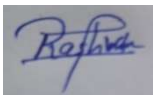
IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: U.P.A.R. Vidushani	
Student Reference Number:10707397	

Module Code: SOFT 336SL	Module Name: Cross Platform Application Development
Coursework Title: Application- Edu-Hive	
Deadline Date: 18.01.2022	Member of staff responsible for coursework: Marius Varga
Programme: BSc(Hons)Software Engineering	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed : 

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

Overall mark _____% **Assessors Initials _____** **Date_____**

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

Content

Chapter 1 Overview

Education is one of the major factors to be achieved by an individual as this factor decides the whole future of the individual and even their loved ones. In order to achieve this factor, one needs to study hard and work hard for his or her education to gain satisfiable grades. To get satisfiable grades, one need to prepare well for examinations. Students prepare for examinations early with positive attitudes and goals mostly by redoing old papers which are assigned to their grades. But in this pandemic situation education is mostly technology based.

Therefore, most of the previously used techniques are changed. For example, lectures are held via applications like zoom, and teams, education resources are shared via whatsapp groups, telegram groups, the ultimate goal of each of these ways is to educate the child to do the examinations and get a considerable result. These exams too are conducted using technologies like learning management systems of the relevant institutions.

Chapter 2 Introduction on Edu-Hive

When all of the important parts of educating a child is covered with technological aspects, the preparation for exams by doing past papers are still not done based on technology. Past papers are one of the best ways to know the dept of the examination. By doing past papers one can:

- Learn to work with the allocated time period, know the speed.
- Get an idea about the number of questions and question types.
- Helps to practice the exam techniques
- Helps in identifying key subject areas which need attention

Even in these situations, manual methods are used for doing past papers and main examinations are done using technology.

This is where Edu-Hive can be used. Edu-Hive is a simple application which is abled to be used in windows and android specifically. This application consists of 4 windows including the main dashboard. This can be used for text editing, time keeping as well as simple calculations.
Assume, a simple writing paper needs to be done. Then you can use the timer to keep up with time, and use text editor for texts. All three windows can be used if you need to do calculations, text editing and keep up with time at the same time.

Chapter 3 GUI Design

- **Main dashboard**

Below image shows the main dashboard of Edu-Hive. The main use of this dashboards that this helps the users to navigate to the relevant window/s they desire using the buttons at the bottom of the window. They can navigate to any of the windows, Notepad, Calculator, or timer. Other than that, this window shows the current time, date and the day to the user of the software.

Main Dashboard -> Notepad

Main Dashboard -> Timer

Main Dashboard -> Calculator

Windows	Android
	

- **Notepad**

The below image shows the window which is opened once the user clicks the Add notes button from the dashboard. The menu bar of the note pad window consists of functions such as,

File tab:

New- new text file	Save as- save the current text file	Exit- exit window
Open- open existing text file	Print- print the current text file	

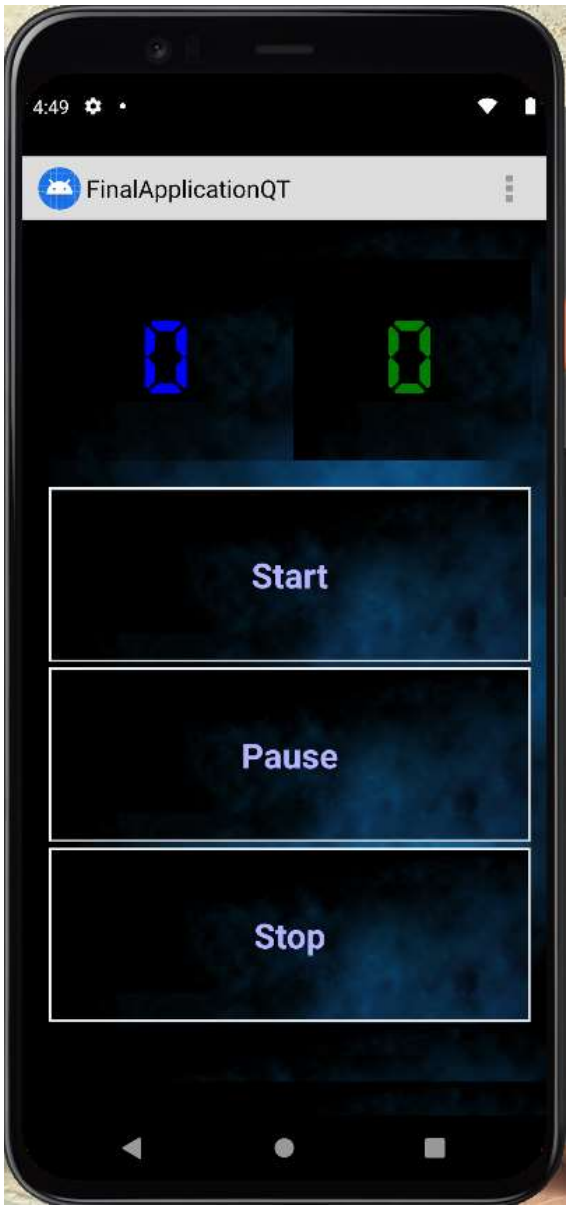
Edit tab:

Cut- Cut text	Paste- Paste text	Redo- redo activity
Copy- Copy text	Undo- undo activity	

Windows	Android
	

- **Timer**

The below image shows the window opened when the timer button is clicked in the dashboard. There are two displays to show the minutes and seconds respectively. One can use the start button to start the process running, pause button to pause the timer and re-click the pause button to keep going the process and stop button to stop the timer.

Windows	Android
	

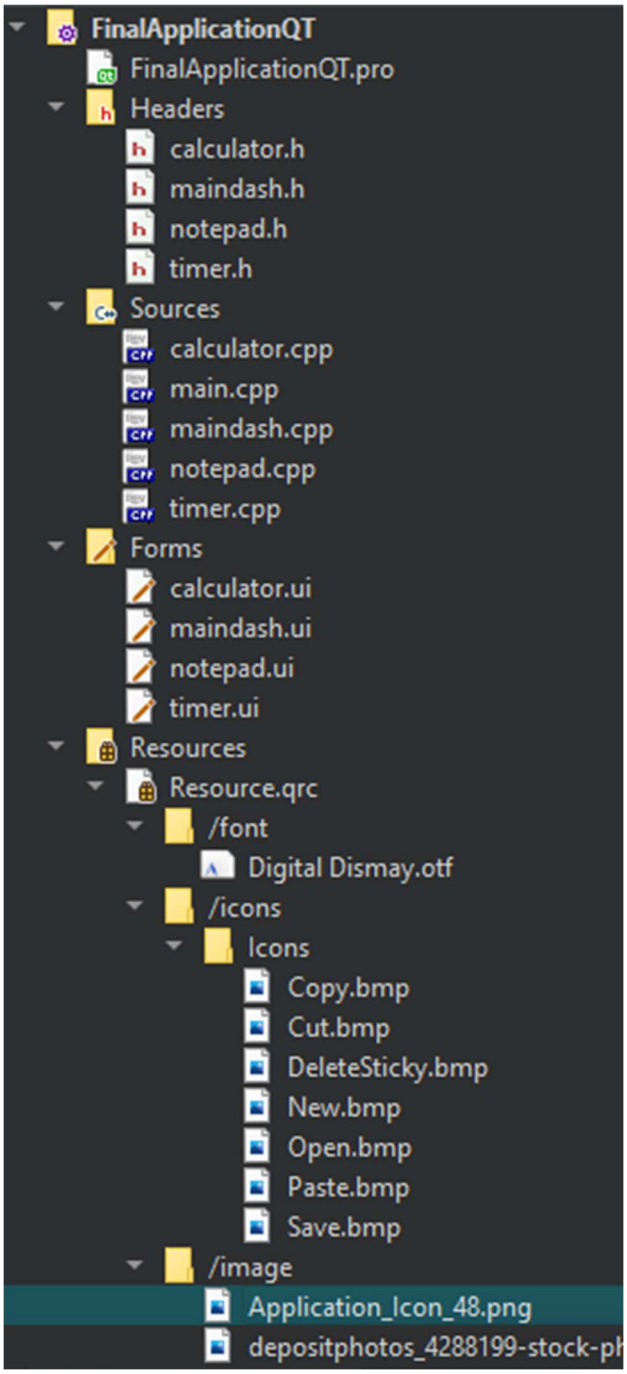
- **Calculator**

The below image shows the window opened when the calculator button is clicked in the dashboard. This window is to be used when doing calculations. Consists of number buttons as well as operation buttons. The number 0 is depicted on the display by default until user clicks a button.

Windows	Android
	

Chapter 4 Code Documentation

- **Code and GUI files**



- **FinalApplicationQT.pro**

```
QT       += core gui printsupport
#printsupport for notepad files
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11

SOURCES += \
    calculator.cpp \
    main.cpp \
    maindash.cpp \
    notepad.cpp \
    timer.cpp
#cpp files
HEADERS += \
    calculator.h \
    maindash.h \
    notepad.h \
    timer.h
#header files
FORMS += \
    calculator.ui \
    maindash.ui \
    notepad.ui \
    timer.ui
#GUIs
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/b
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    Resource.qrc
#resource files
```

```
QT       += core gui printsupport
#printsupport for notepad files
greaterThan(QT_MAJOR_VERSI
ON, 4): QT += widgets
CONFIG += c++11
SOURCES += \ calculator.cpp
                \ main.cpp
                \ maindash.cpp
                \notepad.cpp
                \timer.cpp
        #cpp files
HEADERS += \ calculator.h
            \ maindash.h
                \notepad.h
        \ timer.h
#header files
FORMS +=  \ calculator.ui
                    \maindash.ui
            \notepad.ui
                    \timer.ui
        #GUIs
qnx: target.path =
/tmp/$${TARGET}/bin else:
unix:!android: target.path =
/opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS
+= target
RESOURCES += \ Resource.qrc
#resource files
```

- **calculator.h**

```qt
#ifndef CALCULATOR_H
#define CALCULATOR_H
#include <QMainWindow>

namespace Ui {
class Calculator;
}

class Calculator : public QMainWindow
{
    Q_OBJECT

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

private:
    Ui::Calculator *ui;

private slots:
    void clickNumber();// for clicking numbers
    void clickMathbtn();// for operation buttons
    void clickEqualbtn();// for equal button
    void clickClearbtn();//for clear
};

#endif // CALCULATOR_H
```

```qt
#ifndef CALCULATOR_H
#define CALCULATOR_H
#include <QMainWindow>
namespace Ui { class Calculator; }
class Calculator : public
QMainWindow {    Q_OBJECT
public:    explicit
Calculator(QWidget *parent =
nullptr);
            ~Calculator();
private:    Ui::Calculator *ui;
private slots:
            void clickNumber();
          // for clicking numbers
            void clickMathbtn();
          // for operation buttons
           void clickEqualbtn();
          // for equal button
            void clickClearbtn();
          //for clear };
#endif // CALCULATOR_H
```

- **maindash.h**

```
#ifndef MAINDASH_H
#define MAINDASH_H
#include "notepad.h"
#include "calculator.h"
#include "timer.h"
#include <QMainWindow>


QT_BEGIN_NAMESPACE
namespace Ui { class MainDash; }
QT_END_NAMESPACE

class MainDash : public QMainWindow
{
    Q_OBJECT

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

public slots:
    void openNotepad();
    void openTimer();
    void openCalculator();

private slots:
    void on_notebtn_clicked();
    void on_timerbtn_clicked();
    void showTime();
    void on_calbtn_clicked();

private:
    Ui::MainDash *ui;
    NotePad*notePad;
    Timer*timer;
    Calculator* calculator;
};
#endif // MAINDASH_H
```

```
#ifndef MAINDASH_H
#define MAINDASH_H
#include "notepad.h"
#include "calculator.h"
#include "timer.h"
#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainDash; }
QT_END_NAMESPACE
class MainDash : public
QMainWindow {    Q_OBJECT
public:    MainDash(QWidget
*parent = nullptr);    ~MainDash();
public slots:
void openNotepad();
void openTimer();
void openCalculator();
private slots:
void on_notebtn_clicked();
void on_timerbtn_clicked();
void showTime();
void on_calbtn_clicked();
private:    Ui::MainDash *ui;
  NotePad*notePad;
  Timer*timer;
 Calculator* calculator; };
#endif // MAINDASH_H
```

- **notepad.h**

```
#ifndef NOTEPAD_H
#define NOTEPAD_H
#include <QFile>//for working with files
#include <QFileDialog>//dialog boxes
#include <QTextStream>//for reading texts from files
#include <QMessageBox>//for user errors
#include <QtPrintSupport/QPrinter>//ability to print
#include <QtPrintSupport/QPrintDialog>//choose the prin
#include <QMainWindow>

namespace Ui {
class NotePad;
}

class NotePad : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_actionNew_triggered();
    void on_actionOpen_triggered();
    void on_actionSave_as_triggered();
    void on_actionPrint_triggered();
    void on_actionCut_triggered();
    void on_actionCopy_triggered();
    void on_actionPaste_triggered();
    void on_actionUndo_triggered();
    void on_actionRedo_triggered();
    void on_actionExit_triggered();
private:
    Ui::NotePad *ui;
    QString currentFile="";//entered by the user
};
#endif // NOTEPAD_H
```

```
#ifndef NOTEPAD_H
#define NOTEPAD_H
#include <QFile>//for working with files
#include <QFileDialog>//dialog boxes
#include <QTextStream>//for reading texts from files
#include <QMessageBox>//for user errors
#include <QtPrintSupport/QPrinter>//ability to print
#include <QtPrintSupport/QPrintDialog>//choose the printer you need to print
#include <QMainWindow>
namespace Ui { class NotePad; }
class NotePad : public QMainWindow {    Q_OBJECT
public:    explicit NotePad(QWidget *parent = nullptr);    ~NotePad();
private slots:
void on_actionNew_triggered();
void on_actionOpen_triggered();
void on_actionSave_as_triggered();
void on_actionPrint_triggered();
void on_actionCut_triggered();
void on_actionCopy_triggered();
void on_actionPaste_triggered();
void on_actionUndo_triggered();
void on_actionRedo_triggered();
void on_actionExit_triggered();
private:    Ui::NotePad *ui;
 QString currentFile="";//entered by the user };
#endif // NOTEPAD_H
```

- **timer.h**

```
#ifndef TIMER_H
#define TIMER_H
#include <QDialog>
#include <QTimer>
#include <QMainWindow>

namespace Ui {
class Timer;
}

class Timer : public QMainWindow
{
    Q_OBJECT

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

private slots:
    void on_StartTimer_clicked();
    void on_PauseTimer_clicked();
    void on_StopTimer_clicked();
    void process();

private:
    Ui::Timer *ui;
    short int minutes;
    short int seconds;
    bool pause;
    QTimer reload;

};

#endif // TIMER_H
```

```
#ifndef TIMER_H
#define TIMER_H
#include <QDialog>
#include <QTimer>
#include <QMainWindow>
namespace Ui { class Timer; }
class Timer : public QMainWindow
{   Q_OBJECT
public:   explicit Timer(QWidget
*parent = nullptr);   ~Timer();
private slots:
void on_StartTimer_clicked();
void on_PauseTimer_clicked();
void on_StopTimer_clicked();
void process();
private:   Ui::Timer *ui;
short int minutes;
 short int seconds;
 bool pause;
 QTimer reload;
};
#endif // TIMER_H
```

- **main.cpp**

```
#include "maindash.h"
#include <QApplication>

int main(int argc, char *argv[])
//where all execution is beginned
{
    QApplication a(argc, argv);
    //QApplication function create the appl
    MainDash w;
    w.show();
    //display the main application
    return a.exec();
    //put the application into a loop
}
```

```
#include "maindash.h"
#include <QApplication>
int main(int argc, char *argv[])
//where all execution is beginned {
QApplication a(argc, argv);
//QApplication function create the
application object for us
  MainDash w;   w.show();
//display the main application
return a.exec();   //put the
application into a loop }
```

- **calculator.cpp**

```
#include "calculator.h"
#include "ui_calculator.h"
#include <QFontDatabase>
double calculatedValue=0.0;
bool divoperator=false;
bool muloperator=false;
bool addoperator=false;
bool suboperator=false;

Calculator::Calculator(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Calculator)
{
    ui->setupUi(this);
    QFontDatabase::addApplicationFont(":/font/Digital Dismay.otf");
            QFont digital("Digital dismay", 40, QFont::Normal);
            ui->display->setFont(digital);//font type added to the display

    ui->display->setText(QString::number(calculatedValue));//show the value of calculatedValue variable
    QPushButton * numberbtn[10];//array for number buttons
    for(int i = 0; i<10;++i){
        QString butName="btn"+ QString::number(i);
        numberbtn[i]=Calculator:: findChild<QPushButton*>(butName);
        connect(numberbtn[i],SIGNAL(released()),this,SLOT(clickNumber()));//connecting the number buttons with the slots
    }
    connect(ui->btnadd,SIGNAL(released()),this,SLOT(clickMathbtn()));   //connecting the add button with the slots
    connect(ui->btnsub,SIGNAL(released()),this,SLOT(clickMathbtn()));   //connecting the substract button with the slots
    connect(ui->btnmul,SIGNAL(released()),this,SLOT(clickMathbtn()));   //connecting the multiply button with the slots
    connect(ui->btndiv,SIGNAL(released()),this,SLOT(clickMathbtn()));   //connecting the divide button with the slots
    connect(ui->equal,SIGNAL(released()),this,SLOT(clickEqualbtn()));   //connecting the equal button with the slots
    connect(ui->btnclear,SIGNAL(released()),this,SLOT(clickClearbtn()));//connecting the clear button with the slots
}

Calculator::~Calculator()
{
    delete ui;
}
//method for clicking number buttons
```

```
void Calculator::clickNumber(){
    QPushButton *btn= (QPushButton *)sender();
    QString buttonValue=btn->text();//get number
     QString displayval=ui->display->text(); //display on display
     if((displayval.toDouble()==0)||(displayval.toDouble()==0.0)){
         ui->display->setText(buttonValue);
     }
         else{
         QString newVal= displayval+buttonValue;
         double dblnewval=newVal.toDouble();
         ui->display->setText(QString::number(dblnewval,'g',10));// happens after the 10th val
     }
}
void Calculator::clickMathbtn(){
    divoperator=false;
    muloperator=false;
    addoperator=false;
    suboperator=false;
    QString displayval=ui->display->text();
    calculatedValue=displayval.toDouble();
    QPushButton*btn =(QPushButton *)sender();
    QString buttonValue=btn->text();     // the operations
    if(QString :: compare(buttonValue, "/", Qt:: CaseInsensitive)==0){
        divoperator =true;
    }
    else if(QString :: compare(buttonValue, "*", Qt:: CaseInsensitive)==0){
        muloperator =true;
    }
    else if(QString :: compare(buttonValue, "+", Qt:: CaseInsensitive)==0){
        addoperator =true;
    }
    else{
        suboperator =true;
    }
        ui->display->setText("");
}
```

```
void Calculator::clickEqualbtn(){
    double result=0.0;
    QString displayval=ui->display->text();
    double newdisplayvalue=displayval.toDouble(); // conducting the operations
    if(addoperator||suboperator||muloperator||divoperator){
        if(addoperator){
            result=calculatedValue+newdisplayvalue;
        }
         else if(suboperator){
            result=calculatedValue-newdisplayvalue;
        }
        else if(muloperator){
            result=calculatedValue*newdisplayvalue;
        }
         else{
            result=calculatedValue/newdisplayvalue;
        }
    }
    ui->display->setText(QString::number(result));
}
void Calculator::clickClearbtn(){
    ui->display->setText("");   //clear
}
```

```
#include "calculator.h"
#include "ui_calculator.h"
#include <QFontDatabase>
double calculatedValue=0.0;
bool divoperator=false;
bool muloperator=false;
bool addoperator=false;
bool suboperator=false; Calculator::Calculator(QWidget *parent) :
QMainWindow(parent), ui(new Ui::Calculator) {
  ui->setupUi(this);
  QFontDatabase::addApplicationFont(":/font/Digital Dismay.otf");
QFont digital("Digital dismay", 40, QFont::Normal);
ui->display->setFont(digital);//font type added to the display
ui->display->setText(QString::number(calculatedValue));
//show the value of calculatedValue variable
QPushButton * numberbtn[10];//array for number buttons
  for(int i = 0; i<10;++i){ QString butName="btn"+ QString::number(i);
    numberbtn[i]=Calculator:: findChild<QPushButton*>(butName);
connect(numberbtn[i],SIGNAL(released()),this,SLOT(clickNumber()));
//connecting the number buttons with the slots    }
connect(ui->btnadd,SIGNAL(released()),this,SLOT(clickMathbtn()));
 //connecting the add button with the slots
connect(ui->btnsub,SIGNAL(released()),this,SLOT(clickMathbtn()));
//connecting the substract button with the slots
connect(ui->btnmul,SIGNAL(released()),this,SLOT(clickMathbtn()));
```

```
//connecting the multiply button with the slots
 connect(ui->btndiv,SIGNAL(released()),this,SLOT(clickMathbtn()));
//connecting the divide button with the slots
   connect(ui->equal,SIGNAL(released()),this,SLOT(clickEqualbtn()));
//connecting the equal button with the slots
connect(ui->btnclear,SIGNAL(released()),this,SLOT(clickClearbtn()));
//connecting the clear button with the slots }
Calculator::~Calculator() {    delete ui; } //method for clicking number buttons
void Calculator::clickNumber(){    QPushButton *btn= (QPushButton *)sender();    QString
buttonValue=btn->text();//get number
QString displayval=ui->display->text(); //display on display

  if((displayval.toDouble()==0)||(displayval.toDouble()==0.0)){
   ui->display->setText(buttonValue);     }
else{ QString newVal= displayval+buttonValue;
     double dblnewval=newVal.toDouble();
     ui->display->setText(QString::number(dblnewval,'g',10));   // happens after the 10th val     } }
void Calculator::clickMathbtn(){
divoperator=false;
muloperator=false;
addoperator=false;
suboperator=false;
      QString displayval=ui->display->text();    calculatedValue=displayval.toDouble();
      QPushButton*btn =(QPushButton *)sender();
      QString buttonValue=btn->text();     // the operations
if(QString :: compare(buttonValue, "/", Qt:: CaseInsensitive)==0){
   divoperator =true;    }
          else if(QString :: compare(buttonValue, "*", Qt:: CaseInsensitive)==0){    muloperator =true;  }
          else if(QString :: compare(buttonValue, "+", Qt:: CaseInsensitive)==0){    addoperator =true;  }
          else{suboperator =true;}
          ui->display->setText(""); }
void Calculator::clickEqualbtn(){
double result=0.0;
          QString displayval=ui->display->text();
          double newdisplayvalue=displayval.toDouble(); // conducting the operations
if(addoperator||suboperator||muloperator||divoperator){
      if(addoperator){ result=calculatedValue+newdisplayvalue;        }
             else if(suboperator){result=calculatedValue-newdisplayvalue; }
                 else if(muloperator){ result=calculatedValue*newdisplayvalue; }
            else{ result=calculatedValue/newdisplayvalue;        }    }
   ui->display->setText(QString::number(result)); }
void Calculator::clickClearbtn(){    ui->display->setText("");   //clear }
```

- **maindash.cpp**

```
#include "maindash.h"
#include "ui_maindash.h"
#include "notepad.h"
#include "calculator.h"
#include "timer.h"
#include <QTimer>
#include <QDateTime>
#include <QFontDatabase>

MainDash::MainDash(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainDash)
{
    ui->setupUi(this);
    connect(ui->notebtn,SIGNAL(click()),this,SLOT(openNotepad()));  //slot for notepad button
    connect(ui->timerbtn,SIGNAL(click()),this,SLOT(openTimer()));   //slot for timer button
    connect(ui->calbtn,SIGNAL(click()),this,SLOT(openCalculator()));//slot for Calculator button
    //Display of main dashboard
 QTimer*timer=new QTimer(this);
 connect(timer,SIGNAL(timeout()),this,SLOT(showTime()));    //slot for time
     timer->start();

     QFontDatabase::addApplicationFont(":/font/Digital Dismay.otf");
         QFont digital("Digital dismay", 40, QFont::Normal);
         ui->Digital_clock->setFont(digital);
         ui->Digital_clock->setStyleSheet("color:green");// time font and features

         QDate date=QDate::currentDate();   //get date
         QString date_text=date.toString("dd MMMM yyyy");// date format
         ui->Digital_date->setText(date_text);          //set date
          QString day_text=date.toString("dddd");       //get day
         ui->Day->setText(day_text);                    //set day
}
```

```
void MainDash::showTime(){
    QTime time= QTime:: currentTime();                    //get Time
        QString time_text=time.toString("hh : mm : ss");//time format
        ui->Digital_clock->setText(time_text);
}
MainDash::~MainDash()
{
    delete ui;
}
void MainDash::openNotepad(){
    notePad= new NotePad();
    notePad->show();
}   // method to open the notepad

void MainDash::openTimer(){
    timer =new Timer();
    timer->show();
}   // method to open the timer

void MainDash::openCalculator(){
    calculator= new Calculator();
    calculator->show();
}   // method to open the calculator
void MainDash::on_notebtn_clicked()
{
openNotepad();
}   //method assigned to button

void MainDash::on_timerbtn_clicked()
{
    openTimer();
}    //method assigned to button

void MainDash::on_calbtn_clicked()
{
   openCalculator();
}    //method assigned to button
```

```
#include "maindash.h"
#include "ui_maindash.h"
#include "notepad.h"
#include "calculator.h"
#include "timer.h"
#include <QTimer>
#include <QDateTime>
#include <QFontDatabase>
MainDash::MainDash(QWidget *parent)    : QMainWindow(parent)    ,
ui(new Ui::MainDash) {    ui->setupUi(this);
  connect(ui->notebtn,SIGNAL(click()),
this,SLOT(openNotepad()));
```

```
//slot for notepad button
   connect(ui->timerbtn,SIGNAL(click()),
this,SLOT(openTimer()));
  //slot for timer button
   connect(ui->calbtn,SIGNAL(click()),
this,SLOT(openCalculator()));
//slot for Calculator button
QTimer*timer=new QTimer(this); connect(timer,SIGNAL(timeout()),this,SLOT(showTime()));
//Display of main dashboard
timer->start();
   QFontDatabase::addApplicationFont(":/font/Digital Dismay.otf");
 QFont digital("Digital dismay", 40, QFont::Normal);
            ui->Digital_clock->setFont(digital);
            ui->Digital_clock->setStyleSheet("color:green");// time font and features
   QDate date=QDate::currentDate();   //get date
 QString date_text=date.toString("dd MMMM yyyy");// date format
 ui->Digital_date->setText(date_text);          //set date
 QString day_text=date.toString("dddd");       //get day
ui->Day->setText(day_text);                    //set day }
void MainDash::showTime(){    QTime time= QTime:: currentTime();   //get Time
QString time_text=time.toString("hh : mm : ss");//time format
ui->Digital_clock->setText(time_text); }
MainDash::~MainDash() {    delete ui; }
void MainDash::openNotepad(){    notePad= new NotePad();    notePad->show(); }
 // method to open the notepad
void MainDash::openTimer(){    timer =new Timer();    timer->show(); }   // method to open the timer
void MainDash::openCalculator(){    calculator= new Calculator();
calculator->show(); }   // method to open the calculator
void MainDash::on_notebtn_clicked() { openNotepad(); }   //method assigned to button
void MainDash::on_timerbtn_clicked() {    openTimer(); }    //method assigned to button
void MainDash::on_calbtn_clicked() {   openCalculator(); }    //method assigned to button
```

- **notepad.cpp**

```
#include "notepad.h"
#include "ui_notepad.h"

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

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

void NotePad::on_actionNew_triggered()
{
    currentFile.clear();                //clear the originally created file
    ui->textEdit->setText(QString());   //creation of a new file

}

void NotePad::on_actionOpen_triggered()
{
    QString filename=QFileDialog::getOpenFileName((this),"open the file");  //dialog box to chose the file to open
    QFile file(filename);   //object for reading and writing files
    currentFile=filename;   //store the file name
    if(!file.open(QIODevice::ReadOnly| QFile::Text)){ //open file as read only else a error message
       QMessageBox::warning(this,"warning","cannot open file:"+file.errorString());//showing the error and message
       return;
    }
        setWindowTitle(filename);
        QTextStream in(&file);      //interface for reading text
        QString text=in.readAll();  //capy text in string
        ui->textEdit->setText(text);//show text in text widget screen
        file.close();               //close file
}
```

```
void NotePad::on_actionSave_as_triggered()
{
    QString filename=QFileDialog::getSaveFileName(this,"save as");
       QFile file(filename);
       if(!file.open(QFile::WriteOnly| QFile::Text)){   //write only text
           QMessageBox::warning(this,"warning", "File cannot be saved"+ file.errorString());   //error msg
           return;
       }
       currentFile=filename;
       setWindowTitle(filename);
       QTextStream out(&file);
       QString text=ui->textEdit->toPlainText();
       out<<text;
       file.close();
}

void NotePad::on_actionPrint_triggered()
{
    QPrinter printer;   //to print use qprinter. alows to interact with any printer
    printer.setPrinterName("printer name"); //can actually put the printer name here
    QPrintDialog pdialog(&printer,this);
    if(pdialog.exec()==QDialog::Rejected){   //verify whether it was able to access to the printer
       QMessageBox::warning(this,"warning","Printer cannot be accessed!");//error message
       return;
    }
        ui->textEdit->print(&printer);
}

void NotePad::on_actionCut_triggered()
{
    ui->textEdit->cut();
}   // cut option

void NotePad::on_actionCopy_triggered()
{
     ui->textEdit->copy();
}   // copy option
```

```
void NotePad::on_actionPaste_triggered()
{
    ui->textEdit->paste();
}   // paste option

void NotePad::on_actionUndo_triggered()
{
        ui->textEdit->undo();
}   // undo option

void NotePad::on_actionRedo_triggered()
{
         ui->textEdit->redo();
}   // redo option

void NotePad::on_actionExit_triggered()
{
    QApplication::quit();
}   // Quit option
```

```
#include "notepad.h"
 #include "ui_notepad.h"
NotePad::NotePad(QWidget *parent) :    QMainWindow(parent), ui(new Ui::NotePad) {    ui-
>setupUi(this); }
NotePad::~NotePad() { delete ui; }
void NotePad::on_actionNew_triggered() {    currentFile.clear();//clear the originally created file
ui->textEdit->setText(QString());   //creation of a new file }
void NotePad::on_actionOpen_triggered() {
  QString filename=QFileDialog::getOpenFileName((this),"open the file");  //chose the file to open
QFile file(filename);   //object for reading and writing files    currentFile=filename;   //store the file name
if(!file.open(QIODevice::ReadOnly| QFile::Text)){ //open file as read only else a error message
QMessageBox::warning(this,"warning","cannot open file:"+file.errorString());    return;    }
setWindowTitle(filename);
QTextStream in(&file);      //interface for reading text
QString text=in.readAll();  //copy text in string
    ui->textEdit->setText(text);//show text in text widget screen file.close();             //close file }
void NotePad::on_actionSave_as_triggered() {
 QString filename=QFileDialog::getSaveFileName(this,"save as");
    QFile file(filename);
     if(!file.open(QFile::WriteOnly| QFile::Text)){   //write only text
QMessageBox::warning(this,"warning", "File cannot be saved"+ file.errorString());   //error msg
        return;       }
currentFile=filename;
setWindowTitle(filename);
QTextStream out(&file);
     QString text=ui->textEdit->toPlainText();       out<<text;
file.close(); }
void NotePad::on_actionPrint_triggered() {
  QPrinter printer;   //to print use q printer. allows to interact with any printer
```

```
printer.setPrinterName("printer name"); //printer name
 QPrintDialog pdialog(&printer,this);    if(pdialog.exec()==QDialog::Rejected){
//verify whether it was able to access to the printer
QMessageBox::warning(this,"warning","Printer cannot be accessed!");//error message
return;    }
   ui->textEdit->print(&printer); }
void NotePad::on_actionCut_triggered() {    ui->textEdit->cut(); }   // cut option
void NotePad::on_actionCopy_triggered() {    ui->textEdit->copy(); }   // copy option
void NotePad::on_actionPaste_triggered() {    ui->textEdit->paste(); }   // paste option
void NotePad::on_actionUndo_triggered() {    ui->textEdit->undo(); }   // undo option
void NotePad::on_actionRedo_triggered() {    ui->textEdit->redo(); }   // redo option
void NotePad::on_actionExit_triggered() {    QApplication::quit(); }   // Quit option
```

- **timer.cpp**

```
#include "timer.h"
#include "ui_timer.h"
#include <QFontDatabase>

Timer::Timer(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Timer)
{
    ui->setupUi(this);
    QFontDatabase::addApplicationFont(":/font/Digital Dismay.otf");
            QFont digital("Digital dismay", 40, QFont::Normal);
            ui->minutes->setFont(digital);
            ui->minutes->setStyleSheet("color:blue");// assigned font features for displaying minutes
            ui->seconds->setFont(digital);
            ui->seconds->setStyleSheet("color:green");  // assigned font features for displaying seconds
        seconds=0;  //assigned value for seconds
        minutes=0;  //assigned value for minutes
        pause=false;
        connect(&reload,SIGNAL(timeout()),this,SLOT(process()));    // slot for the process
}

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

void Timer::on_StartTimer_clicked()
{
    reload.start(1000);
}   //method to start timer
```

```
void Timer::on_PauseTimer_clicked()
{
    if(pause==false){
            reload.stop();
            pause=true;

        }
        else{
            pause=false;
            reload.start(1000);
        }
}   //method to pause timer

void Timer::on_StopTimer_clicked()
{
    minutes=0;
        seconds=0;
        pause=false;
        ui->minutes->setText(QString ::number (minutes));
        ui->seconds->setText(QString ::number (seconds));
        reload.stop();
}   //method to stop timer

void Timer:: process(){
    seconds=seconds+1;
         if(seconds==60){
             seconds=0;
             minutes=1;
         }
         else{
             ui->seconds->setText(QString ::number (seconds));
             ui->minutes->setText(QString ::number (minutes));
         }
}   //method to process timer
```

```
#include "timer.h"
#include "ui_timer.h"
#include <QFontDatabase>
Timer::Timer(QWidget *parent) :    QMainWindow(parent),    ui(new Ui::Timer) { ui->setupUi(this);
QFontDatabase::addApplicationFont (":/font/Digital Dismay.otf");
QFont digital("Digital dismay", 40, QFont::Normal);
ui->minutes->setFont(digital);
ui->minutes->setStyleSheet("color:blue");// assigned font features for displaying minutes
ui->seconds->setFont(digital);
ui->seconds->setStyleSheet("color:green"); // assigned font features for displaying seconds
seconds=0;  //assigned value for seconds
minutes=0;  //assigned value for minutes
pause=false;
connect(&reload,SIGNAL(timeout()),this,SLOT(process()));  // slot for the process }
Timer::~Timer() {    delete ui; }
void Timer::on_StartTimer_clicked() {
reload.start(1000); } //method to start timer
void Timer::on_PauseTimer_clicked() {
if(pause==false)
{  reload.stop();
```

```
 pause=true;
    }
 else{pause=false;
        reload.start(1000);      } }   //method to pause timer
void Timer::on_StopTimer_clicked() {
 minutes=0;
seconds=0;
pause=false;
   ui->minutes->setText(QString ::number (minutes));
   ui->seconds->setText(QString ::number (seconds));
   reload.stop(); }   //method to stop timer
void Timer:: process(){
  seconds=seconds+1;
     if(seconds==60){  seconds=0; minutes=1; }
     else{ ui->seconds->setText(QString ::number (seconds));
ui->minutes->setText(QString ::number (minutes));} }   //method to process timer
```

Chapter 5 Components and their Functions

Components	Used functions	Uses
Main dashboard	**openCalculator**();	Opens the calculator window when called.
	openNotepad();	Opens the notepad window when called.
	showTime();	To show current time
	openTimer();	Opens the timer window when called.
Calculator	**clickNumber**();	Function used to click numbers
	clickEqualbtn();	Function used to equal an operation
	clickClearbtn();	Function to reset the display
	clickMathbtn();	Function for clicking the operation buttons
Timer	**process**();	Function to calculate the seconds and minutes counted.

Chapter 7 References

www.newthinktank.com. (n.d.). *Qt Tutorial 2 : C++ Calculator*. [online] Available at: http://www.newthinktank.com/2018/06/qt-tutorial-2-c-calculator/ [Accessed 26 Dec. 2021].