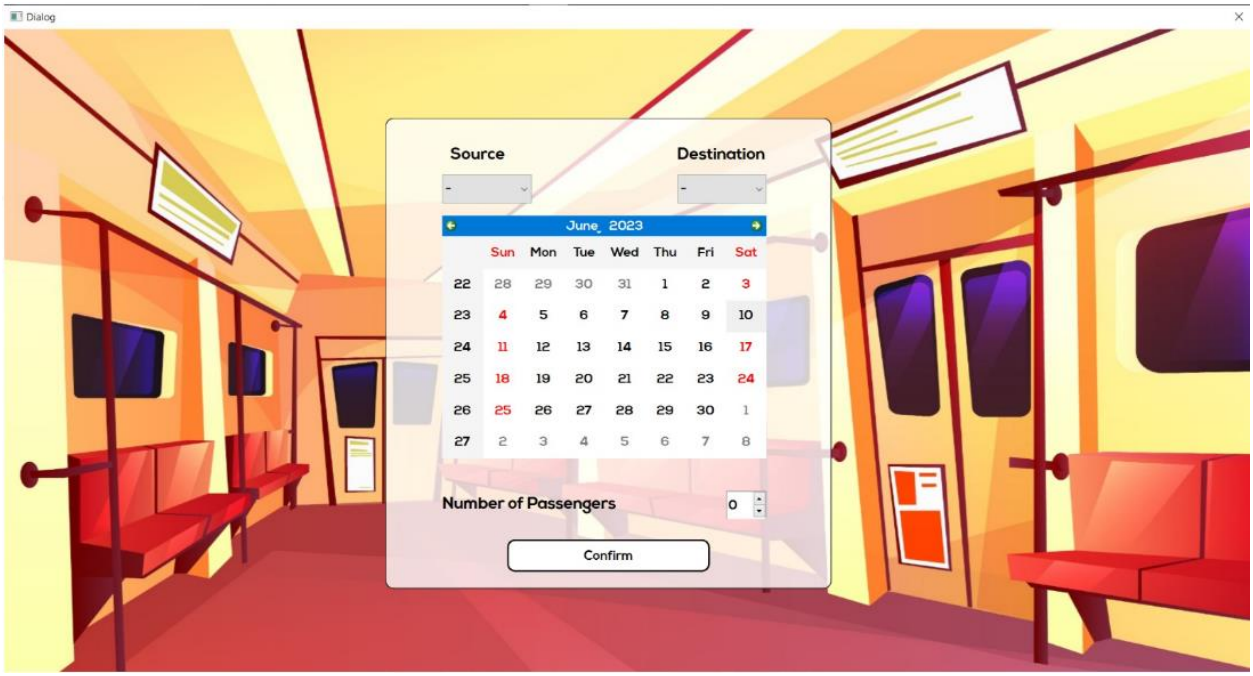
Main.cpp

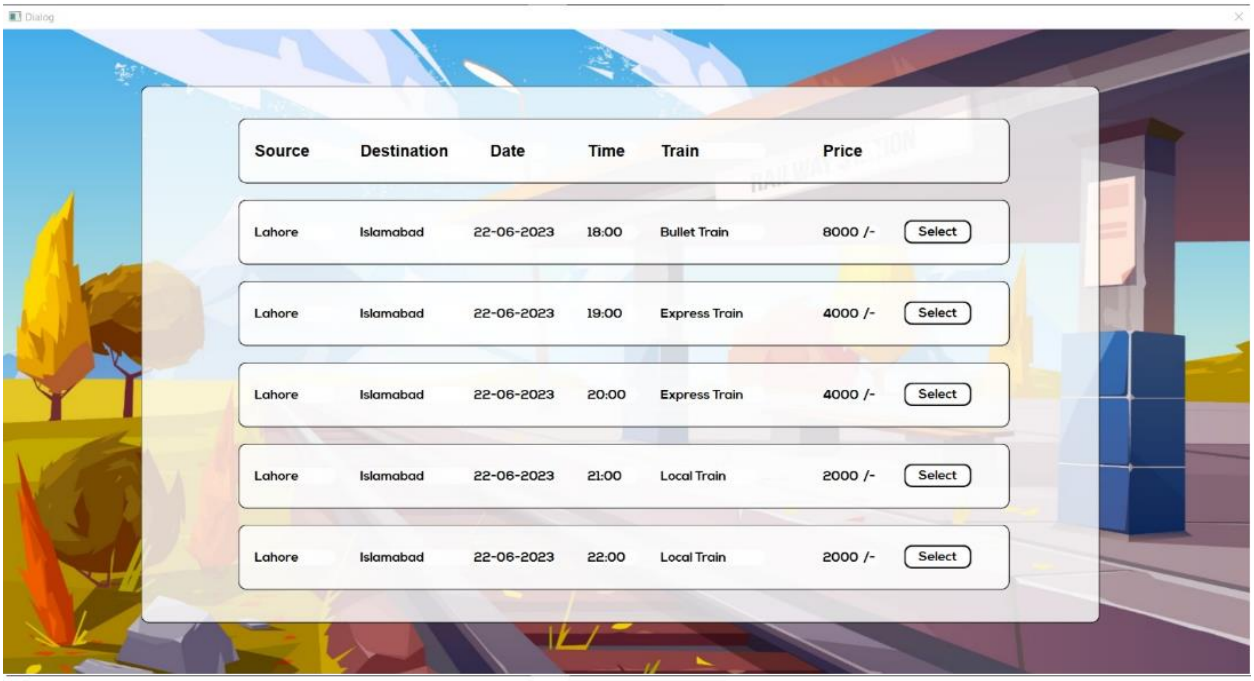
```
#include "mainwindow.h"
#include "qapplication.h"

int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   MainWindow w;
   w.show();

   return a.exec();
}
```

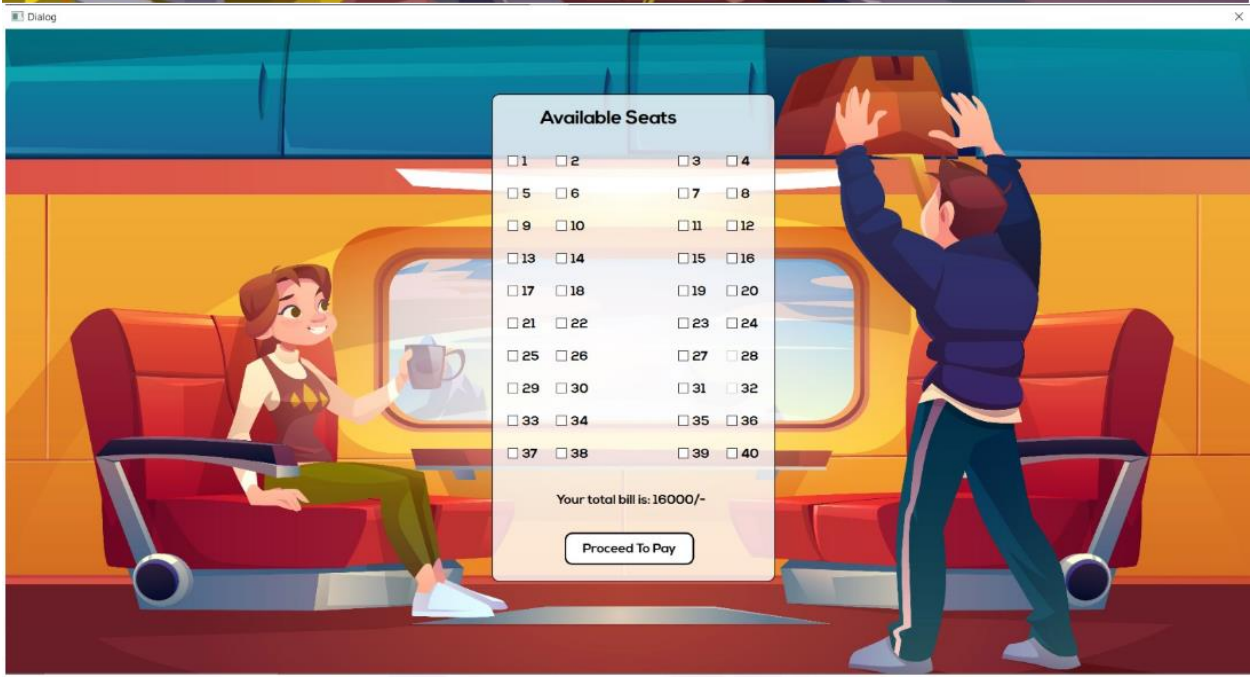
Screenshots

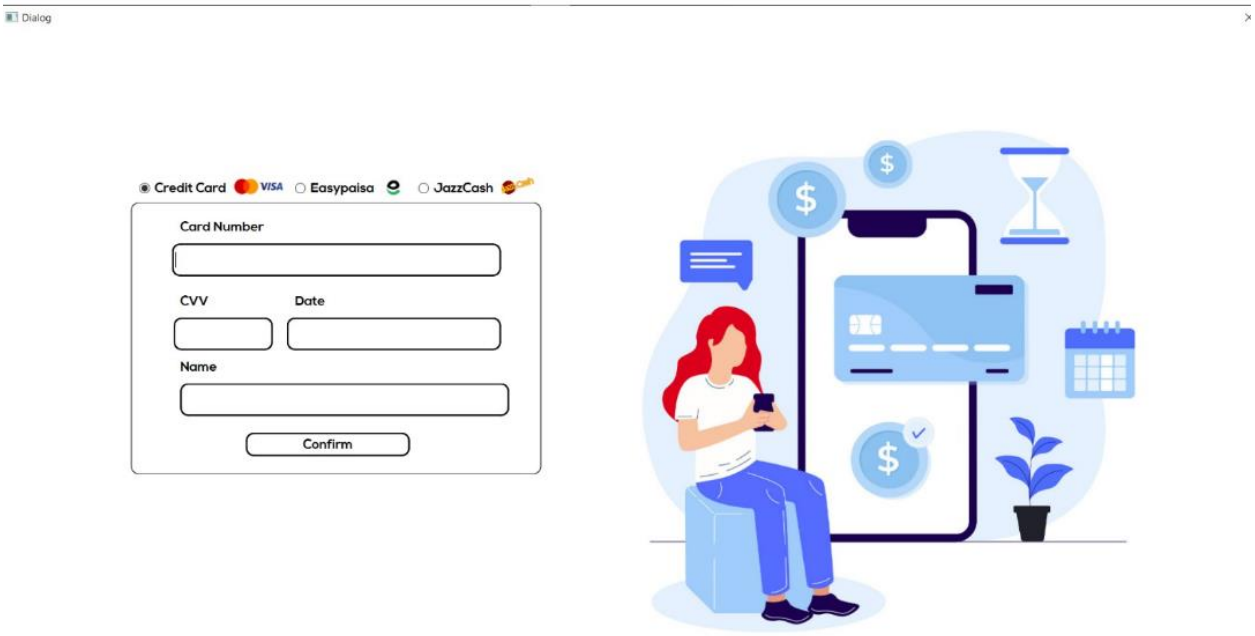


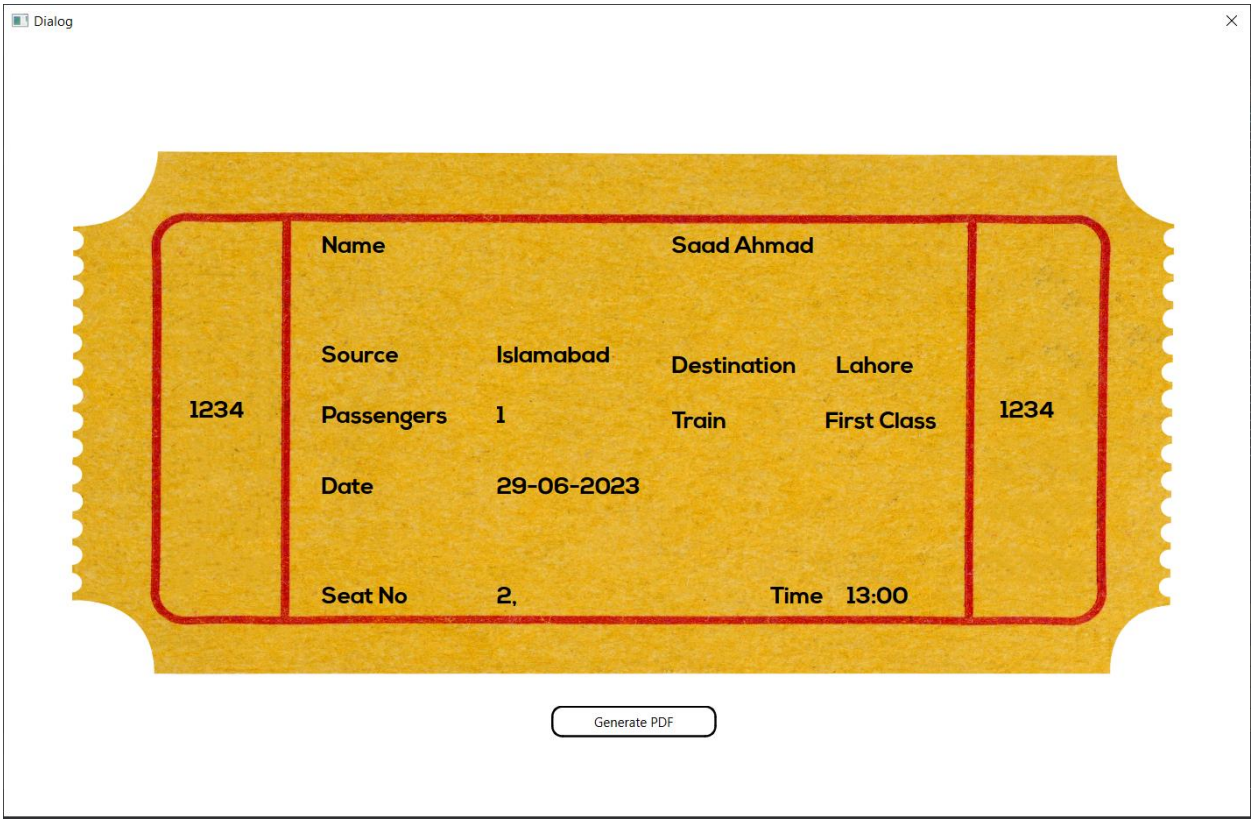

Dialog
10
11
Book E-Ticket
Show Bookings
Modify Bookings


Dialog
Source
Destination
June, 2023
Sun Mon Tue Wed Thu Fri Sat
22 28 29 30 31 1 2 3
23 4 5 6 7 8 9 10
24 11 12 13 14 15 16 17
25 18 19 20 21 22 23 24
26 25 26 27 28 29 30 1
27 2 3 4 5 6 7 8
Number of Passengers
0
Confirm

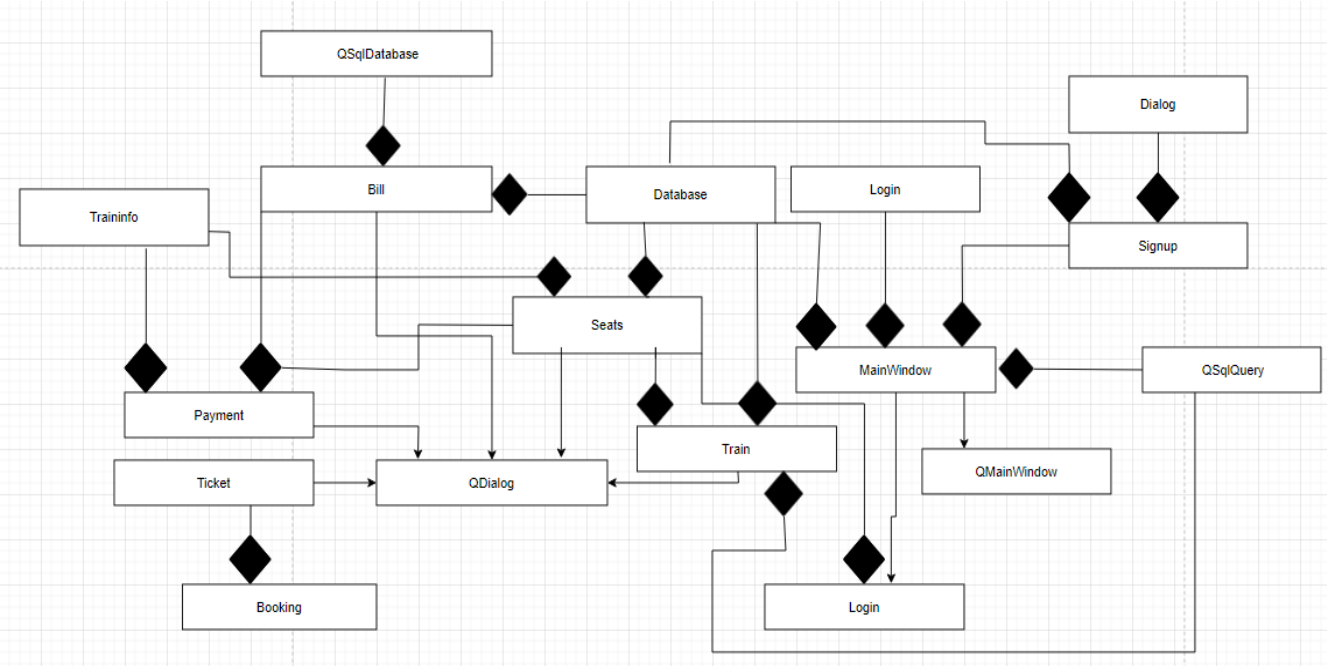

Dialog
Source
Destination
Date
Time
Train
Price
Lahore Islamabad 22-06-2023 18:00 Bullet Train 8000 /- Select
Lahore Islamabad 22-06-2023 19:00 Express Train 4000 /- Select
Lahore Islamabad 22-06-2023 20:00 Express Train 4000 /- Select
Lahore Islamabad 22-06-2023 21:00 Local Train 2000 /- Select
Lahore Islamabad 22-06-2023 22:00 Local Train 2000 /- Select


Dialog
Available Seats
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
21 22 23 24
25 26 27 28
29 30 31 32
33 34 35 36
37 38 39 40
Your total bill is: 16000/-
Proceed To Pay


Dialog
Credit Card
Easypaisa
JazzCash
Card Number
CVV
Date
Name
Confirm


Dialog
Name
Saad Ahmad
Source
Islamabad
Destination
Lahore
1234
Passengers
1
Train
First Class
1234
Date
29-06-2023
Seat No
2,
Time
13:00
Generate PDF

UML Diagram


QSqlDatabase
Dialog
Bill
Database
Login
Traininfo
Signup
Seats
MainWindow
QSqlQuery
Payment
Train
QMainWindow
Ticket
QDialog
Booking
Login