Project Report

for

LAN Based Examination System

Version 1.0 approved

Prepared by **Parita Patel (201201023)**
Amisha Singla (201201026)

Dhirubhai Ambani Institute of Information and Communication Technology

26th October' 14

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this software is to make a useful application while using the concept of socket programming on any platform. This software is made to replace the small pen paper objective tests. With the help of the software the score is also calculated immediately and the examinee is given a predefined fixed amount of time after the exam starts.

1.2 Product Scope

We have created a fully functional demonstration of a LAN based examination system which includes the features like predefined timer, on spot score card. But this can be expanded in various amount to create more complex type of Exam system wherein you can choose the subject of your examination, percentage of correct answers etc.

1.3 References

- http://qt-project.org/forums/viewthread/16196
- https://www.clear.rice.edu/comp310/f12/lectures/lec26/
- http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html

2. Overall Description

2.1 Product Perspective

Our software has a huge demand along with wide range of application in real world as every practical task being replaced or influenced by the technology. This product is the one that can replace the common area and the same time for appearing in an examination. The idea is derived directly from the online examination but the students can take this test only if they are connected or part of the same network and provided that they know the I.P address of the examination server. This provides the security that no random person can take the test. Also this is a very good example to learn and understand about the aspects of the socket programming.

2.2 Client Server Architecture

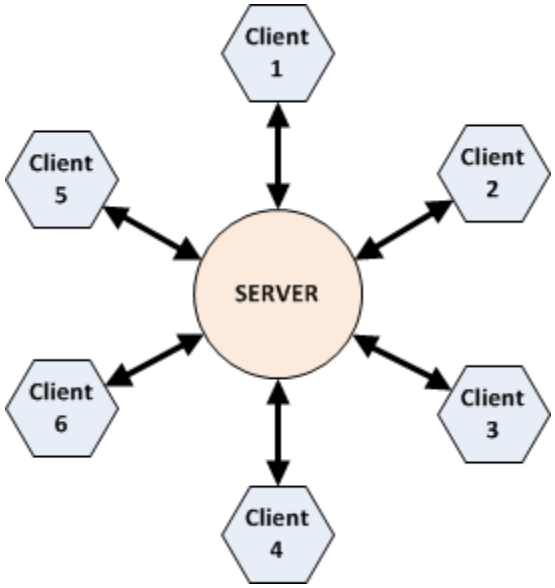

Figure: Sever as the central examiner, clients as the students

2.3 Product Functions

Our products as two parts of functionality namely the server side and the client side or the examiner and the examinee. The server sets the required question paper in a text file. When the examination starts the server gets ready to get send the question paper to the students/text file to the client. When the client successfully gets connected to the server, server sends a file containing the questions to the server.

The client has a button which starts the test along with the fixed predefined timer. The client is capable of reading a question at a time and display accordingly to the student in our UI application form along with the multiple choice options corresponding to the answer. When the student answers that question, the client writes that into another text file. When the time gets over, the test automatically gets over and the answers file is sent back to the server for checking. Even when the student completes the test and clicks the submit button, the test gets over.

The server gets the answer file of the user from the client and it calculates the score after the test by comparing the correct answer file it has with the one received from the client and calculates the score. Then it send back the corresponding score to the client on the spot so that he/she can know the performance.

3. Application

3.1 Product Application

Our product has a wide range of application in today's scenario. Major application are:

- Online examination within their network (LAN). It does not require internet connection.
- This is developed keeping in view today's examination pattern and need of today's student.
- Our Application is ideal for candidates appearing for competitive exams where they have to take multiple choice question papers.
- It gives the result immediately after completion of the examination.
- It can be improved into a powerful Test Engine for Certification Exams.

4. Code

4.1 Sever side code

```
#include <iostream>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string>

using namespace std;
using std::ifstream;
using std::string ;
#include "string";
   ifstream infile;


void dostuff(int); /* function prototype */

/* function to print out the error messages */
void error(char *msg)
{
   perror(msg);
   exit(1);
}
```

/* main function of the server where the socket is made and establishes connection with the client */

```
int main(int argc, char *argv[])
{
    // socket variables and supporting structures
    int sockfd, newsockfd, portno, clilen, pid;
    struct sockaddr_in serv_addr, cli_addr;

          // creating the socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
      error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    //serv_addr.sin_addr.s_addr = inet_addr("10.100.91.1");
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
          sizeof(serv_addr)) < 0)
          error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);

          // forking to handle multiple client requests and connections
          while (1) {
       newsockfd = accept(sockfd,(struct sockaddr *) &cli_addr,  (socklen_t*) &clilen);
       if (newsockfd < 0)
          error("ERROR on accept");
       pid = fork();
       if (pid < 0)
          error("ERROR on fork");
       if (pid == 0)  {
          close(sockfd);
          dostuff(newsockfd);
          exit(0);
       }
       else close(newsockfd);
    } /* end of while */
    return 0; /* we never get here */
}

/******** DOSTUFF() *********************
 There is a separate instance of this function
 for each connection.  It handles all communication
 once a connection has been established.
 *****************************************/
```

```
void dostuff (int sock)
{
  // opening the file that is to be sent to the client
  int n;
  infile.open("server_file.txt");
  char buffer1[1024];
  char buffer[1024];
          bzero(buffer,1024);
          bzero(buffer1,1024);
         // writing the contents of the file in the buffer
         while (!infile.eof())
         {
          infile.getline(buffer1,1024);
          strcat(buffer,buffer1);
          if(!infile.eof())
          strcat(buffer,"\n");
         }
//printf("%s",buffer);
                          // sending the buffer to the client which has the information of the question
paper.
          send(sock,buffer,sizeof(buffer), 0);


   infile.close();

        //the correct answers of the corresponding questions
    char ans_array[10];
    ans_array[0]='a';
    ans_array[1]='a';
    ans_array[2]='a';
    ans_array[3]='a';
    ans_array[4]='a';
    ans_array[5]='a';
    ans_array[6]='a';
    ans_array[7]='a';
    ans_array[8]='a';
    ans_array[9]='a';

    char buff[1024];
    bzero(buff,1024);


  //reading the answer file from the client to evaluate the score
  n = read(sock,buff,1024);

  // calculates the score of the client by comparing with the correct answers
  if (n > 0)
   {

    char score[1];
    bzero(score,1);
```

```
    int score_point =0;
     for(int i=0;i<10;i++)
      {
        if(ans_array[i] == buff[i])
          score_point++;
      }


    score[0] = score_point;

   //sends the final score after evaluation back to the client
    write(sock,score,sizeof(score));
   }


}
```

4.2 Client side code

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
#include "ui_mainwindow.h"


int main(int argc, char *argv[])
{

    // fetching I.P in argv[1] and port number in argv[2]

    QApplication a(argc, argv);
    int port = atoi(argv[2]);

    //open the main window
    MainWindow *w = new MainWindow(argv[1],port);

    w->show();

    return a.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
```

```
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(char *addr,int port, QWidget *parent = 0);
    ~MainWindow();


private slots:
    void on_pushButton_clicked();


private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "test.h"
#include "arpa/inet.h"
#include <sys/socket.h>
#include <sys/types.h>
#include "fstream"

#include "sys/types.h"
#include "netinet/in.h"
#include "netdb.h"
#include "stdio.h"
#include <unistd.h>


#include "sys/stat.h"

using std::ofstream;
using std::ifstream;
using std::string ;

ofstream file_recv;

char *server_address;
int portno;
```

```
//function to print the error message
void error(char *msg)
{
    perror(msg);
    exit(0);
}

MainWindow::MainWindow(char *address, int port, QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{

    //fetching the server address and the port number for connection
    ui->setupUi(this);
    server_address = address;
    portno = port;
    file_recv.open("client_questions.txt",ofstream::trunc);

}

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

void MainWindow::on_pushButton_clicked()
{

    //start the test when the start test button is clicked
    int sockfd, n;



    //creating the socket
    struct sockaddr_in serv_addr;
    struct hostent *server;


    char buffer[1024];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);


    if (sockfd < 0)
        error("ERROR opening socket");

   //server = gethostbyname("192.168.43.15");
   // serv_addr.sin_addr.s_addr = inet_addr("192.168.43.15");

    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }

    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
```

```
          (char *)&serv_addr.sin_addr.s_addr,
          server->h_length);

    serv_addr.sin_port = htons(portno);
    inet_pton(AF_INET,server_address,&serv_addr.sin_addr);

    if (::connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");

    //recieveing the question paper from the server
    recv(sockfd, buffer, 1024, 0);
    file_recv<<buffer;


    file_recv.close();

    Test *x = new Test(sockfd);


    this->close();
    //display the test form
    x->show();

}
```

test.h

```
#ifndef TEST_H
#define TEST_H

#include <QMainWindow>

namespace Ui {
class Test;
}

class Test : public QMainWindow
{
    Q_OBJECT

public:
    explicit Test(int a, QWidget *parent = 0);
    ~Test();


//slot for the timer
public slots:
    void MySlot();

private slots:
    void on_timeEdit_timeChanged(const QTime &time);

    void on_label_windowIconTextChanged(const QString &iconText);

    void on_pushButton_clicked();
```

```
    void on_pushButton_2_clicked();

private:
    Ui::Test *ui;
};

#endif // TEST_H
```

test.cpp

```
#include "test.h"
#include "ui_test.h"
#include "fstream"
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "arpa/inet.h"
#include <sys/socket.h>
#include <sys/types.h>

#include "netinet/in.h"
#include "netdb.h"
#include "stdio.h"
#include <unistd.h>

#include "sys/stat.h"
#include "result.h"
#include <QTime>
#include <QTimer>
#include <QtCore>
#include "string";

using std::ofstream;
using std::ifstream;
using std::fstream;
using std::string ;
int sock;
ifstream infile , ans_file;
ofstream outfile;

// Creating the timer
QTimer *timer ;

void set_question(Ui::Test);


Test::Test(int a, QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::Test)
{

     ui->setupUi(this);
     sock = a;
```

```
     //reading the input question file
     infile.open("client_questions.txt",fstream::in);
     outfile.open("client_ans.txt",fstream::out);


     // initialize the timer
     timer = new QTimer(this);
     connect(timer,SIGNAL(timeout()),this,SLOT(MySlot()));
     //timer counts after every 1s
     timer->start(1000);

     if(!infile.eof())

     //calling the function set_question to display the question on the form
     set_question(*ui);

}

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


//method to support the timer function
//MySlot() is called after every second passes from the timer
void Test::MySlot()
{
    //initializing the maximum time alloted to the client
    static int x = 120;

    //display the remaining time
    if(x > 0)
    {
        x--;
        ui->label_2->setNum(x);
    }
    else
    {
        //automatically click the submit button when the time is over and also
store the last marked answer

        if((*this->ui).option1->isChecked())
            outfile << "a";
        else if((*this->ui).option2->isChecked())
            outfile << "b";
        else if((*this->ui).option3->isChecked())
            outfile << "c";
        else if((*this->ui).option4->isChecked())
            outfile << "d";
        else
            outfile << "n";
        ui->pushButton_2->click();
       timer->stop();
    }
```

```
}

// displays the question by reading the question file sent by the seerver
void set_question(Ui::Test ui)
{
    char str[100];

    //display the question
    infile.getline(str,100);
    ui.label->setText(str);

    //display option1
    infile.getline(str,100);
    ui.option1->setText(str);

    //display option 2
    infile.getline(str,100);
    ui.option2->setText(str);

    //display option 3
    infile.getline(str,100);
    ui.option3->setText(str);

    //display option 4
    infile.getline(str,100);
    ui.option4->setText(str);


    //hide the next button for the last question
    if(infile.eof())
    {
        ui.pushButton->hide();
    }
}


// display next question when next button is clicked and store the answer
void Test::on_pushButton_clicked()
{
    //storing the answer of this question
    if((*this->ui).option1->isChecked())
        outfile << "a";
    else if((*this->ui).option2->isChecked())
        outfile << "b";
    else if((*this->ui).option3->isChecked())
        outfile << "c";
    else if((*this->ui).option4->isChecked())
        outfile << "d";
    else
        outfile << "n";

//display the next question
if(!infile.eof())
set_question(*this->ui);

}
```

```
//submits the answers of the test
void Test::on_pushButton_2_clicked()
{
    //storing the answer of the present question

        if((*this->ui).option1->isChecked())
            outfile << "a";
        else if((*this->ui).option2->isChecked())
            outfile << "b";
        else if((*this->ui).option3->isChecked())
            outfile << "c";
        else if((*this->ui).option4->isChecked())
            outfile << "d";
        else
            outfile << "n";

 outfile.close();
 //close the answer file
 ans_file.open("client_ans.txt",fstream::in);
 char buffer[1024];
 bzero(buffer,1024);
 ans_file.getline(buffer,1024);
 char score[1];

 //writing the answer file back to the server for checking
 write(sock,buffer,sizeof(buffer));
 bzero(score,sizeof(score));

 //reading the score from the server after evaluation
 read(sock, score, sizeof(score));
 result *res = new result(score[0]);
 this->close();
 res->show();
}
```

result.h

```
#ifndef RESULT_H
#define RESULT_H

#include <QWidget>

namespace Ui {
class result;
}

class result : public QWidget
{
    Q_OBJECT

public:
    explicit result(int a,QWidget *parent = 0);
    ~result();
```

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

#endif // RESULT_H
```

result.cpp

```
#include "result.h"
#include "ui_result.h"
#include "stdio.h"

result::result(int a, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::result)
{
    ui->setupUi(this);
    ui->label->setNum(a);

    //display performance according to the score
    if(a>=0 && a <=3)
        ui->label_5->setText("Poor Performance :(");
    else if(a >= 4 && a <=7)
        ui->label_5->setText("Average performance :|");
    else
        ui->label_5->setText("Excellent Performance. :)");
}

result::~result()
{
    delete ui;
}
```