TECHNICAL DOCUMENTATION

ONLINE EXAMINATION SYSTEM

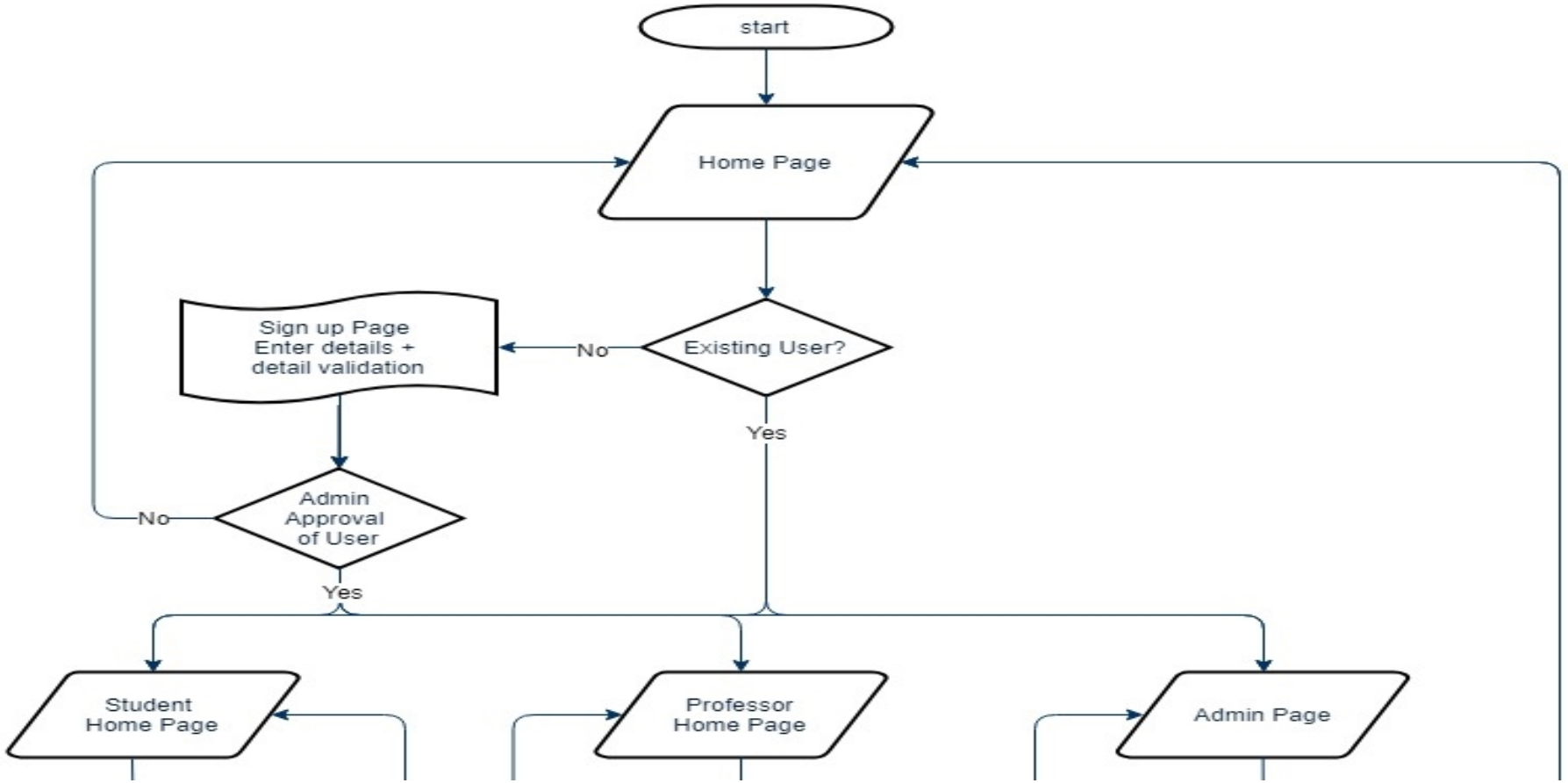
INTRODUCTION

- The application **'Online Examination System'** is based on Visual C++
- The application serves the purpose of conducting examinations for a group of students online without the use of paper to write each and every answer.
- The main features are :-
  * Login Portal for Students & Professors
  * Formation of Groups for giving tests with authentication for any student to join it.
  * Setting of Question Paper and distribution of marks to the Questions by Professors for a particular group
  * Generation of Results and Certificates along with showing the correct & incorrect Answers.
  * Display of Upcoming Tests & Past Tests given by the student to judge his performance and prepare accordingly.
  * Cross-checking of Scripts for plagiarism check and Cheating b/w students by the admin.

SYSTEM REQUIREMENTS

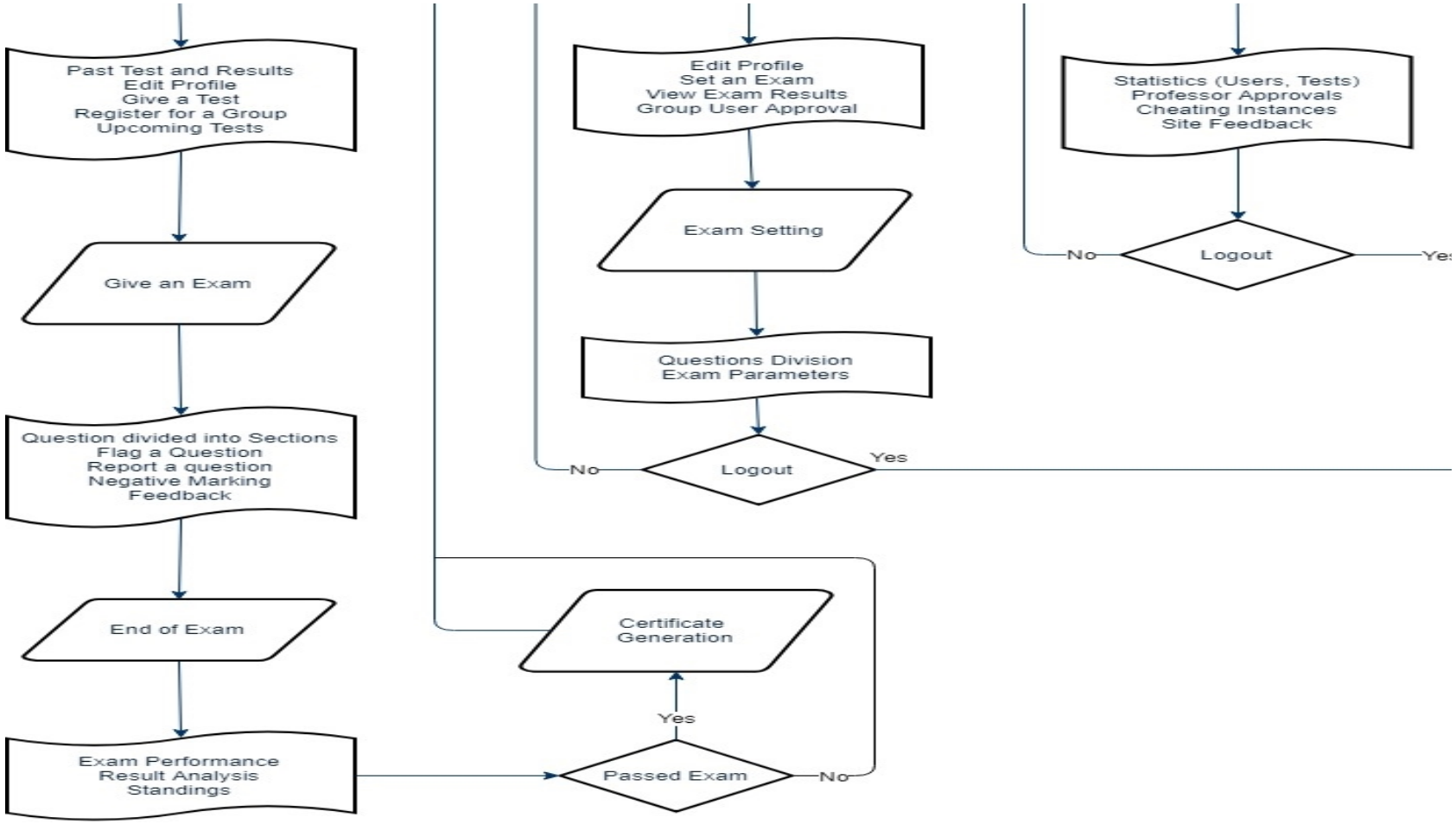
- **Operating System** – Windows XP or higher
- **Processor Speed (Clock Speed)** – 1 GHz or higher
- **RAM** – 1 GB or Higher
- **Hard Disk Space** – 256 GB or higher
- **Graphic Card** - DirectX 9 or later with WDDM 1.0 driver
- **Display** – Works best on 1366 * 768 pixels
- **Visual Studio Version** – 2013 or Higher

LOGIN AND SIGNUP FLOW



DASHBOARD

STUDENTS , PROFESSORS AND ADMIN



FEATURES IN THE PROJECT

- Login Portal for Students & Professors
- Formation of Groups for giving tests with authentication for any student to join it.
- Setting of Question Paper and distribution of marks to the Questions by Professors for a particular group
- Generation of Results and Certificates along with showing the correct & incorrect Answers.
- Display of Upcoming Tests & Past Tests given by the student to judge his performance and prepare accordingly.

LOGIN

```
stem::Void btnLogin_Click(System::Object^  sender, System::EventArgs^  e)

if (Username->Length == 0 || Password->Length == 0)
{
    MessageBox::Show("Empty Username/Password");
    return;
}
// access database and get all values
OES ^Access = gcnew OES();
Access->AddParam("@Username", Username);
Access->ExecQuery("Select Username, FullName, PasswordHash, PasswordSalt, TokenHash, TokenSalt, Email, PhoneNo,
if (Access->DBDT->Rows->Count == 0 || Access->Exception->Length)
{
    MessageBox::Show("Username\\Password is wrong");
    return;
}
```

- On clicking the Login Button , a querry is passed by the program to check if the username exists in the database.
- If it exists, it checks the hash of the password entered with the hash of password already stored in database.
- If the two values match, user is taken to his/her **DASHBOARD**

LOGIN PAGE

- The Features include:
  - Forget password option(**linkLabel1**) which after clicking takes to a form that asks for username and email and if valid, sends a token to email.
  - Signup option(**linkLabel2**) which when clicked loads **Signup.h** form.

- The querry firstly checks if the username exists in the **Users** table of the database. If it exists, it authenticates the user and takes him to the Dashboard (given by **studForm.h** or **profForm.h**)

ADMIN DASHBOARD

- The Admin dashboard is the form **AdminForm.h** which loads when the admin logins in from the **Login.h** form.
- When the admin clicks the update professor button **BtnApproveProf**, the form **ApproveProf.h** opens which contains the necessary utilities to help the admin approve the professor profile creation requests.
- When the admin clicks the Users button **Students**, the form **StudentEditAdmin.h** opens which contains the necessary utilities to help the admin see and edit student profile details.
- When the admin clicks the Users button **examDetails**, the form **ExamList.h** form opens which contains the necessary utilities to help the admin see and edit the upcoming exam's details.

ADMIN DASHBOARD(CONTINUED)

```
private: System::Void Leaderboard_Load(System::Object^  sender, System::EventArgs^  e) {
            dt = gcnew DataTable();
            Session_Num = gcnew OES();
            Session_Num->ExecQuery("SELECT NumSessions FROM  Exam WHERE(ExamCode = "+this->examCode+" )");
            Int32 num_ses = Convert::ToInt32(Session_Num->DBDT->Rows[0]["NumSessions"]);
            if (num_ses < 1){
                btnSession1->Hide();
            }
            if (num_ses < 2){
                btnSession2->Hide();
            }
            if (num_ses < 3){
                btnSession3->Hide();
            }
            if (num_ses < 4){
                btnSession4->Hide();
            }
            if (num_ses < 5){
                btnSession5->Hide();
            }

            Access = gcnew OES();
            Access->ExecQuery("SELECT ExamCode, Username, ObtainedMarks\
                FROM     Performance\
                WHERE(ExamCode = "+ this->examCode +" )\
                ORDER BY ObtainedMarks DESC");
            dsa = gcnew DataSet();
            Access->DBDA->Fill(dsa, "Performance");
            standings->DataSource = dsa->Tables[0];
```

- When the admin clicks the Users button **btnChangePasswd**, the form **ChangePassword.h** form loads which contains the necessary utilities to help the admin change their password.
- When the admin clicks the **leaderboard** button, the **Leaderboard.h** form loads which takes data from the database and ranks and shows the students.

SIGNUP

Online_Exam::prof_signup — prof_signup_Load(System::Object ^ sender, System::EventArgs ^ e)

```
if (validate())
{
    try
    {
        String ^ PassSalt = MakeSalt(10);
        String ^ PassHash = EncryptPassword(passTxt->Text, PassSalt);

        Access->AddParam("@Username", userTxt->Text);
        Access->AddParam("@Fullname", nameTxt->Text);
        Access->AddParam("@PasswordHash", PassHash);
        Access->AddParam("@PasswordSalt", PassSalt);
        Access->AddParam("@Email", mailTxt->Text);
        Access->AddParam("@Phoneno", pNumTxt->Text);
        Access->AddParam("@Branch", branchCb->Text);
        Access->AddParam("@Designation", des);

        //Access->ExecQuery("insert into [Users] ( [Username],[FullName],[PasswordHash],[PasswordSalt],[Email],[PhoneNo],[Branch],[Designa
        Access->ExecQuery("insert into [Users] ( [Username],[FullName],[PasswordHash],[PasswordSalt],[Email],[PhoneNo],[Branch],[Designati


        MessageBox::Show("Signup Successful");
    }
    catch (Exception^ ex)
```

- Signup Form basically asks for some user details which is then validated.
- If the info provided is valid, the password salt and password hash is made and a querry is passed to store the user details in the database along with hash & salt values.
- After success of the querry , **SIGNUP SUCCESSFUL** pops up lest the error message is shown.

SIGNUP
STUDENT/PROFESSOR

```
private: System::Void studBtn_Click(System::Object^  sender, System::EventArgs^  e) {
             OES^ Access = gcnew OES();
             String^ des = "Student";
             int check = 0;
             if (memChkBox->Checked)
                 check = 1;
             if (validate()){
                    // MessageBox::Show(Convert::ToString(memChkBox->Checked));
                     String ^ PassSalt = MakeSalt(10);
                     String ^ PassHash = EncryptPassword(passTxt->Text, PassSalt);

                     Access->AddParam("@Username", userTxt->Text);
                     Access->AddParam("@Fullname", nameTxt->Text);
                     Access->AddParam("@PasswordHash", PassHash);
                     Access->AddParam("@PasswordSalt", PassSalt);
                     Access->AddParam("@Email", mailTxt->Text);
                     Access->AddParam("@Phoneno", pNumTxt->Text);
                     Access->AddParam("@Rollno", rNumTxt->Text);
                     Access->AddParam("@Branch", branchCb->Text);
                     Access->AddParam("@Designation", des);
                     //Access->ExecQuery("insert into [Users] ( [Username],[FullName],[PasswordHash
                     Access->ExecQuery("insert into [Users] ( [Username],[FullName],[PasswordHash],
                MessageBox::Show("Signup Successfull");
             }
}
```

- On clicking the button **studBtn**, **signup_student.h** form loads which has the following feature:
  - The student can enter details such as username, full name, roll number, email, phone number, password, branch and if he is an IITG member.

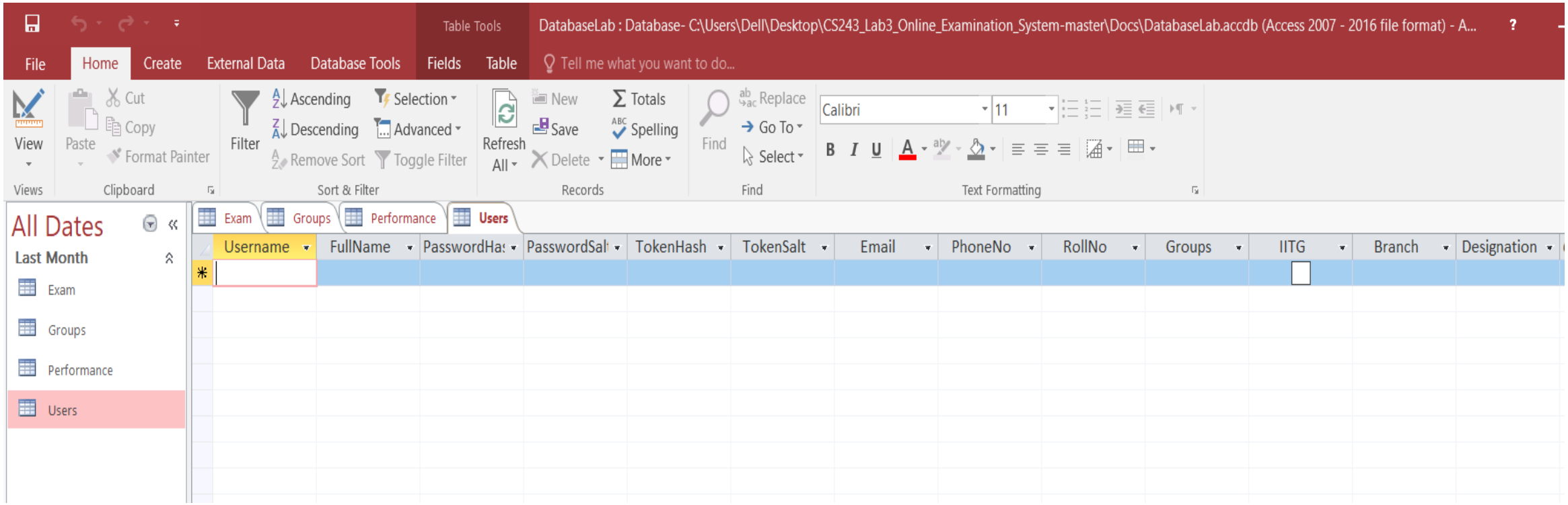
DATA VALIDATION IN SIGNUP

- *Data Validation:*
  - The roll number can only be an integer.
  - The phone number can only be an integer.
  - The Username shouldn’t contain space.
  - All the names are trimmed.
  - The password shouldn’t contain space.

```
    OES^Access = gcnew OES();
    String ^str = userTxt->Text;
    Access->AddParam("@userName", str);
    Access->ExecQuery("SELECT * FROM Users WHERE Username =@userName");
    if (Access->DBDT->Rows->Count != 0 || Access->Exception->Length)
    {
        MessageBox::Show("UserName exist Already or error Occurs");
        return false;
    }


    if (passTxt->Text != confirmPassTxt->Text)
    {
        MessageBox::Show("Password do not match");
        passTxt->Clear();
        confirmPassTxt->Clear();
        return false;
    }
    /*if (pNumTxt->TextLength != 10)
    {
        MessageBox::Show("Phone Number Length should be 10");
        return false;
    }

    if (userTxt->Text->Trim() == "")
    {
        MessageBox::Show("UserName can not be Empty");
        return false;
    }*/
    for (int i = 0; i < pNumTxt->TextLength; i++)
    {
        if (pNumTxt->Text[i]>'9' || pNumTxt->Text[i] < '0')
        {
            MessageBox::Show("Phone Number should consist only digits 0-9", "Wrong Details");
```

validate function to validate data in **profSignup.h**

USERS TABLE IN THE DATABASE



The Querry first checks if a user with the same username exists as provided in the **profSignup.h** or **studSignup.h** form . If it doesn't exist , querry makes a new entry in the **Users** table in the database with the following fields after validation of data.

APPROVING PROFESSORS

ADMIN

```
private: System::Void btnUpdate_Click(System::Object^  sender, System::EventArgs^  e) {
             cmdb = gcnew OleDbCommandBuilder(Access->DBDA);
             Access->DBDA->Update(dsa, "Users");
             dsa->Clear();
             OES ^ Access1 = gcnew OES();
             Access->RecordCount = Access->DBDA->Fill(dsa, "Users");
             dt = dsa->Tables[0];

             if (Access->RecordCount == 0){
                 this->profList->Hide();
                 this->textBox1->Hide();
                 label1->Show();
             }
             else{
                 this->profList->Show();
                 this->textBox1->Show();
                 label1->Hide();
             }
             //this->profList->Columns["isApproved"]->ReadOnly = false;
```

- The function **ApproveProf_Load()** loads a list of un approved professors and the admin is asked to check the new professors.
- On clicking the Update button, the **btnUpdate_Click()** function accesses and updates the database.

FORMING NEW GROUPS

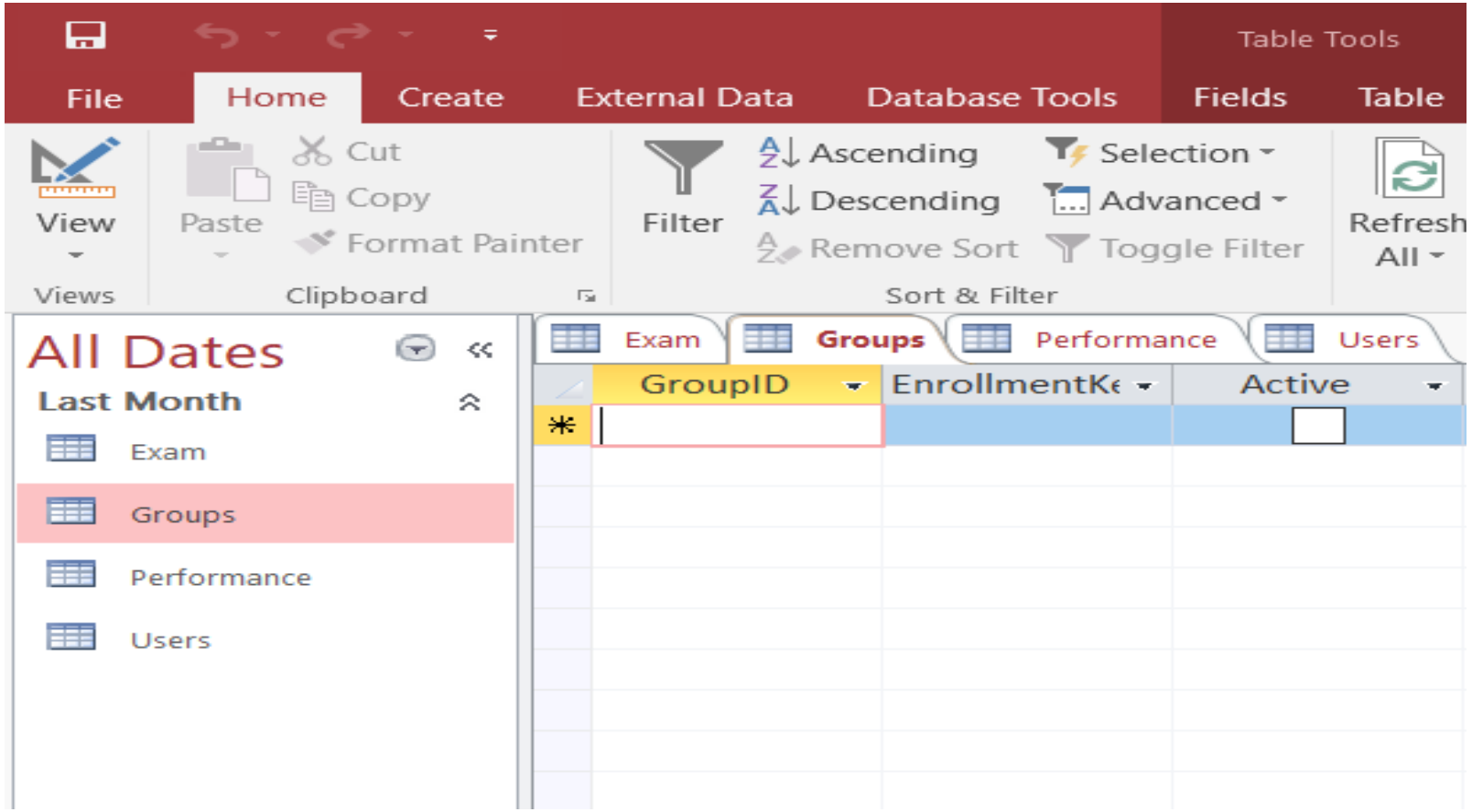
PROFESSORS

Online_Exam::CreateGroup — CreateGroup_Load(System::Object ^ sender, System::EventArgs ^ e)

```
private: System::Void btnCreate_Click(System::Object^  sender, System::EventArgs^  e) {
            if (checkBox->Checked == true)
            {

                OES ^Access = gcnew OES();
                Access->ExecQuery("insert into Groups (GroupID, EnrollmentKey) values('" + txtGroupName->Text + "', '" + txtEnroll->Text + "')");

            }
            else
            {
                OES ^Access = gcnew OES();
                Access->ExecQuery("insert into Groups (GroupID) values('" + txtGroupName->Text + "')");

            }

            MessageBox::Show("Group successfully created");
}
private: System::Void checkBox_CheckedChanged(System::Object^  sender, System::EventArgs^  e) {
            if (checkBox->Checked == true)
            {
                txtEnroll->Show();
                lblEnroll->Show();
            }
            else
            {
                txtEnroll->Hide();
```

- The form asks the user to give a name to the group. If the name already exists, message pops up for the same.
- If the checkbox of enrolment key is checked , the group is private otherwise the group is public.
- As per the checkbox status, a group is added to the groups table in the database.
- A **GROUP SUCCESSFULLY CREATED** message pops up on successful creation of the group.

GROUPS TABLE IN DATABASE



1)The group Data is stored in the **Groups** table of our database.

2)The form contains a checkbox which if signed, asks for a enrolment key for the group.

3)The querry passes the text in enrolment key(if applicable) along with group name to the **Groups** table.

4) The code to this feature is present in **createGroup.h**

ENROLLING IN GROUP

STUDENTS

```
Online_Exam::GroupEnroll                    GroupEnroll_Load(System::Object ^ sender, System::EventArgs ^ e)
    String ^cur_group;
    if (Access->RecordCount > 0)
    {
        studGroup = Convert::ToString(Access->DBDT->Rows[0]["GroupID"]);
        Access1->ExecQuery("Select * from Users where Username = '" + gVar::Username + "' and Groups Like '%-" + studGroup + "-%'");
      //  MessageBox::Show("Select * from Users where Username = '" + gVar::b + "' and Groups Like '%-" + studGroup + "-%'");
        if (Access1->RecordCount > 0)
        {
            MessageBox::Show("You are already enrolled to the group");
        }
        else{
            if (txtEnroll->Text == Convert::ToString(Access->DBDT->Rows[0]["EnrollmentKey"]))
            {
                Access->ExecQuery("Select * from Users where Username= '" + gVar::Username + "'");
                cur_group = Convert::ToString(Access->DBDT->Rows[0]["Groups"]);
                cur_group = cur_group + "-" + studGroup + "-";
                Access->ExecQuery("UPDATE Users SET Groups='" + cur_group + "' WHERE Username = '" + gVar::Username + "'");
                MessageBox::Show("successfully enrolled!");
                txtEnroll->Clear();
                comboGroupName->Text = "";
            }
            else
            {
                MessageBox::Show("Wrong enrollment key");
            }
        }
```

- Here , the user is asked to specify the name of group . With the username being primary key , it is firstly checked if the user is already enrolled in the group.
- If user is not enrolled and the enrolment key is correct, the querry passed returns a success message **SUCCESSFULLY ENROLLED**
- Otherwise, **Wrong Enrollment Key** pops up as a message.

CREATING EXAMS

PROFESSORS

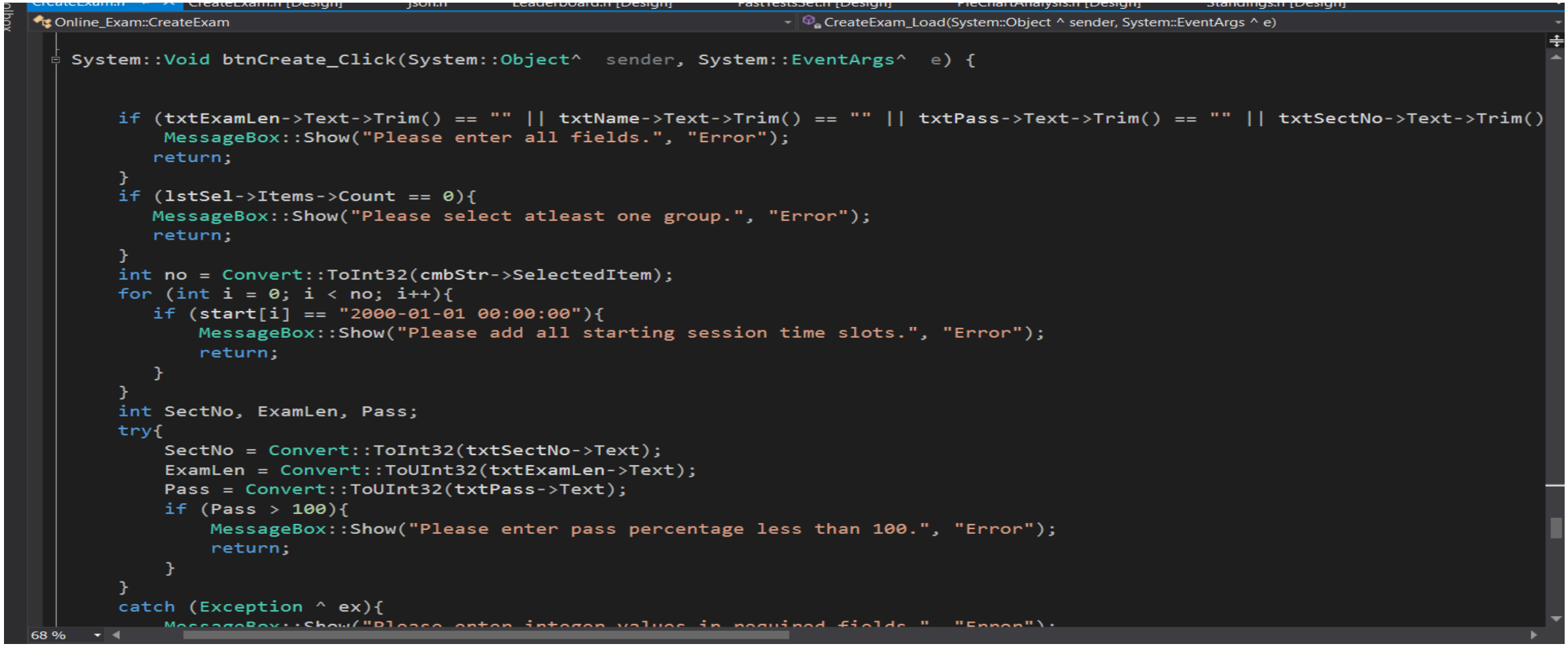
line_Exam::CreateExam ▾ btnRem_Click(System::Object ^ sender, System::EventArgs ^ e) ▾

```
private: System::Void btnAdd_Click(System::Object^  sender, System::EventArgs^  e) {
            if (lstUnsel->SelectedIndex == -1)
                return;
            String ^ selItemText = Convert::ToString(lstUnsel->SelectedItem);
            int selItemIndex = lstUnsel->SelectedIndex;
            lstSel->Items->Add(selItemText);
            if (lstUnsel->Items->Count != 0){
                lstUnsel->Items->RemoveAt(selItemIndex);
            }
            gSel[selItemText] = 1;
}
private: System::Void btnRem_Click(System::Object^  sender, System::EventArgs^  e) {
            if (lstSel->SelectedIndex == -1)
                return;
            String ^ selItemText = Convert::ToString(lstSel->SelectedItem);
            int selItemIndex = lstSel->SelectedIndex;
            lstUnsel->Items->Add(selItemText);
            if (lstSel->Items->Count != 0){
                lstSel->Items->RemoveAt(selItemIndex);
            }
            gSel[selItemText] = 0;
}
```

- The professor adds the group for which the exam is created. The groups added will only be eligible to give the tests and thus the students enrolled in the relevant groups are eligible.
- The Professor selects the number of sessions for conducting exams .
- The number of sections present in the exam along with weightage and number of questions per section are also fed by the professor. Also, the time allocated is selected.
- All these features are then put up into a specific format where questions pop up based on the section.

DATA VALIDATION FOR CREATING EXAMS

PROFESSORS



STORING EXAMINATION DETAILS IN DATABASE

```
String ^output = JsonConvert::SerializeObject(js, Formatting::None);
Accessqwert->AddParam("@QuestionSet", output);
Accessqwert->ExecQuery("INSERT Into Exam (ExamName, Professor, GroupID, NumSections, WgtSections, ExamLength, MaxScore, PassPer
MessageBox::Show("Questions Saved. Test Created Successfully");
this->Close();
```

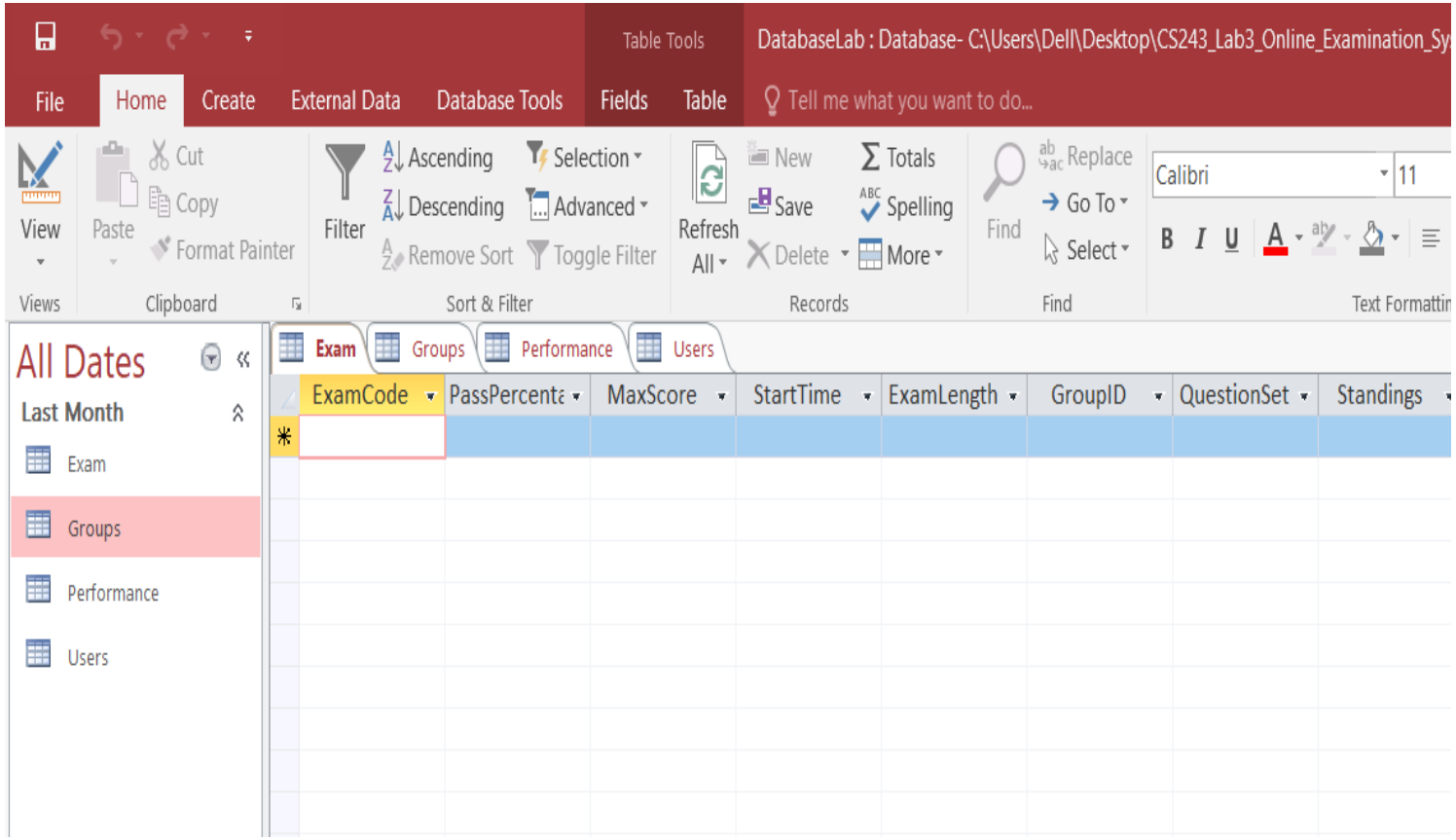
The **querry** inserts the name of examination , professor, group for which it is meant to be set, number of sections, weight of sections , exam length , max score and passing score in the **Exams** Table of Database.

All these values are set by the Professor

These values are added after certain validations on the data entered as text

The code is present in **ExamPaper.h**

EXAMS TABLE IN DATABASE



The **Exams** table in database is used for saving data which can be utilized for generating results as well as for storing the general features of the exams for the respective group.

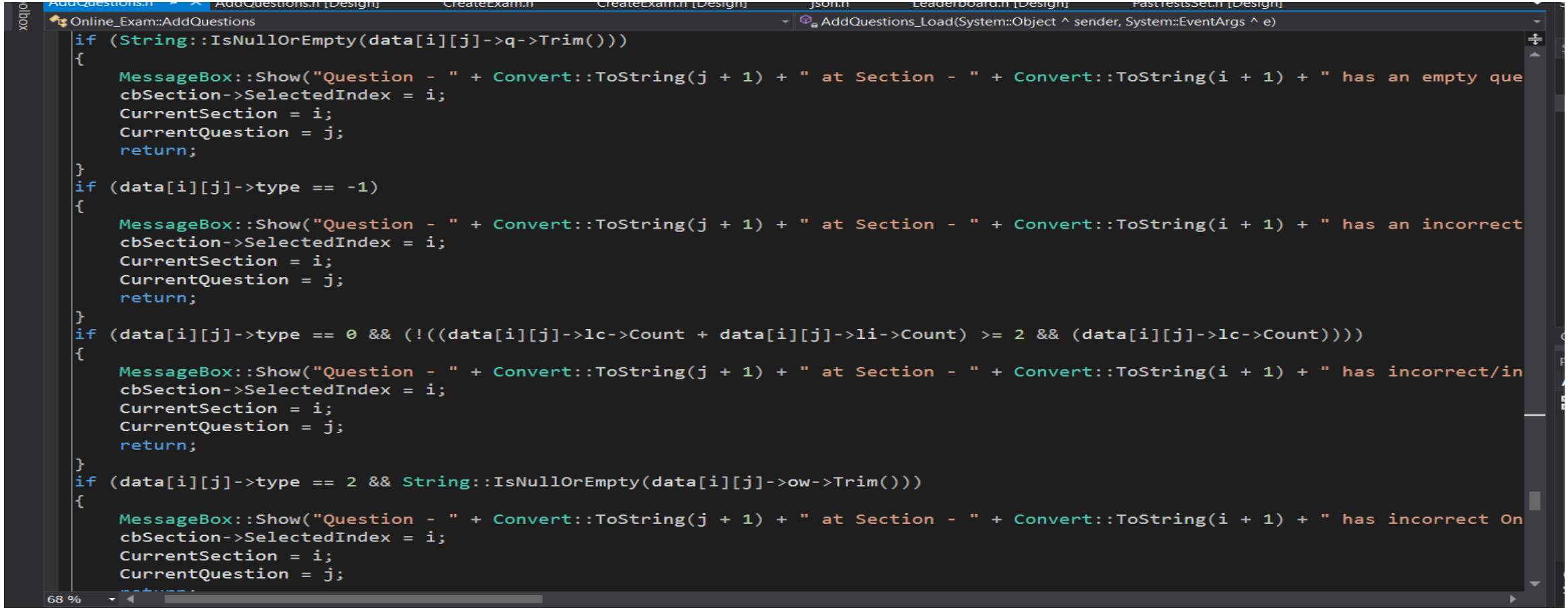
All the values are set after validation. Any wrong value set is shown from the Message Box pop up.

ADDING QUESTIONS

PROFESSORS

```
nline_Exam::AddQuestions                    AddQuestions_Load(System::Object ^ se
private: System::Void AddQuestions_Load(System::Object^  sender, System::EventArgs^  e) {
    cbSection->Items->Clear();
    for (int i = 0; i < SectionQues->Length;i++)
        cbSection->Items->Add(i+1);
    cbSection->SelectedIndex = 0;
    btnPrev->Visible = false;
    if (SectionQues[0] == 1)
        btnNext->Visible = false;
    label5->Text = Convert::ToString(SectionQues[0]);
}
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
    btnNext->Visible = true;
    if (CurrentQuestion >=2)
        btnPrev->Visible = true;
    else btnPrev->Visible = false;

    SaveData(CurrentSection, CurrentQuestion);
    CurrentQuestion--;
    LoadData(CurrentSection, CurrentQuestion);
}
private: System::Void btnNext_Click(System::Object^  sender, System::EventArgs^  e) {
    if (CurrentQuestion < SectionQues[CurrentSection] - 2)
        btnNext->Visible = true;
    else btnNext->Visible = false;
    btnPrev->Visible = true;
```

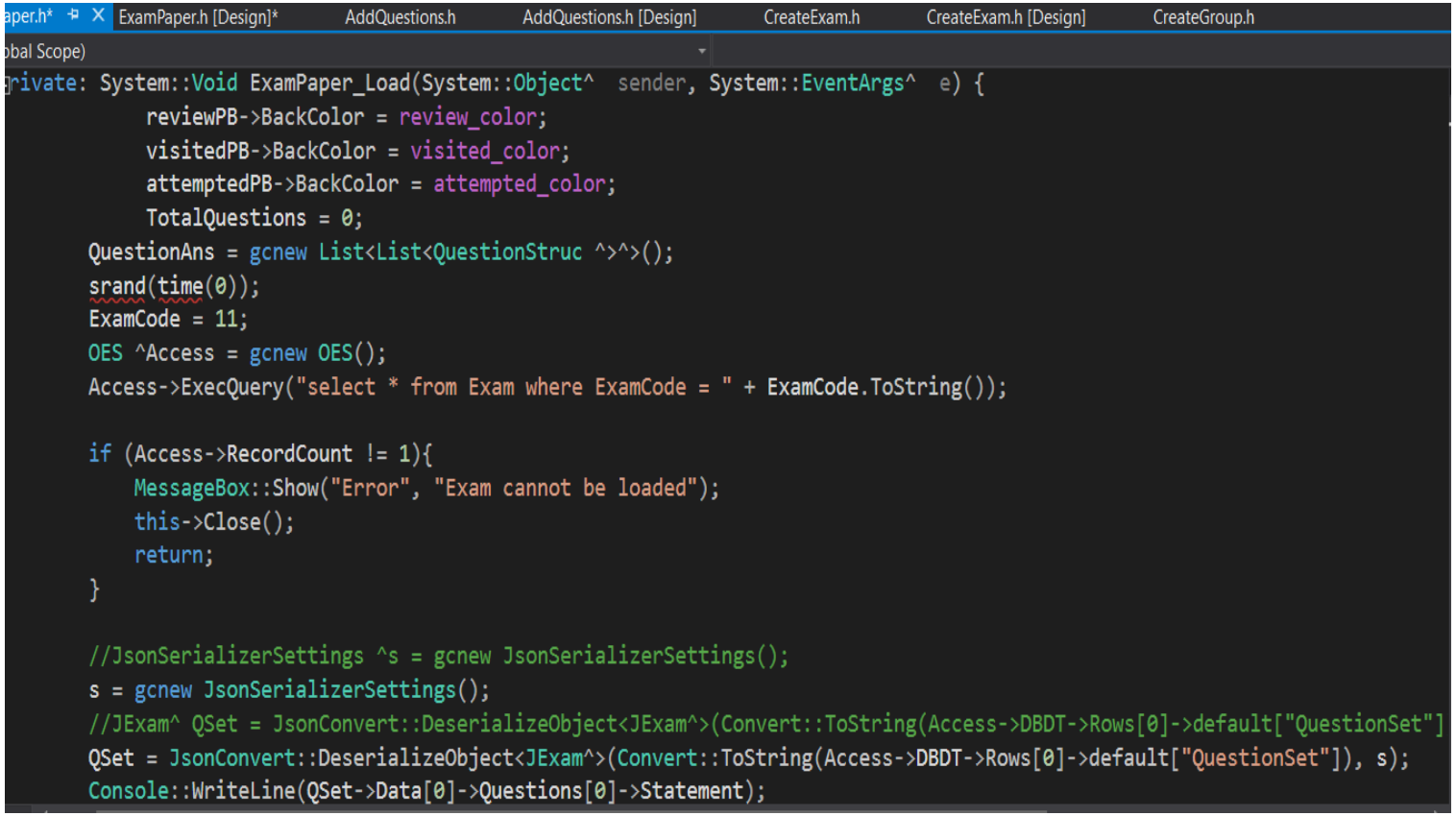
- After forming the basic exam layout, the professor is directed to adding questions.
- Based on number of sections selected by professor & number of questions per section, the professor adds questions in each section.
- After adding question of type say MCQ, the professor feeds the correct option and incorrect options which are presented randomly during giving examination.
- After successfully adding the question, it is saved in the *.json* file which is loaded during the examination.



DATA VALIDATION FOR DIFFERENT TYPES OF QUESTIONS in **AddQuestions.h**

GIVING EXAMINATION ☺

STUDENTS



- The exam paper is loaded if the exam-code is correct. Otherwise, error message pops up.
- The *.json* file is loaded where the details of questions of the examination are stored. The questions are then seen on the screen with a Legend on the left showing questions with different type of question type i.e. **‘Visited’** , **‘Attempted’** & **‘Review’** .

LOADING THE EXAM PAPER ☺ . . . . .

```
TotalQuestions = 0;
QuestionAns = gcnew List<List<QuestionStruc ^>^>();
srand(time(0));
OES ^Access = gcnew OES();
Access->ExecQuery("select * from Exam where ExamCode = " + ExamCode.ToString());

if (Access->RecordCount != 1){
    MessageBox::Show("Error", "Exam cannot be loaded");
    this->Close();
    return;
}
```

The Exam Paper set by the prof is stored in the database with a exam-code.

After the exam-code is selected , the exam is loaded as per **examPaper.h**

VARIOUS SECTIONS OF EXAMINATION

```
            btnPaper[sectionNo][curQuesNo - 1] = btn;
        }
    }*/
    if (attempted[sectionNo]->default[curQuesNo - 1] == 1)
    {
        Button ^ btn = gcnew Button();
        btn = btnPaper[sectionNo]->default[curQuesNo - 1];
        btn->BackColor = attempted_color;
        btnPaper[sectionNo]->default[curQuesNo - 1] = btn;
    }
    else
    {
        Button ^ btn = gcnew Button();
        btn = btnPaper[sectionNo]->default[curQuesNo - 1];
        btn->BackColor = visited_color;
        btnPaper[sectionNo]->default[curQuesNo - 1] = btn;
    }
```

Different Sections of the Test Paper are shown with different colours for the convenience of user.

There are unvisited, marked for review & attempted types of questions in the examination.

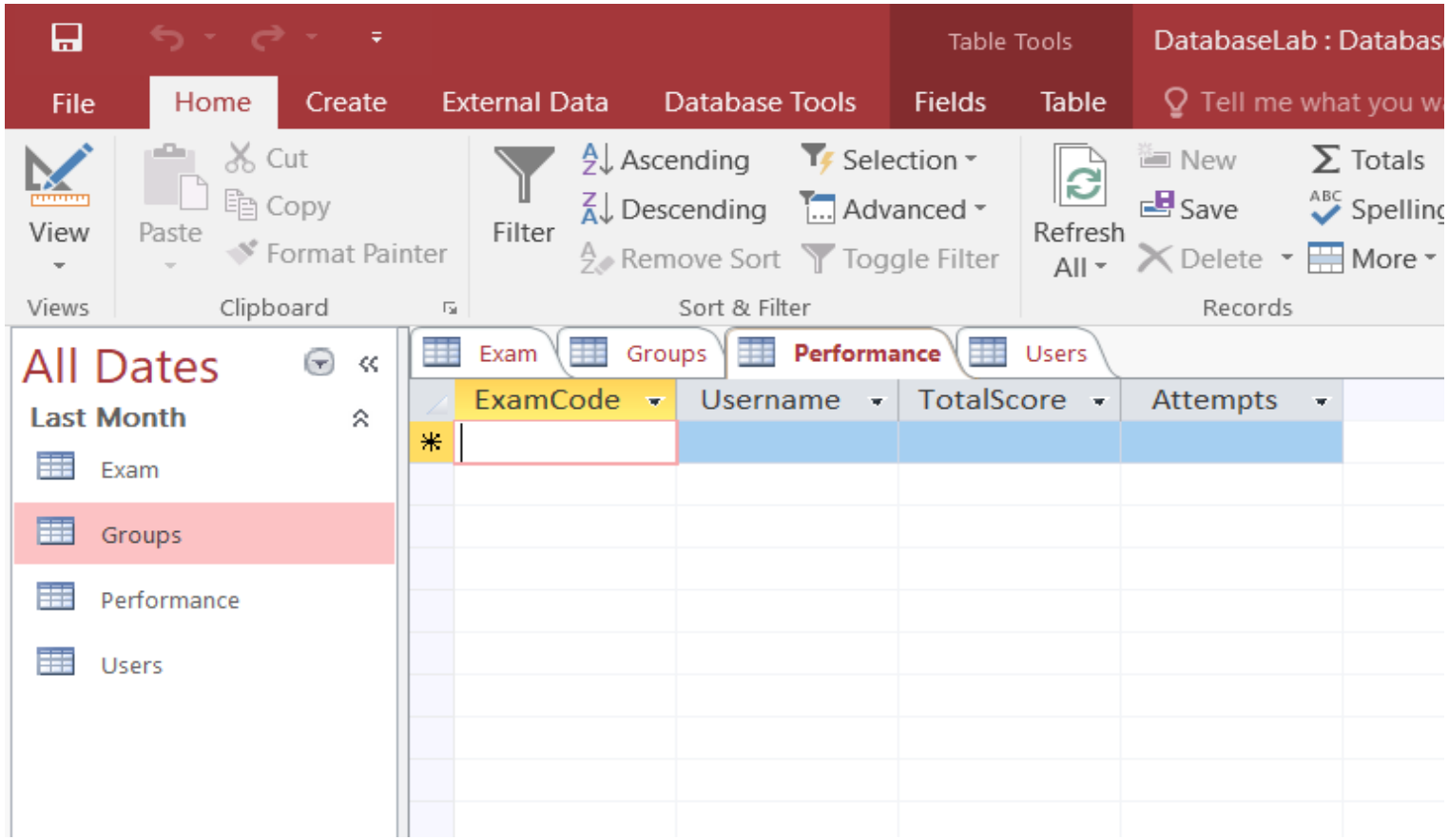
SHOWING ANSWERS & RESULTS

```
OES^ Access1 = gcnew OES();
Access1->ExecQuery("Select * from Performance where Username = '" + gVar::Username + "' AND ExamCode = " + ExamCode.ToString(
if (Access1->RecordCount != 1){
    MessageBox::Show("Error in obtaining performance", "Error");
    return;
}
String ^s = Environment::NewLine;
array<String^>^delimiters = { s };

SessNo = static_cast<int>(Convert::ToInt32(Access1->DBDT->Rows[0]->default["SessionNumber"]));
lblSesNo->Text = SessNo.ToString();

QuesGiven = gcnew List<List<int>^>();
AttemptAns = gcnew List<List<String ^>^>();
AttemptAnsStr = Convert::ToString(Access1->DBDT->Rows[0]->default["AttemptedAns"])->Split(delimiters, StringSplitOptions::None
QuesGivenStr = Convert::ToString(Access1->DBDT->Rows[0]->default["QuesGiven"])->Split(delimiters, StringSplitOptions::None);
ObtainedMarks = static_cast<int>(Convert::ToInt32(Access1->DBDT->Rows[0]->default["ObtainedMarks"]));
int sectProc = -1;
for (int i = 1; i < AttemptAnsStr->Length; i++){
    String^ a="", ^b ="";
    int done = 0;
    for (int j = 0; j < QuesGivenStr[i]->Length; j++){
        if (QuesGivenStr[i][j] != '.'){
```

The querry fetches the whole row from the **Performance** table in the database from the Username and Exam-Code. If there is no record , an error message is shown .

As per the Session Number & Data stored in Database after addition of Marks, the result is shown to user in the form **displayAnswer.h**

PERFORMANCE TABLE IN THE DATABASE



The **Performance** table in the Database stores the results of all the users for a given exam specified by a unique exam-code.

The database stores Total Score and attempts of the User for the examination which can then by used to show the result to user in form of **Pie-Chart** or **Bar-Graph**

GENERATING PASSING CERTIFICATES

If the user goes to **Certificates** option in **StudentForm.h** , he is certified to have passed in the exam he gave before the date he accesses the Application given he meets the passing percentage criteria set up by the prof in **createExam.h**

DIFFERENT OPTIONS IN STUDENT DASHBOARD

- **My Profile** – To Show the current Details of User like Full Name, Phone Number, Email , Roll Number, etc.
- **Edit Profile** – To Edit details of the User stored in **Users** table using a querry same as that in **studentSignup.h**
- **Change Password** – To Change the Password of the user which is updated in **Users** table
- **Upcoming Tests** – Show the Tests meant for the group he is enrolled in as per the current time
- **Past Tests** – To know which test he has appeared in past and get the result by redirecting to **displayAnswer.h**
- **Enroll to a Group** – To enrol to a new group (if it is public or if he has the enrolment key for private groups)
- **Unenroll me from a Group** – To un-enrol from any group he is already enrolled in.
- **Certificates** – To print certificates of any exam he has passed in .
- **Log out** – To go back to the home page of **Login/Signup { Online_Exam.h }**

DIFFERENT OPTIONS IN PROFESSOR DASHBOARD

- **My Profile** – To Show the current Details of User like Full Name, Phone Number, Email , Roll Number, etc.
- **Edit Profile** – To Edit details of the User stored in **Users** table using a querry same as that in **studentSignup.h**
- **Change Password** – To Change the Password of the user which is updated in **Users** table .
- **Create a Test** – To create a new test for the students. This is re-directed to **createExams.h**
- **Past Tests Set** – To show the list of previous Tests he set using a querry by accessing **Exams** & **Users** tables.
- **Create a group** – To create a New Group for the students. This is re-directed to **createGroup.h**
- **Log out** – To go back to the home page of **Login/Signup { Online_Exam.h }**

NEW FEATURES TO BE ADDED

- Plagiarism and cheating detection
- Setting up an Andriod Application relevant to the same and working on the same backend.
- Ability to print the Question Paper, Answer Sheets, Results & Analysis, Certificates, etc.
- Adding Calendar feature in the dashboard with pop-up notifications for results of previous tests & reminder notifications for future Tests.
- Feedback System on the Question Paper & Solution/Answer Key.
- Feedback System for Incorrect Question & Incorrect Choices of Answers.
- Adding up of Lecture Notes, Video Tutorials & Sample Questions relevant to the group/test.