**MẪU BÁO CÁO TIẾN ĐỘ ĐỒ ÁN TIỂU LUẬN**

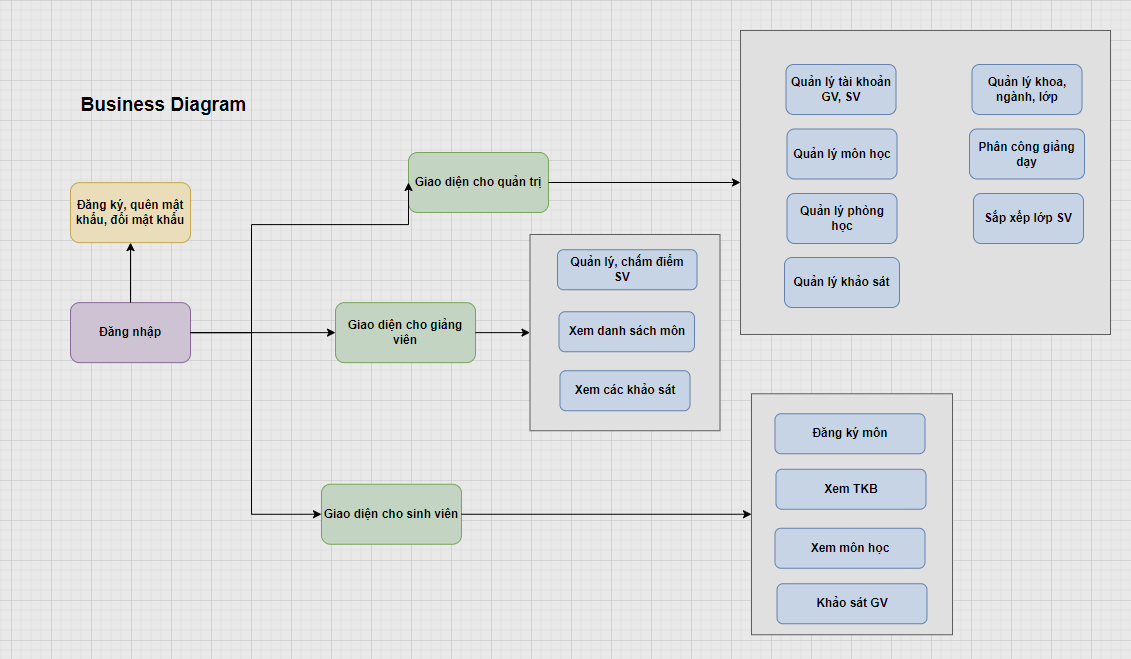
**Học phần: Lập trình ứng dụng trên Windows**

**Đến ngày: 22/6/ 2023**

**TÊN ĐỀ TÀI**

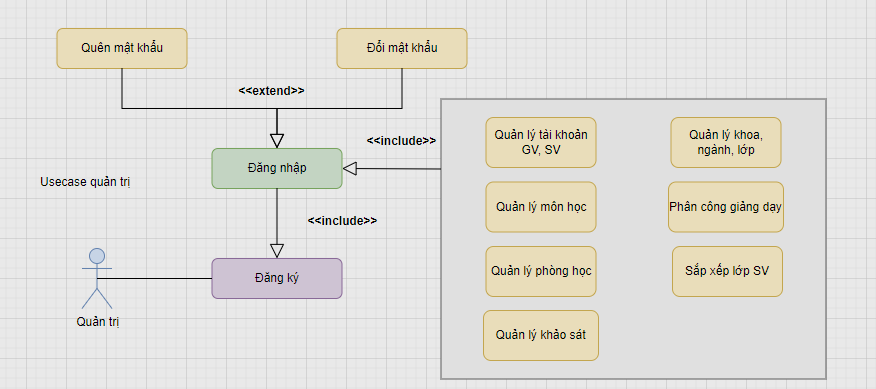
**XÂY DỰNG ỨNG DỤNG QUẢN LÝ HOẠT ĐỘNG DẠY HỌC**

1. **Sơ đồ nghiệp vụ (Business Diagram) và chức năng (Usecase Diagram):**
2. **Business Diagram**

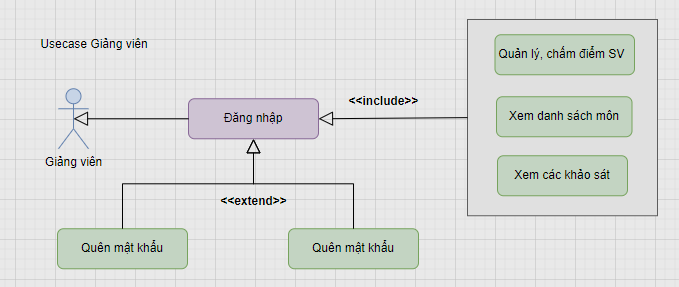
****

*Hình ảnh 1.1. Business Diagram*

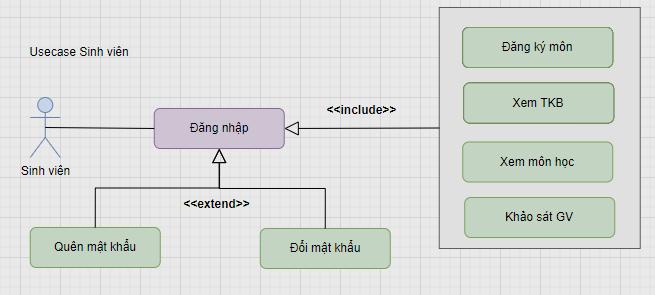
1. **Usecase Diagram**

****

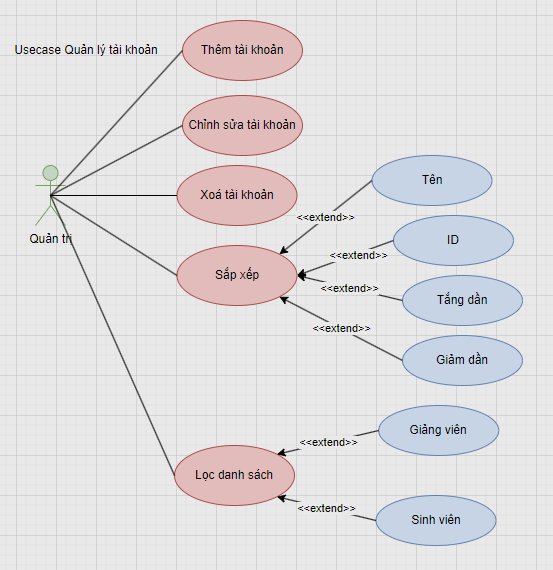
*Hình ành 2.1. Usecase Quản trị*

**

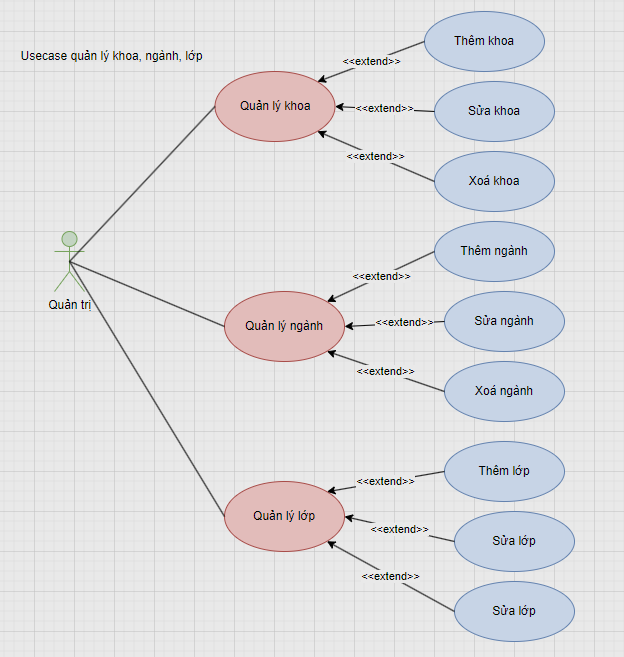
*Hình ảnh 2.2. Usecase Giảng viên*

**

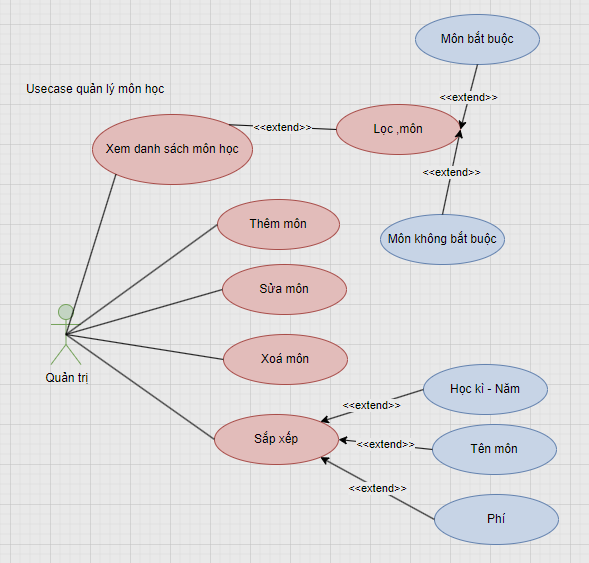
*Hình ảnh 2.3. Usecase Sinh viên*

**

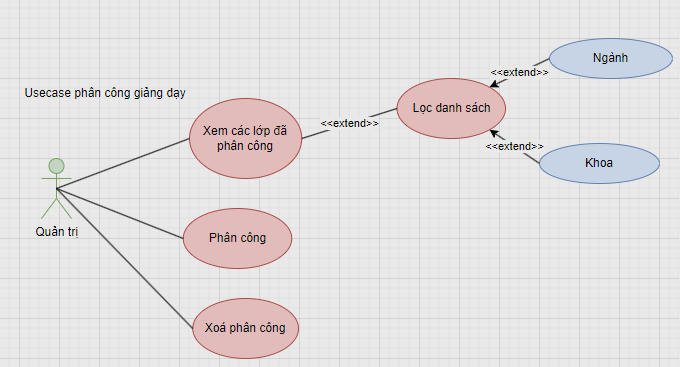
*Hình ảnh 2.4. Usecase Quản lý tài khoản*

****

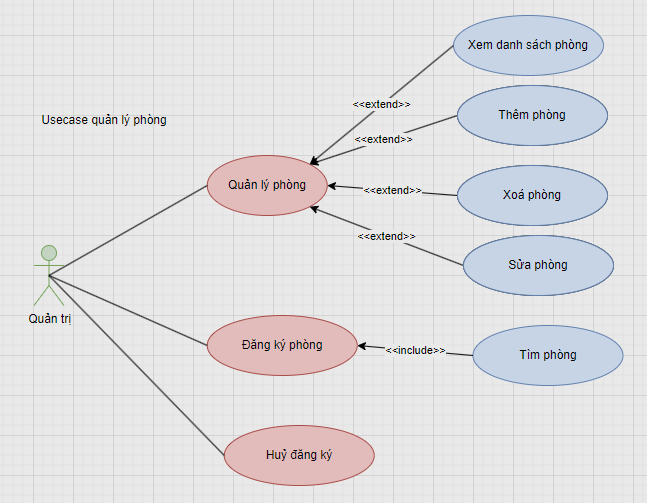
*Hình ảnh 2.5. Usecase Quản lý khoa, ngành, lớp*

**

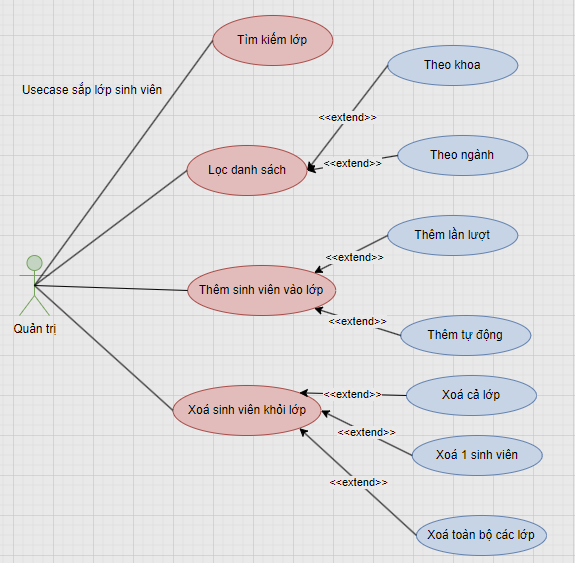
*Hình ảnh 2.6. Usecase Quản lý môn học*

**

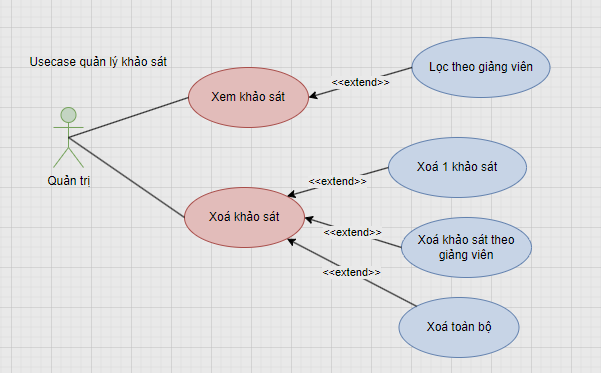
*Hình ảnh 2.7. Usecase Phân công giảng dạy*

**

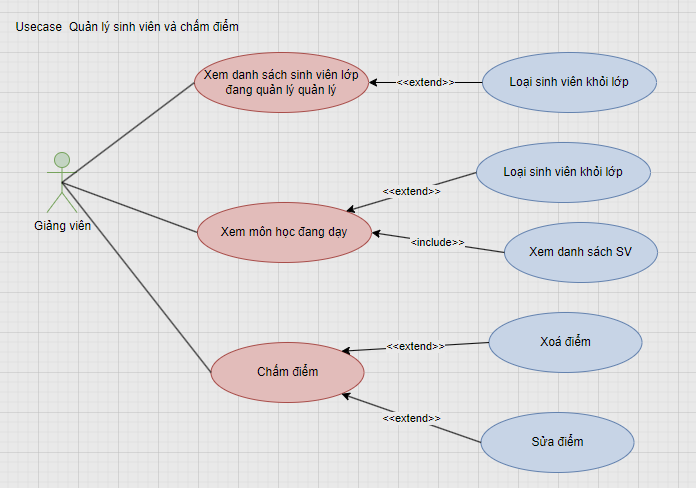
*Hình ảnh 2.8. Usecase Quản lý phòng*

**

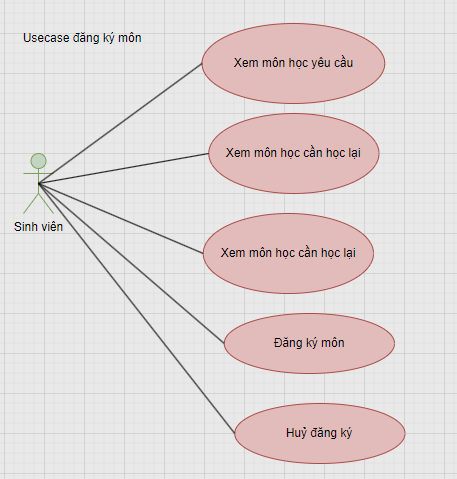
*Hình ảnh 2.9. Usecase Sắp lớp sinh viên*

**

*Hình ảnh 2.10. Usecase Quản lý khảo sát*

**

*Hình 2.11. Usecase Quản lý sinh viên và chấm điểm*

**

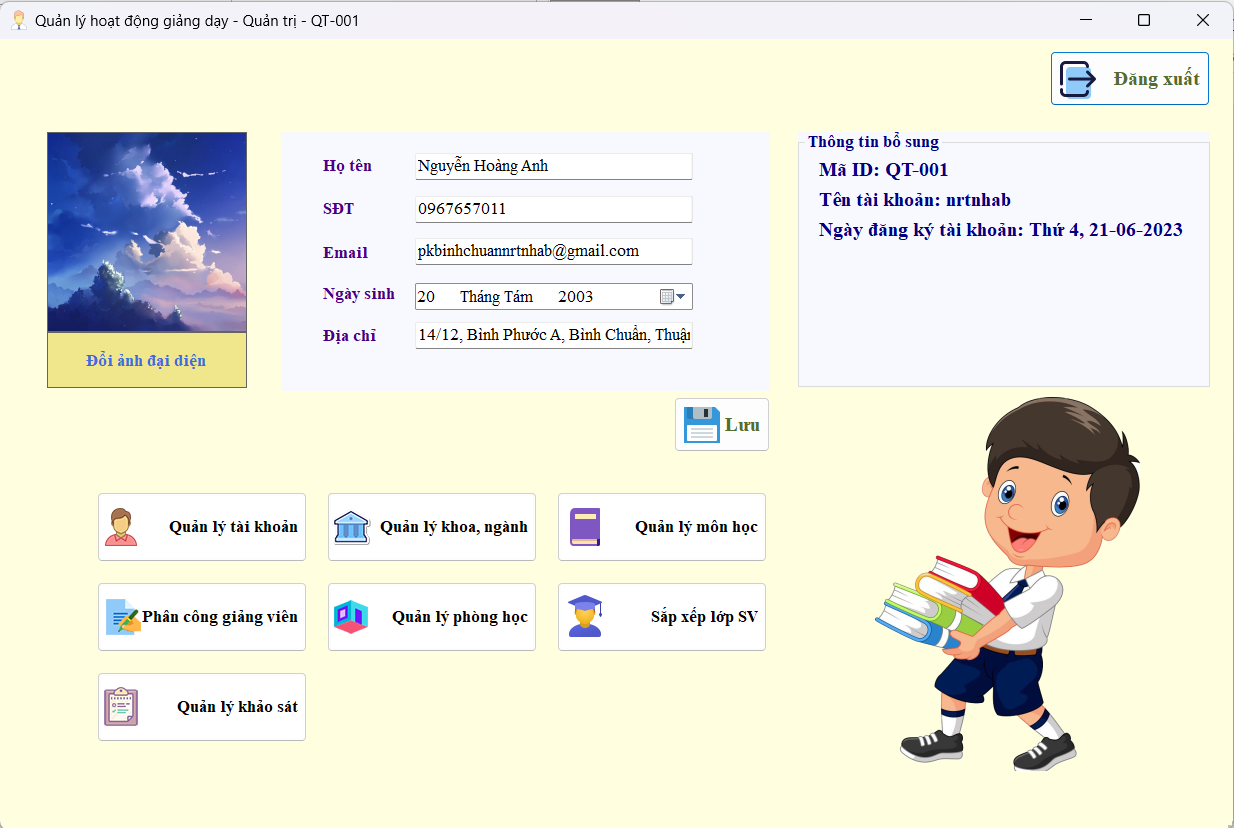
*Hình ảnh 2.12. Usecase Đăng ký*

1. **Khối lượng công việc (đã thực hiện):**

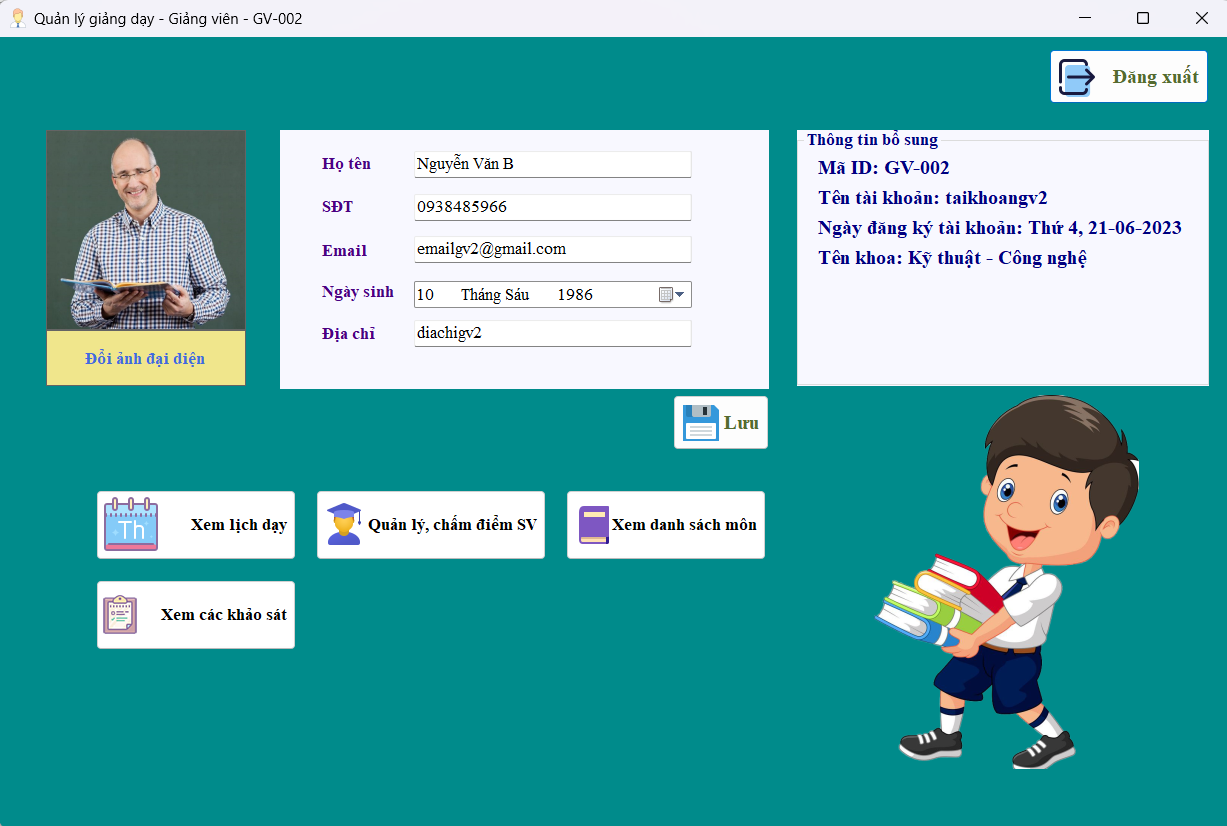
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TT** | **Nội dung** | **Công cụ** | **Sản phẩm** | **Người thực hiện** |
| **1** | **Giao diện đăng nhập, đăng ký, quên mật khẩu, đổi mật khẩu** | **Visual Studio** | **Hoàn thành** | **Nguyễn Minh Hiếu** |
| **2** | **Giao diện chính quản lý hoạt động dạy học** | **Visual Studio** | **Hoàn thành** | **Lê Hồng Lộc, Nguyễn Minh Hiếu** |
| **3** | **Quản lý tài khoản giảng viên và sinh viên** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh, Lê Hồng Lộc, Nguyễn Minh Hiếu** |
| **4** | **Quản lý khoa, ngành, thêm lớp** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **5** | **Quản lý môn học** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **6** | **Phân công giảng dạy** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh, Nguyễn Minh Hiếu** |
| **7** | **Quản lý phòng học** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh, Nguyễn Minh Hiếu, Lê Hồng Lộc** |
| **8** | **Sắp lớp sinh viên vào lớp** | **Visual Studio** | **Hoàn thành** | **Nguyễn Minh Hiếu, Lê Hồng Lộc** |
| **9** | **Quản lý khảo sát** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **10** | **Xem lịch dạy học** | **Visual Studio** | **Hoàn thành** | **Nguyễn Minh Hiếu** |
| **11** | **Quản lý sinh viên và chấm điểm** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh, Nguyễn Minh Hiếu, Lê Hồng Lộc** |
| **12** | **Xem các môn học thuộc khoa (Giảng viên)** | **Visual Studio** | **Hoàn thành** | **Lê Hồng Lộc** |
| **13** | **Xem các khảo sát (Giảng viên)** | **Visual Studio** | **Hoàn thành** | **Lê Hồng Lộc** |
| **14** | **Đăng ký môn** | **Visual Studio** | **Hoàn thành** | **Lê Hồng Lộc, Nguyễn Hoàng Anh** |
| **15** | **Xem thời khóa biểu** | **Visual Studio** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **16** | **Xem môn học thuộc ngành** | **Visual Studio** | **Hoàn thành** | **Nguyễn Minh Hiếu** |
| **17** | **Khảo sát giảng viên** | **Visual Studio** | **Hoàn thành** | **Lê Hồng Lộc** |
| **18** | **Vẽ Diagram** | **Diagram.io** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **19** | **Thiết kế SQL** | **Microsoft Sql Management 8** | **Hoàn thành** | **Nguyễn Hoàng Anh** |
| **20** | **Nạp dữ liệu** | **Trực tiếp trên Winforms** | **Đang trong tiến trình** | **Nguyễn Minh Hiếu, Lê Hồng Lộc** |

1. **Màn hình chính (Form Main):**
2. **Giao diện:**

* **Giao diện cho quản trị:**



* **Giao diện cho giảng viên:**



* **Giao diện cho sinh viên:**

****

1. **Các sự kiện – Phương thức Event Handlers :**

* **Click chọn đổi ảnh đại diện**

private void selectChangeAvt(object sender, MouseEventArgs e)

{

OpenFileDialog openDialog = new OpenFileDialog()

{

Multiselect = false,

Filter = "Tệp ảnh (\*.png, \*.jpg, \*jpeg)|\*.png;\*.jpg;\*.jpeg",

Title = "Chọn ảnh đại diện của bạn (tốt nhất là ảnh có chiều dài bằng chiều rộng)",

};

if (openDialog.ShowDialog() == DialogResult.OK)

{

newPathAvatar = openDialog.FileName;

ptrAvt.Image = Image.FromFile(newPathAvatar);

ptrAvt.SizeMode = PictureBoxSizeMode.StretchImage;

}

}

* **Sự kiện chạm, rời chuột khỏi control:**

private void mouseHover(object sender, EventArgs e)

{

Control control = (Control)sender;

control.BackColor = Color.Yellow;

}

private void mouseLeave(object sender, EventArgs e)

{

Control control = (Control)sender;

control.BackColor = Color.Khaki;

}

* **Sự kiện đóng form:**

private void frmManageSchool\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!quickClose)

{

if (MessageInfo.makeMessage("Question", "Thắc mắc", "Bạn có chắc muốn thoát?") != DialogResult.Yes)

{

e.Cancel = true;

}

}

}

* **Click cập nhật thông tin vào SQL:**

private void selectSaveInfo(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Account account = new Account(nameAct);

info.updateFullName(textInfos[0]);

info.updateBirth(textInfos[1]);

info.updatePhone(textInfos[2]);

info.updateEmail(textInfos[3]);

info.updateAddress(textInfos[4]);

if (info is Teacher)

{

Teacher teacher = (Teacher)info;

teacher.updateTeacher();

}

else if (info is Student)

{

Student student = (Student)info;

student.updateStudent();

}

else

{

info.updatePerson();

}

if (!string.IsNullOrEmpty(newPathAvatar))

{

account.updateAvatar(newPathAvatar);

}

}

* **Các sự kiện click nút chức năng:**

private void selectManageAcct(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmManageAccount frmAcct = new frmManageAccount(info.getID());

frmAcct.ShowDialog();

Show();

}

private void selectManageKhoa(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmManageKhoa frm = new frmManageKhoa();

frm.ShowDialog();

Show();

}

private void selectAssignmentTeacher(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmAssignmentTeacher frm = new frmAssignmentTeacher();

frm.ShowDialog();

Show();

}

private void selectManageClassroom(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmManageClassroom frm = new frmManageClassroom();

frm.ShowDialog();

Show();

}

private void selectManageSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmManageSubject frm = new frmManageSubject();

frm.ShowDialog();

Show();

}

private void selectAssignmentStudent(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmAssignmentStudent frm = new frmAssignmentStudent();

frm.ShowDialog();

Show();

}

private void selectManageStudent(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmManageStudent frm = new frmManageStudent(nameAct);

frm.ShowDialog();

Show();

}

private void selectManageSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

Hide();

frmManageSurvey frm = new frmManageSurvey();

frm.ShowDialog();

Show();

}

private void selectWatchSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmWatchSurvey frm;

if (typeAcct == Account.typeGV)

{

frm = new frmWatchSurvey(nameAct);

}

else

{

frm = new frmWatchSurvey();

}

frm.ShowDialog();

Show();

}

private void selectEnrollSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmEnrollSubject frm = new frmEnrollSubject(nameAct);

frm.ShowDialog();

Show();

}

private void selectWatchSubject(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmWatchSubject frm = new frmWatchSubject(nameAct);

frm.ShowDialog();

Show();

}

private void selectSurvey(object sender, MouseEventArgs e)

{

if (e.Button != MouseButtons.Left)

{

return;

}

clearSelectItem();

Hide();

frmSurvey frm = new frmSurvey(nameAct);

frm.ShowDialog();

Show();

}

* **Click đăng xuất**

private void btnLogOut\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn đăng xuất không?") != DialogResult.Yes)

{

return;

}

quickClose = true;

Hide();

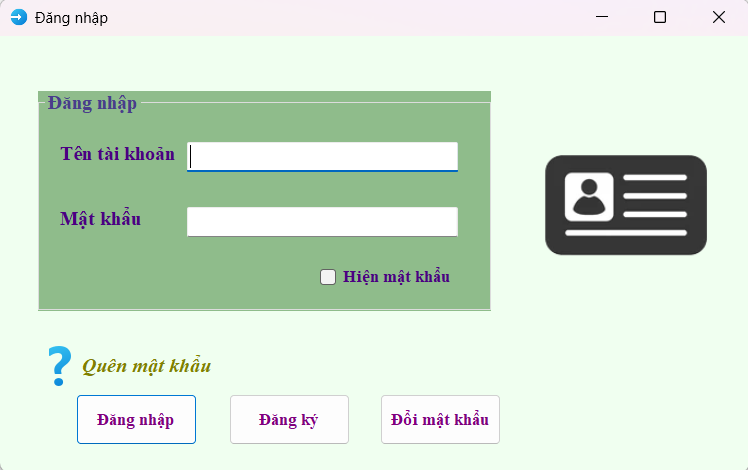
frmSignIn frm = new frmSignIn();

frm.ShowDialog();

Close();

}

1. **Form Đăng nhập:**
2. **Giao diện**

****

1. **Các sự kiện – Phương thức Event Handlers :**

* **Click đăng nhập:**

private void btnSignIn\_Click(object sender, EventArgs e)

{

foreach (Control control in gbSignIn.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin!!!");

control.Focus();

return;

}

}

}

if (!Account.isExistsName(txtNameAcct.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên tài khoản không tồn tại!");

txtNameAcct.Focus();

return;

}

if (!Account.isRightPassword(txtNameAcct.Text, txtPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu không đúng!");

txtNameAcct.Focus();

return;

}

Hide();

Person info = Account.getInfoByName(txtNameAcct.Text);

string typeAccount;

if (info is Student)

{

typeAccount = Account.typeSV;

}

else if (info is Teacher)

{

typeAccount = Account.typeGV;

}

else

{

typeAccount = Account.typeQT;

}

frmManageSchool frm = new frmManageSchool(typeAccount, txtNameAcct.Text );

frm.ShowDialog();

Close();

}

* **Click chuyển sang form đăng ký, đổi mật khẩu, quên mật khẩu:**

private void btnSignUp\_Click(object sender, EventArgs e)

{

Hide();

frmSignUp frm = new frmSignUp();

frm.ShowDialog();

Show();

}

private void btnChangePassword\_Click(object sender, EventArgs e)

{

Hide();

frmChangePassword frm = new frmChangePassword();

frm.ShowDialog();

Show();

}

* **CheckedChanged hiện mật khẩu**

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

* **Chạm và rời chuột khỏi control:**

private void lblForgotPassword\_MouseHover(object sender, EventArgs e)

{

lblForgotPassword.ForeColor = Color.Navy;

}

private void lblForgotPassword\_MouseLeave(object sender, EventArgs e)

{

lblForgotPassword.ForeColor = Color.Olive;

}

* **Click quên mật khẩu:**

private void lblForgotPassword\_Click(object sender, EventArgs e)

{

Hide();

frmForgotPassword frm = new frmForgotPassword();

frm.ShowDialog();

Show();

}

1. **Form Đăng ký**
2. **Giao diện**

****

1. **Các sự kiện – Phương thức Event Handlers :**

* **CheckedChanged hiện mật khẩu:**

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtRePassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

* **Click đăng ký:**

private void btnSignUp\_Click(object sender, EventArgs e)

{

foreach (Control control in gbInfoAccount.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin tài khoản!");

control.Focus();

return;

}

}

}

foreach (Control control in gbInfoPerson.Controls)

{

if (control is TextBox)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đầy đủ thông tin cá nhân!");

control.Focus();

return;

}

}

}

if (txtPassword.Text != txtRePassword.Text)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu nhập lại không khớp!");

txtRePassword.Focus();

return;

}

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không khả dụng!");

txtRePassword.Focus();

return;

}

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không khả dụng!");

txtRePassword.Focus();

return;

}

Person info = new Person(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value));

if (Account.signUp(txtNameAcct.Text, txtPassword.Text, info))

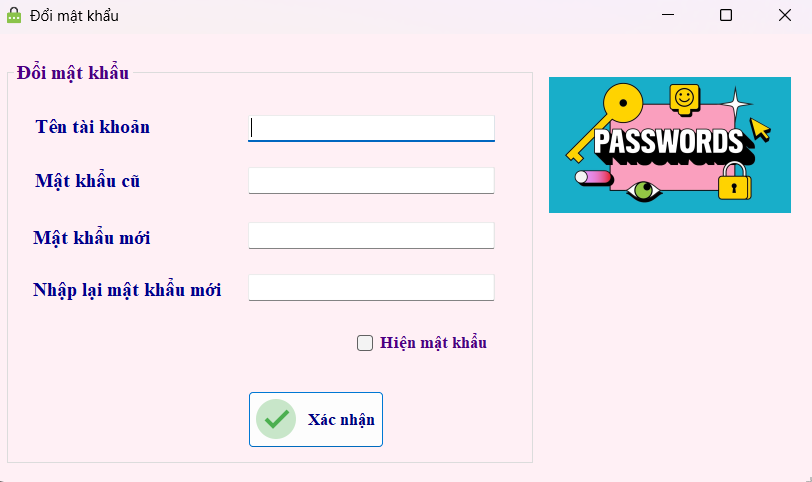
{

Close();

}

}

1. **Form Đổi mật khẩu:**
2. **Giao diện**

****

1. **Các sự kiện – Phương thức Event Handlers :**

* **Keypress nhập tên tài khoản:**

private void txt\_keyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != (char)Keys.Enter)

{

if (e.KeyChar == ' ' || char.IsSymbol(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30))

{

e.Handled = true;

}

}

}

* **Click xác nhận đổi mật khẩu:**

private void btnAccept\_Click(object sender, EventArgs e)

{

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox && string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

control.Focus();

return;

}

}

if (!Account.isPasswordValid(txtNewPassword.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu mới không hợp lệ!");

txtNewPassword.Focus();

return;

}

if (txtNewPassword.Text != txtReNewPassword.Text)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu nhập lại không khớp!");

txtReNewPassword.Focus();

return;

}

if (!Account.isExistsName(txtNameAcct.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Tên tài khoản không tồn tại!");

txtNameAcct.Focus();

return;

}

Account acct = new Account(txtNameAcct.Text);

if (acct.getPassword() != txtOldPassword.Text)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Mật khẩu không chính xác!");

txtOldPassword.Focus();

return;

}

if (Account.updateField(acct.getID(), "TenTK", txtNewPassword.Text))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đổi mật khẩu thành công!");

Close();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đổi mật khẩu thất bại!");

}

}

* **Click hiện mật khẩu:**

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtOldPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtReNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

1. **Form quên mật khẩu:**
2. **Giao diện**

****

1. **Các sự kiện – Phương thức Event Handlers :**

* **Click xác nhận đổi lại mật khẩu mới:**

private void selectAccept(object sender, EventArgs e)

{

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox && string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

control.Focus();

return;

}

}

if (!Account.isPasswordValid(txtNewPass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mật khẩu phải tối thiểu 8 kí tự, bao gồm chữ cái in hoa, in thường, số và kí tự đặc biệt!!!");

return;

}

if (txtRePass.Text != txtNewPass.Text)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Mật khẩu nhập lại không trùng khớp!");

}

try

{

string nameAcct = (string)Person.getFieldPerson("TenTK", "SDT", txtPhone.Text);

Account acct = new Account(nameAcct);

Account.updateField(acct.getID(), "MatKhau", txtNewPass.Text);

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã đổi mật khẩu thành công!!");

Close();

}

catch

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Không tồn tại tài khoản có số điện thoại đã nhập!!");

}

}

* **Click hiện mật khẩu**

private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

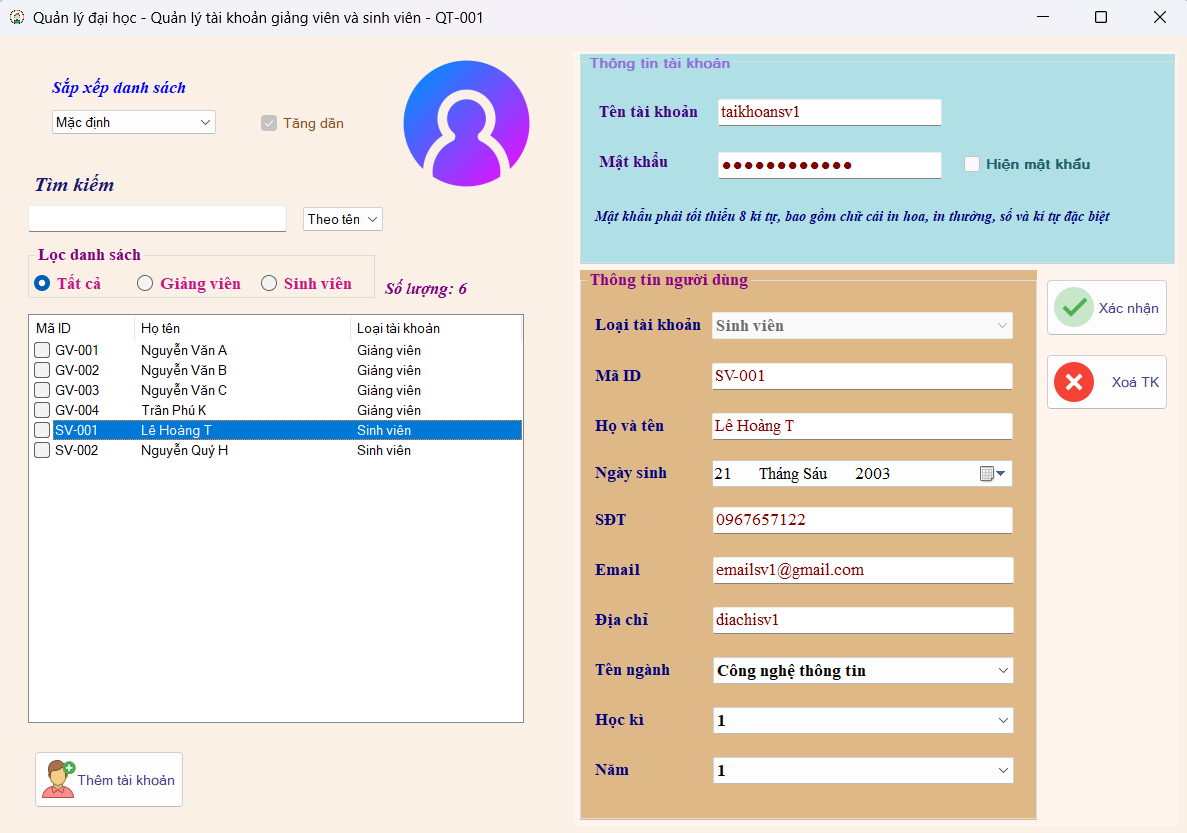
{

txtNewPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

txtRePassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

1. **Form quản lý tài khoản giảng viên và sinh viên:**
2. **Giao diện:**

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **Textchanged Gõ tìm kiếm:**

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

reloadList();

}

* **Click cập nhật hoặc thêm tài khoản:**

private void btnAccept\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtNameAccount.Text) || string.IsNullOrEmpty(txtPassword.Text) || string.IsNullOrEmpty(txtName.Text) || string.IsNullOrEmpty(txtPhone.Text) || string.IsNullOrEmpty(txtEmail.Text) || string.IsNullOrEmpty(txtAddress.Text) || txtID.Text[txtID.Text.Length - 1] == '-')

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (string.IsNullOrEmpty(cbbTypeAcct.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn kiểu tài khoản!");

return;

}

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không hợp lệ!");

return;

}

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không hợp lệ!");

return;

}

if (cbbTypeAcct.SelectedIndex == 0) //giảng viên

{

string tenKhoa = null;

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbTenKhoa")

{

tenKhoa = control.Text;

}

}

if (currentAccount == null)

{

Teacher newTeacher = new Teacher(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), tenKhoa);

Account.signUp(txtNameAccount.Text, txtPassword.Text, newTeacher);

currentAccount = new Account(txtNameAccount.Text);

}

else

{

Account acct = new Account(txtNameAccount.Text, txtPassword.Text);

Teacher newTeacher = new Teacher(currentAccount.getID(), txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), tenKhoa);

acct.updateAccount(newTeacher, txtID.Text);

currentAccount = acct;

}

}

else

{

int year = 0;

string nameMajor = null;

int hocKi = 1;

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbNameMajor")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn tên ngành học!");

return;

}

nameMajor = control.Text;

}

else if (control.Name == "cbbYear")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn năm của sinh viên!");

return;

}

year = int.Parse(control.Text);

}

else if (control.Name == "cbbHocKi")

{

if (string.IsNullOrEmpty(control.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn học kì của sinh viên!");

return;

}

hocKi = int.Parse(control.Text);

}

}

if (currentAccount == null)

{

Student newStudent = new Student(txtID.Text, txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), hocKi, year, nameMajor);

Account.signUp(txtNameAccount.Text, txtPassword.Text, newStudent);

currentAccount = new Account(txtNameAccount.Text);

}

else

{

Account acct = new Account(txtNameAccount.Text, txtPassword.Text);

Student newStudent = new Student(currentAccount.getID(), txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value), hocKi, year, nameMajor);

acct.updateAccount(newStudent, txtID.Text);

currentAccount = acct;

}

}

reloadList();

}

* **Chọn item danh sách tài khoản:**

private void lvInfo\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvInfo.SelectedIndices.Count == 0)

{

currentAccount = null;

pnInfo.Enabled = false;

btnDelete.Enabled = false;

return;

}

btnDelete.Enabled = true;

int index = lvInfo.SelectedIndices[0];

ListViewItem lv = lvInfo.Items[index];

string ID = lv.SubItems[0].Text;

Person person = Person.getInfo("ID", ID);

currentAccount = accounts[index];

cbbTypeAcct.SelectedIndex = -1;

if (lv.SubItems[2].Text == "Sinh viên")

{

cbbTypeAcct.SelectedIndex = 1;

}

else

{

cbbTypeAcct.SelectedIndex = 0;

}

txtName.Text = person.getFullName().getFullName();

txtPhone.Text = person.getPhone();

txtEmail.Text = person.getEmail();

txtAddress.Text = person.getAddress();

txtID.Text = person.getID();

txtNameAccount.Text = accounts[index].getName();

txtPassword.Text = accounts[index].getPassword();

dtpBirth.Value = person.getBirth().getDate();

cbbTypeAcct.Enabled = false;

pnInfo.Enabled = true;

ckbShowPassword.Checked = false;

ckbShowPassword.Enabled = false;

}

* **Chọn loại tài khoản (combobox):**

private void cbbTypeAcct\_SelectedIndexChanged(object sender, EventArgs e)

{

defaultGBInfo();

if (cbbTypeAcct.SelectedIndex == -1)

{

return;

}

if (cbbTypeAcct.SelectedIndex == 0)

{

showMoreInfoTeacher();

}

else

{

showMoreInfoStudent();

}

}

* **Chọn tên ngành (combobox):**

private void cbbNameMajor\_SelectedIndexChanged(object sender, EventArgs e)

{

ComboBox cbbNameMajor = (ComboBox)sender;

if (cbbNameMajor.SelectedIndex == -1)

{

return;

}

string nameMajor = cbbNameMajor.Text;

int years = (int)Major.getField("SoNam", "Ten", nameMajor);

List<int> datas = new List<int>();

for (int i = 1; i <= years; i++)

{

datas.Add(i);

}

foreach (Control control in gbInfo.Controls)

{

if (control.Name == "cbbYear")

{

ComboBox cbb = (ComboBox)control;

cbb.DataSource = datas;

return;

}

}

}

* **Check chọn lọc danh sách:**

private void rdoFilter\_CheckedChanged(object sender, EventArgs e)

{

reloadList();

}

* **Click chọn chức năng thêm tài khoản:**

private void btnAdd\_Click(object sender, EventArgs e)

{

defaultGBInfo();

foreach (Control control in gbAccount.Controls)

{

if (control is TextBox)

{

control.Text = "";

}

}

foreach (Control control in gbInfo.Controls)

{

if (control is TextBox)

{

control.Text = "";

}

}

gbInfo.Height = 397;

cbbTypeAcct.SelectedIndex = -1;

cbbTypeAcct.Enabled = true;

currentAccount = null;

pnInfo.Enabled = true;

ckbShowPassword.Enabled = true;

}

* **KeyPress nhập ID tài khoản:**

private void txtID\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

{

if (!string.IsNullOrEmpty(txtID.Text) && txtID.Text[txtID.Text.Length - 1] == '-')

{

e.Handled = true;

}

}

else if (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30))

{

e.Handled = true;

}

}

* **Click xoá tài khoản:**

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắn chắn muốn xoá tài khoản này?") != DialogResult.Yes)

{

return;

}

if (Student.isExistStudentID(currentAccount.getID()))

{

if (Student.deleteStudent(currentAccount.getID()))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Xoá tài khoản thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá tài khoản thất bại!");

}

}

else

{

if (Teacher.deleteTeacher(currentAccount.getID()))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Xoá tài khoản thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá tài khoản thất bại!");

}

}

reloadList();

btnDelete.Enabled = false;

currentAccount = null;

}

* **CheckedChanged sắp xếp tăng dần hay giảm dần:**

private void ckbIncrease\_CheckedChanged(object sender, EventArgs e)

{

reloadList();

}

* **Chọn kiểu sắp xếp (combobox):**

private void cbbSortBy\_SelectedIndexChanged(object sender, EventArgs e)

{

reloadList();

}

* **CheckedChanged hiện mật khẩu:**

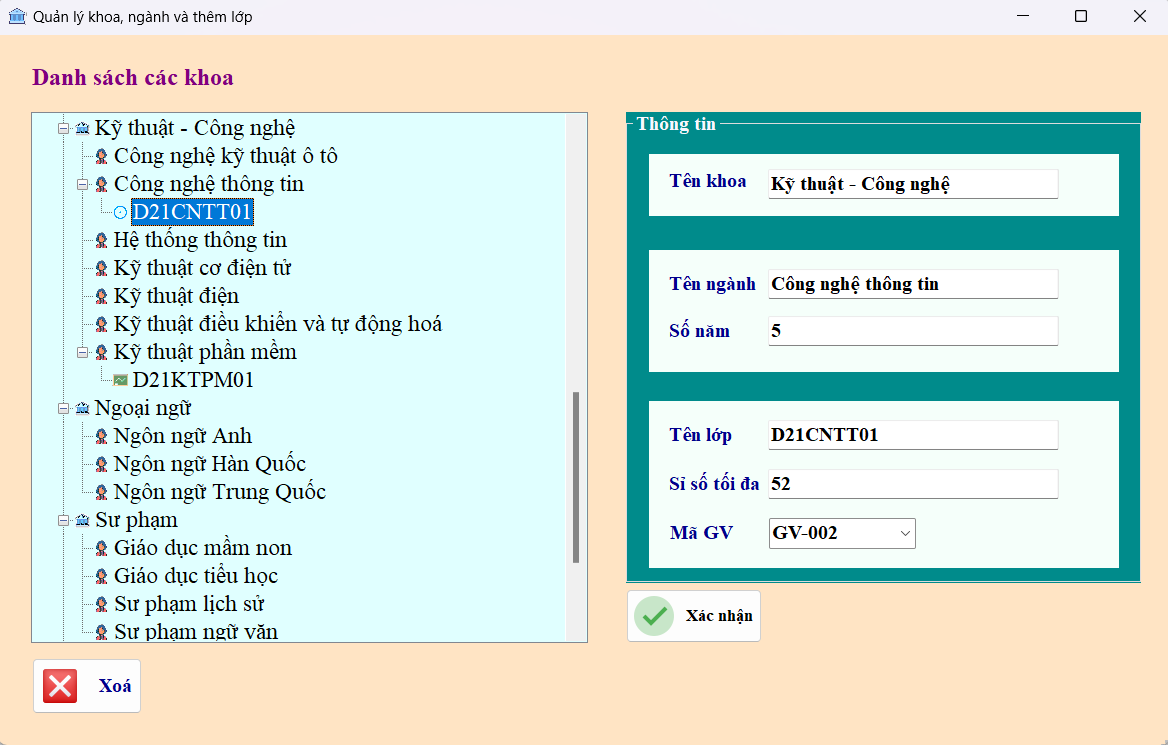
private void ckbShowPassword\_CheckedChanged(object sender, EventArgs e)

{

txtPassword.UseSystemPasswordChar = !ckbShowPassword.Checked;

}

1. **Form quản lý khoa ngành và thêm lớp:**
2. **Giao diện:**



1. **Các sự kiện – Event Handlers (kèm code):**

* **AfterSelect Node (TreeNode):**

private void tvKhoa\_AfterSelect(object sender, TreeViewEventArgs e)

{

currentNode = e.Node;

string tag = (string)currentNode.Tag;

clearText();

gbInfo.Enabled = true;

btnAccept.Enabled = true;

if (tag == "0")

{

btnDelete.Enabled = false;

pnClass.Enabled = false;

return;

}

btnDelete.Enabled = true;

pnClass.Enabled = true;

if (tag == "1")

{

txtNameKhoa.Text = currentNode.Text;

}

else if (tag == "2")

{

txtNameKhoa.Text = currentNode.Parent.Text;

txtNameMajor.Text = currentNode.Text;

Major major = Major.getInfo(txtNameMajor.Text);

txtCountYear.Text = major.getCountYear().ToString();

}

else

{

txtNameKhoa.Text = currentNode.Parent.Parent.Text;

txtNameMajor.Text = currentNode.Parent.Text;

txtNameClass.Text = currentNode.Text;

Khoa khoa = new Khoa(txtNameKhoa.Text);

Major major = Major.getInfo(txtNameMajor.Text);

Class classs = Class.getInfo(txtNameClass.Text);

txtCountYear.Text = major.getCountYear().ToString();

txtCountMax.Text = classs.getMaxCount().ToString();

List<Class> lstClass = Class.getAllClass();

List<Teacher> teachers = khoa.getTeachers();

cbbMaGV.Items.Add(classs.getMaGV());

for (int i = 0; i < teachers.Count; i++)

{

bool isOk = true;

for (int j = 0; j < lstClass.Count; j++)

{

if (teachers[i].getID() == lstClass[j].getMaGV())

{

isOk = false;

break;

}

}

if (isOk)

{

cbbMaGV.Items.Add(teachers[i].getID());

break;

}

}

cbbMaGV.SelectedIndex = 0;

}

}

* **Click xoá:**

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắn chắn muốn xoá không?") != DialogResult.Yes)

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "1")

{

Khoa.deleteKhoa(currentNode.Text);

}

else if (tag == "2")

{

Major.deleteNganh(currentNode.Text);

}

else if (tag == "3")

{

Class.deleteClass(currentNode.Text);

}

tvKhoa.Nodes.Remove(currentNode);

NoSelected();

}

* **Textchange nhập tên khoa:**

private void txtNameKhoa\_TextChanged(object sender, EventArgs e)

{

cbbMaGV.Items.Clear();

if (string.IsNullOrEmpty(txtNameKhoa.Text))

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "0" || !Khoa.isExistKhoa(txtNameKhoa.Text))

{

return;

}

Khoa khoa = new Khoa(txtNameKhoa.Text);

List<Teacher> teachers = khoa.getTeachers();

List<Class> classs = Class.getAllClass();

if (tag == "3")

{

return;

}

for (int j = 0; j < teachers.Count; j++)

{

bool isOK = true;

for (int k = 0; k < classs.Count; k++)

{

if (classs[k].getMaGV() == teachers[j].getID())

{

isOK = false;

break;

}

}

if (isOK)

{

for (int k = 0; k < cbbMaGV.Items.Count; k++)

{

if (cbbMaGV.Items[k].ToString() == teachers[j].getID())

{

isOK = false;

break;

}

}

if (isOK)

{

cbbMaGV.Items.Add(teachers[j].getID());

}

}

}

}

* **Click xác nhận (Lưu, thêm mới):**

private void btnAccept\_Click(object sender, EventArgs e)

{

if (currentNode == null)

{

return;

}

string tag = (string)currentNode.Tag;

if (tag == "0")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa!");

txtNameKhoa.Focus();

return;

}

if (Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

if (!Khoa.addKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

if (Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!Major.addMajor(txtNameMajor.Text, txtNameKhoa.Text, int.Parse(txtCountYear.Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

}

}

}

}

else if (tag == "1")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa!");

txtNameKhoa.Text = currentNode.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !(string.IsNullOrEmpty(txtCountYear.Text)))

{

if (Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!Major.addMajor(txtNameMajor.Text, txtNameKhoa.Text, int.Parse(txtCountYear.Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

else

{

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm lớp vì không đủ thông tin!");

}

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm ngành, lớp vì không đủ thông tin!");

}

}

else if (tag == "2")

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa để sửa!");

txtNameKhoa.Text = currentNode.Parent.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Parent.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Parent.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

Major major = Major.getInfo(currentNode.Text);

if (txtNameMajor.Text != currentNode.Text && Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!major.updateMajor(txtNameMajor.Text, int.Parse(txtCountYear.Text), txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật ngành thất bại!");

txtNameMajor.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

if (!Class.addClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi xảy ra!");

}

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Không thể thêm lớp vì không đủ thông tin!");

}

}

}

else

{

if (string.IsNullOrEmpty(txtNameKhoa.Text)) //sửa khoa

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập tên khoa để sửa!");

txtNameKhoa.Text = currentNode.Parent.Parent.Text;

txtNameKhoa.Focus();

return;

}

if (txtNameKhoa.Text != currentNode.Parent.Parent.Text && Khoa.isExistKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên khoa đã tồn tại!");

txtNameKhoa.Focus();

return;

}

Khoa khoa = new Khoa(currentNode.Parent.Parent.Text);

if (!khoa.updateKhoa(txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật khoa thất bại!");

txtNameKhoa.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameMajor.Text) && !string.IsNullOrEmpty(txtCountYear.Text))

{

Major major = Major.getInfo(currentNode.Parent.Text);

if (txtNameMajor.Text != currentNode.Parent.Text && Major.isExistsMajor(txtNameMajor.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên ngành đã tồn tại!");

txtNameMajor.Focus();

return;

}

if (!major.updateMajor(txtNameMajor.Text, int.Parse(txtCountYear.Text), txtNameKhoa.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật ngành thất bại!");

txtNameMajor.Focus();

return;

}

if (!string.IsNullOrEmpty(txtNameClass.Text) && cbbMaGV.SelectedIndex != -1 && !string.IsNullOrEmpty(txtCountMax.Text))

{

if (txtNameClass.Text != currentNode.Text && Class.isExistsClass(txtNameClass.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tên lớp đã tồn tại!");

txtNameClass.Focus();

return;

}

Class classs = Class.getInfo(currentNode.Text);

if (!classs.updateClass(txtNameClass.Text, cbbMaGV.Text, int.Parse(txtCountMax.Text), txtNameMajor.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Cập nhật lớp thất bại!");

}

}

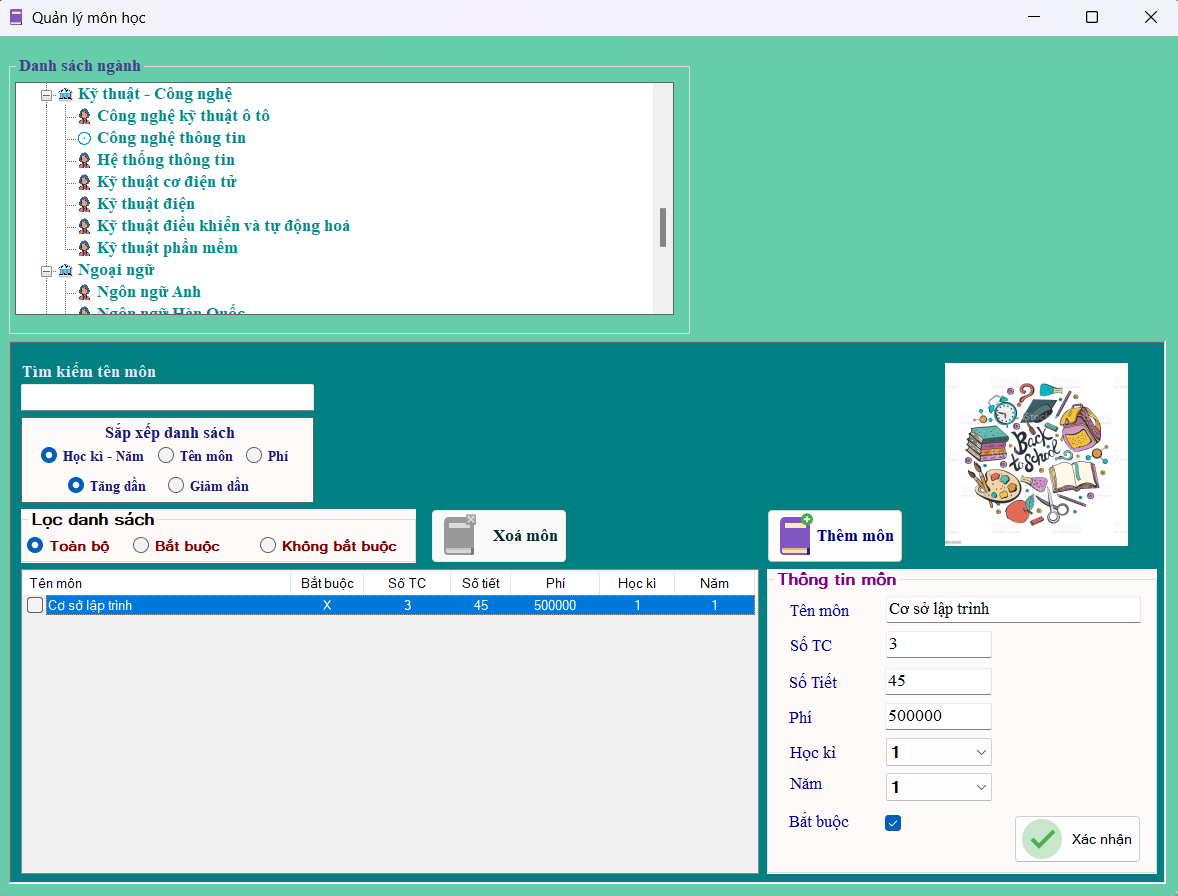
}

}

uploadInfo();

}

1. **Form quản lý môn học:**
2. **Giao diện:**



1. **Các sự kiện – Event Handlers (kèm code):**

* **Click chọn khoa, ngành:**

private void tvKhoa\_AfterSelect(object sender, TreeViewEventArgs e)

{

string tag = (string)e.Node.Tag;

if (tag == "2")

{

currentKhoa = new Khoa(e.Node.Parent.Text);

currentMajor = Major.getInfo(e.Node.Text);

loadSubject();

pnSubject.Enabled = true;

}

else

{

currentMajor = null;

if (tag == "1")

{

currentKhoa = new Khoa(e.Node.Parent.Text);

pnSubject.Enabled = true;

loadSubject();

}

else

{

currentKhoa = null;

lvSubject.Items.Clear();

pnSubject.Enabled = false;

}

}

}

* **Click thêm môn học:**

private void btnAdd\_Click(object sender, EventArgs e)

{

lvSubject.SelectedIndices.Clear();

gbInfo.Text = "Thêm môn học";

resetDefault();

gbInfo.Enabled = true;

}

* **KeyPress nhập phí môn học:**

private void txtMoney\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != ',' && (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30)))

{

e.Handled = true;

}

}

* **Textchanged tìm kiếm môn học theo tên:**

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

findAndFilter();

search();

findAndSort();

uploadList();

}

* **Radio CheckedChanged kiểu sắp xếp:**

private void radio\_CheckedTypeSort(object sender, EventArgs e)

{

if (rdoInscrease.Checked)

{

currentInscrease = true;

}

else

{

currentInscrease = false;

}

findAndFilter();

search();

findAndSort();

uploadList();

}

* **Click xoá môn học:**

private void btnDelete\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn xoá các môn đã check không?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvSubject.CheckedIndices.Count; i++)

{

subjects[lvSubject.CheckedIndices[i]].deleteSubject();

}

loadSubject();

resetDefault();

}

* **SelectedIndexChanged môn học:**

private void lvSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvSubject.SelectedIndices.Count == 0)

{

gbInfo.Enabled = false;

return;

}

gbInfo.Enabled = true;

int index = lvSubject.SelectedIndices[0];

currentSubject = subjects[index];

txtName.Text = subjects[index].getName();

txtSoTC.Text = subjects[index].getSoTC().ToString();

txtSoTiet.Text = subjects[index].getSoTiet().ToString();

txtMoney.Text = subjects[index].getMoney().ToString();

ckMust.Checked = subjects[index].getMust();

cbbYear.Text = subjects[index].getYear().ToString();

cbbHocKi.Text = subjects[index].getHocKi().ToString();

}

* **ItemChecked môn học:**

private void lvSubject\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvSubject.CheckedIndices.Count != 0)

{

btnDelete.Enabled = true;

}

else

{

btnDelete.Enabled = false;

}

}

* **Click xác nhận (Lưu, thêm mới):**

private void btnAccept\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtName.Text) || string.IsNullOrEmpty(txtSoTC.Text) || string.IsNullOrEmpty(txtMoney.Text) || cbbYear.SelectedIndex == -1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (cbbHocKi.SelectedIndex == -1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn học kì cho môn học!");

return;

}

Subject subject = new Subject(txtName.Text, ckMust.Checked, int.Parse(txtSoTC.Text), int.Parse(txtSoTiet.Text), double.Parse(txtMoney.Text), int.Parse(cbbHocKi.SelectedItem.ToString()), int.Parse(cbbYear.Text));

List<Major> majors;

if (currentMajor == null)

{

majors = currentKhoa.getMyMajor();

}

else

{

majors = new List<Major>() { currentMajor };

}

if (currentSubject == null)

{

if (subject.addSubject(majors))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Thêm môn thành công!");

loadSubject();

resetDefault();

gbInfo.Enabled = true;

txtName.Focus();

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Thêm môn thất bại!");

}

}

else

{

if (currentSubject.updateSubject(subject, majors))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Cập nhật thành công!");

loadSubject();

resetDefault();

}

else

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Cập nhật thất bại");

}

}

}

* **Radio CheckedChanged chọn kiểu lọc:**

private void rdo\_CheckedChangeFilter(object sender, EventArgs e)

{

findAndFilter();

search();

findAndSort();

uploadList();

}

* **Radio CheckedChange chọn kiểu sắp xếp:**

private void rdoSortBy\_CheckedChanged(object sender, EventArgs e)

{

findAndFilter();

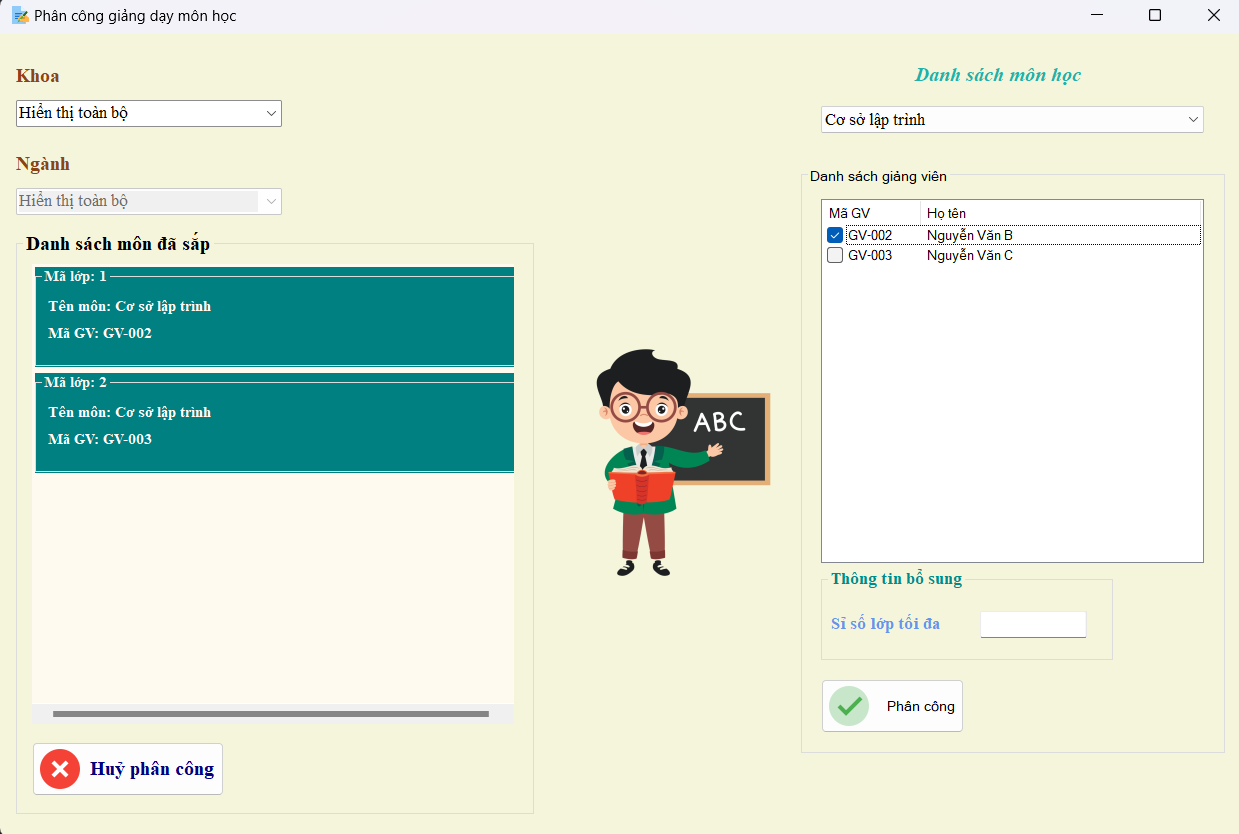
search();

findAndSort();

uploadList();

}

1. **Form phân công giảng dạy môn học:**
2. **Giao diện**



1. **Các sự kiện – Event Handlers (kèm code):**

* **Click chọn môn đã sắp:**

private void selectAssignSubject(object sender, EventArgs e)

{

currentInfoAssign = (InfoAssignTeacher)((Control)sender).Tag;

btnCancel.Enabled = true;

}

* **ItemChecked giảng viên:**

private void lvTeachers\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvTeachers.CheckedIndices.Count != 0)

{

btnTeach.Enabled = true;

}

else

{

btnTeach.Enabled = false;

}

}

* **Click nút phân công:**

private void btnTeach\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtMaxCount.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập sỉ số lớp tối đa!");

txtMaxCount.Focus();

return;

}

for (int i = 0; i < lvTeachers.CheckedIndices.Count; i++)

{

ListViewItem lv = lvTeachers.Items[lvTeachers.CheckedIndices[i]];

InfoAssignTeacher info = new InfoAssignTeacher(currentSubject.getName(), lv.SubItems[0].Text, int.Parse(txtMaxCount.Text));

info.addAssign();

}

MessageInfo.makeMessage("Information", "Chúc mừng", "Phân công thành công!");

loadListStudent();

loadAssignSubjects();

}

* **Click huỷ phân công:**

private void btnCancel\_Click(object sender, EventArgs e)

{

if (currentInfoAssign == null)

{

return;

}

if (currentInfoAssign.deleteAssign())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Xoá phân công thành công!");

loadListStudent();

loadAssignSubjects();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá phân công thất bại!");

}

}

* **SelectedIndexChanged combobox môn học:**

private void cbbSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

currentSubject = Subject.getInfo("Ten", cbbSubject.Text);

loadListStudent();

gbTeacher.Enabled = true;

}

* **SelectedIndexChanged combobox khoa:**

private void cbbKhoa\_SelectedIndexChanged(object sender, EventArgs e)

{

List<Major> majors;

if (cbbKhoa.SelectedIndex == 0)

{

cbbMajor.Enabled = false;

majors = Major.getAllMajor();

}

else

{

cbbMajor.Enabled = true;

Khoa khoa = new Khoa(cbbKhoa.Items[cbbKhoa.SelectedIndex].ToString());

majors = khoa.getMyMajor();

}

cbbMajor.Items.Clear();

cbbMajor.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i< majors.Count; i++)

{

cbbMajor.Items.Add(majors[i].getName());

}

cbbMajor.SelectedIndex = 0;

}

* **SelectedIndexChanged combobox ngành:**

private void cbbMajor\_SelectedIndexChanged(object sender, EventArgs e)

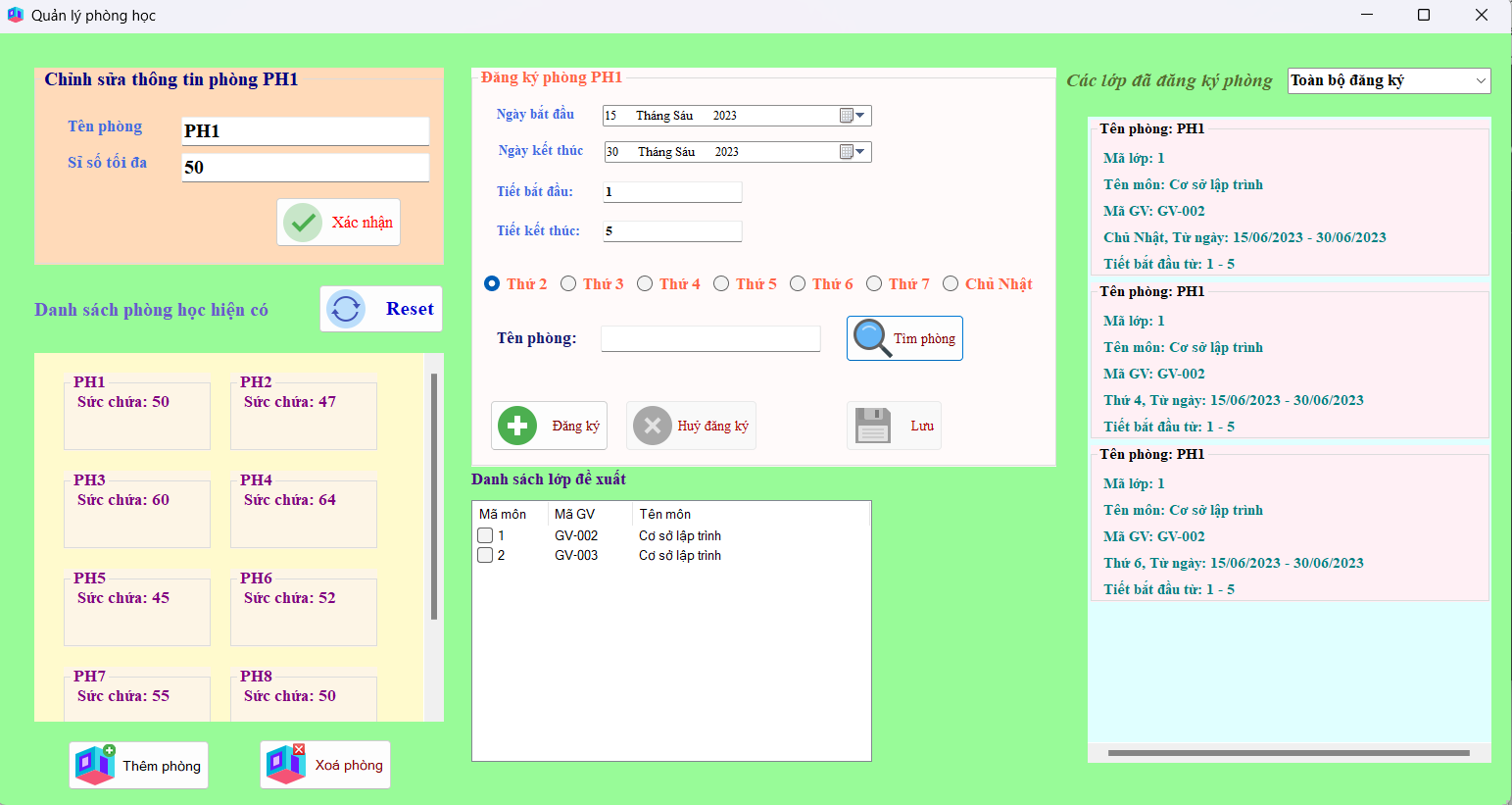
{

loadAssignSubjects();

loadSubjects();

}

1. **Form quản lý phòng học:**
2. **Giao diện**



1. **Các sự kiện – Event Handlers (kèm code):**

* **Click môn học đề xuất:**

private void selectSubjectAssign(object sender, EventArgs e)

{

GroupBox gb = (GroupBox)sender;

InfoAssignRoom infoAssignRoom = (InfoAssignRoom)gb.Tag;

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignRoom.getMaLop());

Subject sub = Subject.getInfo("Ten", infoAssignTeacher.getNameSubject());

loadSubject();

showInfoSubject(sub);

dtpDateStart.Value = infoAssignRoom.getDateStart().getDate();

dtpDateEnd.Value = infoAssignRoom.getDateEnd().getDate();

txtTietStart.Text = infoAssignRoom.getTietStart().ToString();

txtTietEnd.Text = infoAssignRoom.getTietEnd().ToString();

currentAssign = infoAssignRoom;

gbSubject.Enabled = true;

lvClassSubject.Enabled = true;

for (int i = 0; i < lvClassSubject.Items.Count; i++)

{

if (sub.getName() == lvClassSubject.Items[i].SubItems[2].Text && infoAssignRoom.getMaLop() == int.Parse(lvClassSubject.Items[i].SubItems[0].Text) && infoAssignTeacher.getMaGV() == lvClassSubject.Items[i].SubItems[1].Text)

{

lvClassSubject.Items[i].Checked = true;

break;

}

}

btnAssignSubject.Enabled = false;

btnCancelSubject.Enabled = true;

btnSave.Enabled = true;

gbSubject.Text = "Chỉnh sửa thông tin đăng ký";

}

* **SelectedIndexChanged combobox mã lớp đã đăng ký phòng hiện tại:**

private void cbbAssignRoom\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbAssignRoom.SelectedIndex == -1)

{

cbbAssignRoom.SelectedIndex = 0;

return;

}

if (cbbAssignRoom.SelectedIndex == 0)

{

loadSubjectAssignRoom();

}

else

{

gbSubject.Text = "Đăng ký phòng " + currentRoom.getName();

currentAssign = null;

fpnClassAssign.Controls.Clear();

btnSearch.Enabled = true;

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

string item = cbbAssignRoom.Items[cbbAssignRoom.SelectedIndex].ToString();

int maLop = int.Parse(item.Substring(item.LastIndexOf(' ')));

List<InfoAssignRoom> assignRooms = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < assignRooms.Count; i++)

{

if (assignRooms[i].getMaLop() == maLop)

{

fpnClassAssign.Controls.Add(getGroupSubjectAssign(assignRooms[i]));

}

}

}

}

* **Click chọn phòng:**

private void selectRoom(object sender, EventArgs e)

{

cbbAssignRoom.Enabled = true;

currentRoom = (Classroom)((Control)sender).Tag;

txtName.Text = currentRoom.getName();

txtCountMax.Text = currentRoom.getMaxCount().ToString();

gbInfo.Enabled = true;

lvClassSubject.Enabled = true;

gbSubject.Enabled = true;

btnDelete.Enabled = true;

gbInfo.Text = "Chỉnh sửa thông tin phòng " + currentRoom.getName();

cbbAssignRoom.Items.Clear();

cbbAssignRoom.Items.Add("Toàn bộ đăng ký");

List<InfoAssignRoom> assignRooms = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < assignRooms.Count; i++)

{

bool isOk = true;

for (int j = 0; j < cbbAssignRoom.Items.Count; j++)

{

if (cbbAssignRoom.Items[j].ToString() == "Mã lớp " + assignRooms[i].getMaLop())

{

isOk = false;

break;

}

}

if (isOk)

{

cbbAssignRoom.Items.Add("Mã lớp " + assignRooms[i].getMaLop());

}

}

cbbAssignRoom.SelectedIndex = 0;

loadSubject();

if (string.IsNullOrEmpty(txtTietStart.Text) || string.IsNullOrEmpty(txtTietEnd.Text))

{

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

}

else

{

btnAssignSubject.Enabled = true;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

}

}

* **Click xác nhận (thêm mới, lưu phòng):**

private void btnAccept\_Click(object sender, EventArgs e)

{

foreach (Control c in gbInfo.Controls)

{

if (c is TextBox)

{

if (string.IsNullOrEmpty(c.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin phòng!");

c.Focus();

return;

}

}

}

if (currentRoom == null)

{

Classroom room = new Classroom(txtName.Text, int.Parse(txtCountMax.Text));

if (room.addNewRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Thêm phòng thành công!");

FilterRooms();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Thêm phòng thất bại!");

}

}

else

{

List<Subject> subs = currentRoom.getSubject();

for (int i = 0; i < subs.Count; i++)

{

List<Teacher> teachers = subs[i].getTeacherTeach();

for (int j = 0; j < teachers.Count; j++)

{

List<InfoAssignTeacher> infoAssigns = InfoAssignTeacher.getAllAssign();

for (int k = 0; k < infoAssigns.Count; k++)

{

List<Student> students = InfoAssignSubject.getStudentsAssignSubject(infoAssigns[k].getMaLop());

if (students.Count > int.Parse(txtCountMax.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Sửa sỉ số tối đa của phòng thất bại!");

currentRoom.updateRoom(txtName.Text, currentRoom.getMaxCount());

txtCountMax.Text = currentRoom.getMaxCount().ToString();

return;

}

}

}

}

if (currentRoom.updateRoom(txtName.Text, int.Parse(txtCountMax.Text)))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã sửa thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Thêm phòng thất bại!");

}

FilterRooms();

}

}

* **Click huỷ đặt phòng:**

private void btnCancelSubject\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn huỷ đăng ký phòng?") != DialogResult.Yes)

{

return;

}

try

{

if (currentAssign.deleteAssignRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Huỷ đăng ký phòng thành công!");

loadSubject();

loadSubjectAssignRoom();

btnAssignSubject.Enabled = true;

btnCancelSubject.Enabled = false;

btnSave.Enabled = false;

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Huỷ đăng ký phòng thất bại!");

}

}

catch

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Huỷ đăng ký phòng thất bại!");

}

}

* **Click đăng ký phòng:**

private void btnAssignSubject\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtTietStart.Text) || string.IsNullOrEmpty(txtTietEnd.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (lvClassSubject.CheckedIndices.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn lớp để đăng ký phòng!");

return;

}

try

{

Date dateStart = new Date(dtpDateStart.Value);

Date dateEnd = new Date(dtpDateEnd.Value);

if (Date.compareDate(dateStart, dateEnd) == 1)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Ngày bắt đầu không được lớn hơn ngày kết thúc!");

return;

}

int tietStart = int.Parse(txtTietStart.Text);

int tietEnd = int.Parse(txtTietEnd.Text);

if (tietStart > tietEnd)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Tiết bắt đầu không được lớn nhớ tiết kết thúc!");

return;

}

string dayOfWeek = findDayOfWeek();

ListViewItem li = lvClassSubject.Items[lvClassSubject.CheckedIndices[0]];

int maLop = int.Parse(li.SubItems[0].Text);

List<InfoAssignRoom> infoRoomsByMaLop = Classroom.getAssignRoom(maLop);

for (int i = 0; i < infoRoomsByMaLop.Count; i++)

{

if (General.isDateAInB(dateStart, dateEnd, tietStart, tietEnd, dayOfWeek, infoRoomsByMaLop[i].getDateStart(), infoRoomsByMaLop[i].getDateEnd(), infoRoomsByMaLop[i].getTietStart(), infoRoomsByMaLop[i].getTietEnd(), infoRoomsByMaLop[i].getStrDayOfWeek()))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mã lớp " + maLop + " đã đăng ký: " + infoRoomsByMaLop[i].getCalender() + "!");

return;

}

}

List<InfoAssignRoom> infoRoomsByNameRoom = Classroom.getAssignRoom(currentRoom.getName());

for (int i = 0; i < infoRoomsByNameRoom.Count; i++)

{

if (General.isDateAInB(dateStart, dateEnd, tietStart, tietEnd, dayOfWeek, infoRoomsByNameRoom[i].getDateStart(), infoRoomsByNameRoom[i].getDateEnd(), infoRoomsByNameRoom[i].getTietStart(), infoRoomsByNameRoom[i].getTietEnd(), infoRoomsByNameRoom[i].getStrDayOfWeek()))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Phòng đã được đăng ký: " + infoRoomsByNameRoom[i].getCalender() + "!");

return;

}

}

int STT = int.Parse(li.SubItems[0].Text);

InfoAssignRoom info = new InfoAssignRoom(currentRoom.getName(), STT, dateStart, dateEnd, tietStart, tietEnd, dayOfWeek);

if (info.addAssignRoom())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đăng ký phòng thành công!");

loadSubject();

loadSubjectAssignRoom();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đăng ký phòng thất bại!");

}

}

catch (Exception ex)

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Đăng ký phòng thất bại! " + ex.Message);

}

}

* **Click xoá phòng học:**

private void btnDelete\_Click(object sender, EventArgs e)

{

if (currentRoom != null)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn xoá phòng " + currentRoom.getName() +"?") != DialogResult.Yes)

{

return;

}

currentRoom.deleteRoom();

currentRoom = null;

btnDelete.Enabled = false;

cbbAssignRoom.Enabled = false;

FilterRooms();

}

}

* **Click lưu thông tin đặt phòng:**

private void btnSave\_Click(object sender, EventArgs e)

{

if (lvClassSubject.CheckedIndices.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa chọn môn để chỉnh sửa!");

return;

}

ListViewItem li = lvClassSubject.Items[lvClassSubject.CheckedIndices[0]];

int STT = int.Parse(li.SubItems[0].Text);

string dayOfWeek = findDayOfWeek();

InfoAssignRoom info = new InfoAssignRoom(currentRoom.getName(), STT, new Date(dtpDateStart.Value), new Date(dtpDateEnd.Value), int.Parse(txtTietStart.Text), int.Parse(txtTietEnd.Text), dayOfWeek);

if (currentAssign.updateAssignRoom(info))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Chỉnh sửa thông tin lớp thành công!");

loadSubjectAssignRoom();

currentAssign = info;

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Chỉnh sửa thông tin lớp thất bại!");

}

}

* **Check chọn danh sách môn đề xuất:**

private void lvSubjects\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvClassSubject.CheckedIndices.Count == 0 && currentAssign == null)

{

return;

}

if (e.Item.Checked == false)

{

return;

}

if (lvClassSubject.CheckedIndices.Count >= 1)

{

for (int i = 0; i < lvClassSubject.CheckedIndices.Count; i++)

{

if (lvClassSubject.Items[lvClassSubject.CheckedIndices[i]] != e.Item)

{

lvClassSubject.Items[lvClassSubject.CheckedIndices[i]].Checked = false;

}

}

}

gbSubject.Enabled = true;

if (currentAssign == null)

{

btnAssignSubject.Enabled = true;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

gbSubject.Text = "Đăng ký phòng " + currentRoom.getName();

}

else

{

btnCancelSubject.Enabled = true;

btnSave.Enabled = true;

btnAssignSubject.Enabled = false;

gbSubject.Text = "Chỉnh sửa thông tin đăng ký phòng " + currentRoom.getName();

}

}

* **Click tìm kiếm phòng:**

private void btnSearch\_Click(object sender, EventArgs e)

{

FilterRooms();

}

* **Load lại danh sách phòng học:**

private void btnReset\_Click(object sender, EventArgs e)

{

resetInputSubject();

btnAssignSubject.Enabled = false;

btnSave.Enabled = false;

btnCancelSubject.Enabled = false;

btnDelete.Enabled = false;

txtName.Clear();

txtCountMax.Clear();

createRooms();

currentRoom = null;

fpnClassAssign.Controls.Clear();

lvClassSubject.Items.Clear();

cbbAssignRoom.Enabled = false;

}

1. **Form phân bổ sinh viên:**
2. **Giao diện:**



1. **Các sự kiện – Event Handlers (kèm code):**

* **SelectedIndexChanged khoa (combobox):**

private void cbbKhoa\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbKhoa.SelectedIndex == -1)

{

fpnClasss.Controls.Clear();

gbStudentInClass.Enabled = false;

gbStudents.Enabled = false;

cbbMajor.Text = "";

cbbMajor.Items.Clear();

cbbMajor.Enabled = false;

return;

}

else

{

if (cbbKhoa.SelectedIndex == 0)

{

cbbMajor.Text = "";

cbbMajor.Items.Clear();

cbbMajor.Enabled = false;

loadClass();

}

else

{

cbbMajor.Enabled = true;

loadMajor();

}

}

}

* **SelectedIndexChanged ngành (combobox):**

private void cbbMajor\_SelectedIndexChanged(object sender, EventArgs e)

{

loadClass();

}

* **Click Chọn lớp học:**

private void selectClass(object sender, EventArgs e)

{

currentClass = (Class)((Control)sender).Tag;

loadListStudentInClass();

loadListStudents();

gbStudentInClass.Enabled = true;

gbStudents.Enabled = true;

gbStudentInClass.Text = "Sinh viên lớp " + currentClass.getName();

}

* **ItemChecked chọn sinh viên tự do:**

private void lvStudents\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvStudents.CheckedIndices.Count != 0)

{

btnAdd.Enabled = true;

}

else

{

btnAdd.Enabled = false;

}

}

* **ItemChecked sinh viên đã trong lớp:**

private void lvStudentInClass\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvStudentInClass.CheckedIndices.Count != 0)

{

btnKick.Enabled = true;

}

else

{

btnKick.Enabled = false;

}

}

* **Click loại sinh viên khỏi lớp:**

private void btnKick\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại sinh viên khỏi lớp " + currentClass.getName() +"?") != DialogResult.Yes)

{

return;

}

for (int i = 0; i < lvStudentInClass.CheckedIndices.Count; i++)

{

ListViewItem li = lvStudentInClass.Items[lvStudentInClass.CheckedIndices[i]];

InfoAssignStudent.kickStudent(li.Text);

}

loadListStudentInClass();

loadListStudents();

}

* **Click thêm sinh viên vào lớp:**

private void btnAdd\_Click(object sender, EventArgs e)

{

for (int i = 0; i < lvStudents.CheckedIndices.Count; i++)

{

ListViewItem li = lvStudents.Items[lvStudents.CheckedIndices[i]];

InfoAssignStudent.addAssignStudent(li.Text, currentClass.getName());

}

loadListStudentInClass();

loadListStudents();

}

* **TextChange tìm kiếm lớp:**

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

classs = Class.getAllClass();

for (int i = 0; i < txtSearch.Text.Length; i++)

{

removeListByCharName(txtSearch.Text[i]);

}

fpnClasss.Controls.Clear();

for (int i = 0; i < classs.Count; i++)

{

fpnClasss.Controls.Add(getGroupClass(classs[i]));

}

}

* **Click sắp lớp tự động cho toàn bộ sinh viên:**

private void btnAssignAll\_Click(object sender, EventArgs e)

{

List<Class> classs = Class.getAllClass();

for (int i = 0; i < classs.Count; i++)

{

Major major = Major.getInfo(classs[i].getNameMajor());

List<Student> studentInMajor = major.getStudents();

for (int j = 0; j < studentInMajor.Count; j++)

{

if (Student.checkHadClass(studentInMajor[j]) || InfoAssignStudent.isExistsAssignStudent(studentInMajor[j].getID()))

{

studentInMajor.RemoveAt(j);

j--;

}

}

while (studentInMajor.Count != 0 && classs[i].getStudents().Count() < classs[i].getMaxCount())

{

if (!InfoAssignStudent.addAssignStudent(studentInMajor[0].getID(), classs[i].getName()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình phân bổ sinh viên!");

break;

}

studentInMajor.RemoveAt(0);

}

}

loadListStudentInClass();

loadListStudents();

}

* **Kick tất cả sinh viên khỏi lớp các lớp:**

private void btnKickAll\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại toàn bộ sinh viên khỏi mọi lớp?") != DialogResult.Yes)

{

return;

}

List<Class> classs = Class.getAllClass();

for (int i = 0; i < classs.Count; i++)

{

List<Student> studentInMajor = classs[i].getStudents();

for (int j = 0; j < studentInMajor.Count; j++)

{

if (!InfoAssignStudent.kickStudent(studentInMajor[j].getID()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình loại bỏ sinh viên!");

break;

}

}

}

loadListStudentInClass();

loadListStudents();

}

* **Kick tất cả sinh viên khỏi lớp đang chọn:**

private void btnKickAllClass\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn các loại toàn bộ sinh viên khỏi lớp " + currentClass.getName() + "?") != DialogResult.Yes)

{

return;

}

List<Student> studentInMajor = currentClass.getStudents();

for (int i = 0; i < studentInMajor.Count; i++)

{

if (!InfoAssignStudent.kickStudent(studentInMajor[i].getID()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Có lỗi trong quá trình loại bỏ sinh viên!");

break;

}

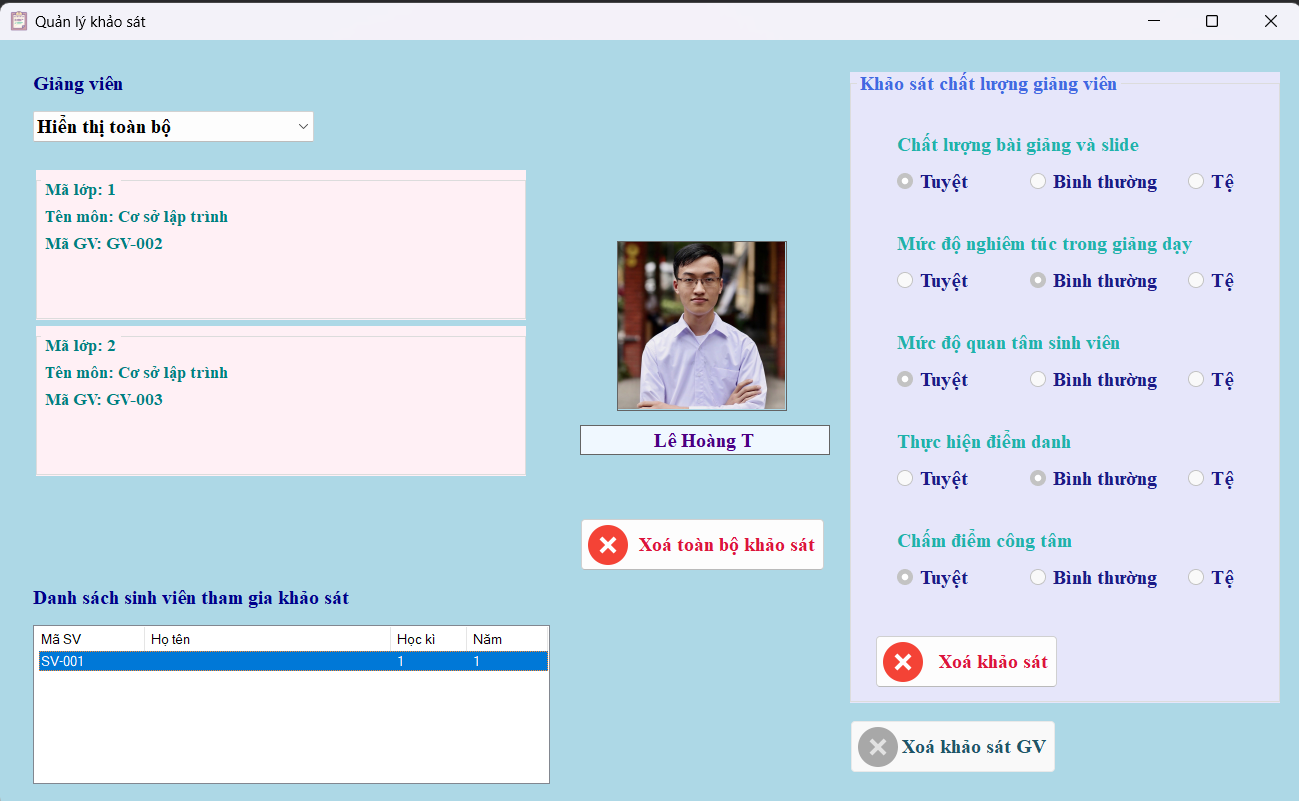
}

loadListStudentInClass();

loadListStudents();

}

1. **Form quản lý khảo sát:**
2. **Giao diện**

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **Click chọn môn học cần khảo sát:**

private void selectInfoAssign(object sender, EventArgs e)

{

currentAssign = (InfoAssignTeacher)((Control)sender).Tag;

lvStudent.Items.Clear();

List<Survey> survey = currentAssign.getAllSurveys();

for (int i = 0; i < survey.Count; i++)

{

ListViewItem li = new ListViewItem()

{

Text = survey[i].getMaSV(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = (string)Student.getFieldStudentByID("Ten", survey[i].getMaSV()) });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = survey[i].getHocKi().ToString()});

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = survey[i].getYear().ToString() });

lvStudent.Items.Add(li);

}

clearSurvey();

}

* **Click xoá khảo sát:**

private void btnDelete\_Click(object sender, EventArgs e)

{

ListViewItem li = lvStudent.Items[lvStudent.SelectedIndices[0]];

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn xoá khảo sát: " + li.SubItems[0].Text + "-" + currentAssign.getMaLop() + "-" + li.SubItems[2].Text + "-" + li.SubItems[3].Text + "?") != DialogResult.Yes)

{

return;

}

if (!Survey.deleteSurvey(li.SubItems[0].Text, currentAssign.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text)))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá khảo sát thất bại!");

}

}

* **SelectedIndexChanged Chọn sinh viên để xem khảo sát:**

private void lvStudent\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvStudent.SelectedIndices.Count != 0)

{

btnDelete.Enabled = true;

gbSurvey.Enabled = true;

ListViewItem li = lvStudent.Items[lvStudent.SelectedIndices[0]];

Survey survey = Survey.getSurvey(li.SubItems[0].Text, currentAssign.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

if (survey == null)

{

return;

}

Student student = (Student)Person.getInfo("ID", survey.getMaSV());

lblNameStudent.Text = student.getFullName().getFullName();

object img = Account.getField("AnhDaiDien", "ID", student.getID());

if (img != null)

{

ptrAvt.Image = Avatar.ConvertBytesToImage((byte[])img);

}

else

{

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

else

{

btnDelete.Enabled = false;

clearSurvey();

}

}

* **SelectedIndexChange combobox mã giảng viên lọc khảo sát**

private void cbbMaGV\_SelectedIndexChanged(object sender, EventArgs e)

{

loadSubjectClass();

}

* **Click xoá toàn bộ khảo sát:**

private void btnDeleteAll\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn muốn xoá toàn bộ khảo sát!") != DialogResult.Yes)

{

return;

}

if (!Survey.deleteAllSurvey())

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá toàn bộ khảo sát thất bại!");

}

else

{

loadSubjectClass();

}

}

* **Xoá toàn bộ khảo sát của giảng viên hiện tại:**

private void btnDeleteAllSurveyGV\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc muốn muốn xoá toàn bộ khảo sát của giảng viên + " + cbbMaGV.Text + "!") != DialogResult.Yes)

{

return;

}

if (!Survey.deleteAllSurvey(currentAssign.getMaGV()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá toàn bộ khảo sát của giảng viên " + currentAssign.getMaGV() + " thất bại!");

}

else

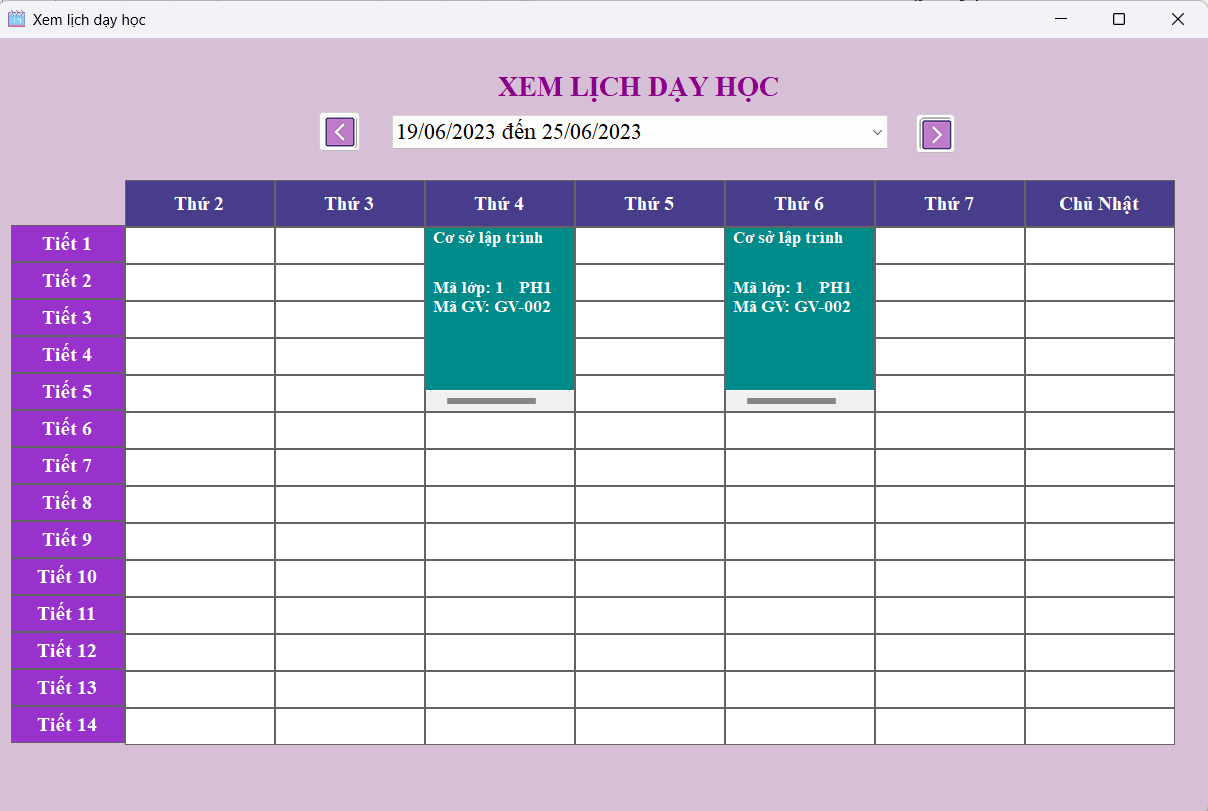
{

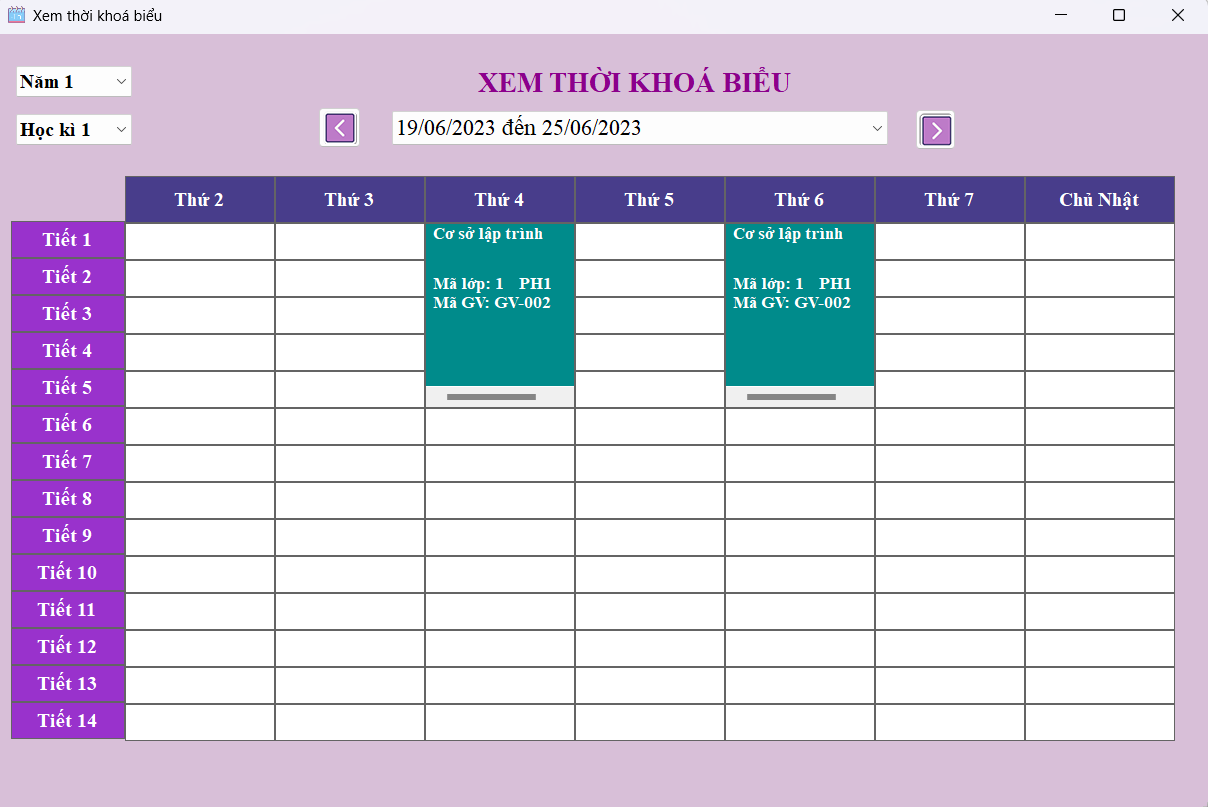
loadSubjectClass();

}

}

1. **Form xem lịch dạy học (Giảng viên) / Xem thời khoá biểu (Sinh viên):**
2. **Giao diện:**

****

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **SelectedIndexChanged Combobox năm (Sinh viên):**

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

for (int i = 1; i <= student.getHocKi(); i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = student.getHocKi() - 1;

}

else

{

for (int i = 1; i <= 2; i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = 0;

}

}

* **SelectedIndexChanged Combobox học kì (Sinh viên):**

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

loadDate();

}

* **Click quay lại ngày trước:**

private void btnPrev\_Click(object sender, EventArgs e)

{

cbbDate.SelectedIndex--;

}

* **Click ngày kế:**

private void btnNext\_Click(object sender, EventArgs e)

{

cbbDate.SelectedIndex++;

}

* **SelectedIndexChanged Combobox ngày:**

private void cbbDate\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbDate.SelectedIndex == -1)

{

return;

}

enableButton();

loadCellCalendar();

string strDateStart = cbbDate.Text.Substring(0, cbbDate.Text.IndexOf(' '));

Date dateStart = new Date(strDateStart, '/');

if (infoAssignSubjects != null)

{

int l = 0;

while (dateStart.getStrDayOfWeek() != "Chủ Nhật")

{

for (int i = 0; i < infoAssignSubjects.Count; i++)

{

List<InfoAssignRoom> infoAssignRooms = InfoAssignRoom.getInfo(infoAssignSubjects[i].getMaLop());

for (int j = 0; j < infoAssignRooms.Count; j++)

{

if (dateStart.getStrDayOfWeek() == infoAssignRooms[j].getStrDayOfWeek())

{

if (Date.compareDate(infoAssignRooms[j].getDateStart(), dateStart) <= 0 && Date.compareDate(infoAssignRooms[j].getDateEnd(), dateStart) >= 0)

{

int width = widthCell;

int height = 0;

for (int k = infoAssignRooms[j].getTietStart(); k <= infoAssignRooms[j].getTietEnd(); k++)

{

height += heightCell;

pnCalendar.Controls.Remove(cells[k - 1, l]);

}

string[] split = cells[infoAssignRooms[j].getTietStart() - 1, l].Tag.ToString().Split(' ');

int yStart = int.Parse(split[1]);

int xStart = int.Parse(split[0]);

FlowLayoutPanel fpn = new FlowLayoutPanel()

{

Size = new Size(width, height),

Location = new Point(xStart, yStart),

AutoScroll = true,

BackColor = Color.DarkCyan,

BorderStyle = BorderStyle.FixedSingle,

};

fpn.BringToFront();

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignRooms[j].getMaLop());

fpn.Controls.Add(new Label() { Text = infoAssignTeacher.getNameSubject(), Size = new Size(170, 50), Font = new Font("Times New Roman", 10, FontStyle.Bold), ForeColor = Color.White });

fpn.Controls.Add(getLabel("Mã lớp: " + infoAssignRooms[j].getMaLop()));

fpn.Controls.Add(getLabel(infoAssignRooms[j].getNameRoom()));

fpn.Controls.Add(getLabel("Mã GV: " + infoAssignTeacher.getMaGV()));

pnCalendar.Controls.Add(fpn);

}

}

}

}

dateStart = Date.getNextDate(dateStart);

l++;

}

}

else

{

int l = 0;

while (dateStart.getStrDayOfWeek() != "Chủ Nhật")

{

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

List<InfoAssignRoom> infoAssignRooms = InfoAssignRoom.getInfo(infoAssignTeachers[i].getMaLop());

for (int j = 0; j < infoAssignRooms.Count; j++)

{

if (dateStart.getStrDayOfWeek() == infoAssignRooms[j].getStrDayOfWeek())

{

if (Date.compareDate(infoAssignRooms[j].getDateStart(), dateStart) <= 0 && Date.compareDate(infoAssignRooms[j].getDateEnd(), dateStart) >= 0)

{

int width = widthCell;

int height = 0;

for (int k = infoAssignRooms[j].getTietStart(); k <= infoAssignRooms[j].getTietEnd(); k++)

{

height += heightCell;

pnCalendar.Controls.Remove(cells[k - 1, l]);

}

string[] split = cells[infoAssignRooms[j].getTietStart() - 1, l].Tag.ToString().Split(' ');

int yStart = int.Parse(split[1]);

int xStart = int.Parse(split[0]);

FlowLayoutPanel fpn = new FlowLayoutPanel()

{

Size = new Size(width, height),

Location = new Point(xStart, yStart),

AutoScroll = true,

BackColor = Color.DarkCyan,

BorderStyle = BorderStyle.FixedSingle,

};

fpn.BringToFront();

fpn.Controls.Add(new Label() { Text = infoAssignTeachers[i].getNameSubject(), Size = new Size(170, 50), Font = new Font("Times New Roman", 10, FontStyle.Bold), ForeColor = Color.White });

fpn.Controls.Add(getLabel("Mã lớp: " + infoAssignRooms[j].getMaLop()));

fpn.Controls.Add(getLabel(infoAssignRooms[j].getNameRoom()));

fpn.Controls.Add(getLabel("Mã GV: " + infoAssignTeachers[i].getMaGV()));

pnCalendar.Controls.Add(fpn);

}

}

}

}

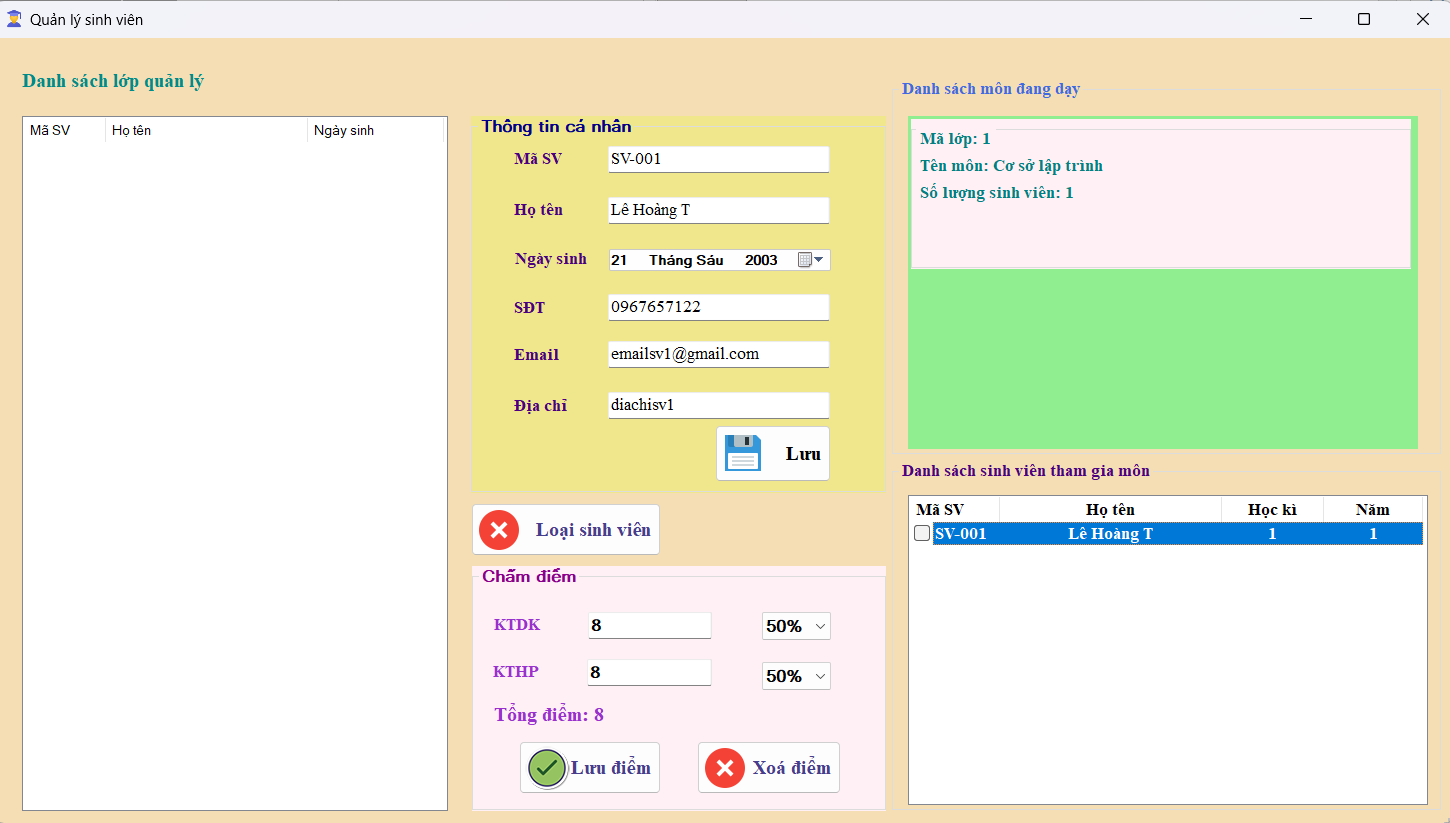
dateStart = Date.getNextDate(dateStart);

l++;

}

}

1. **Form quản lý sinh viên:**
2. **Giao diện:**

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **Thay đổi phần trăm điểm:**

private void changePercent\_ComboBox(object sender, EventArgs e)

{

if (cbbPercentKTDK.SelectedIndex == cbbPercentKTHP.Items.Count - cbbPercentKTHP.SelectedIndex - 1)

{

return;

}

ComboBox currentCbb = (ComboBox)sender;

if (currentCbb == cbbPercentKTDK)

{

cbbPercentKTHP.SelectedIndex = currentCbb.Items.Count - currentCbb.SelectedIndex - 1;

}

else

{

cbbPercentKTDK.SelectedIndex = currentCbb.Items.Count - currentCbb.SelectedIndex - 1;

}

}

* **Click chọn môn đang dạy:**

private void selectInfoAssign(object sender, EventArgs e)

{

currentAssignSubject = (InfoAssignTeacher)((Control)sender).Tag;

lvStudentLearn.Items.Clear();

List<Student> students = InfoAssignSubject.getStudentsAssignSubject(currentAssignSubject.getMaLop());

for (int j = 0; j < students.Count; j++)

{

List<InfoAssignSubject> infoAssigns = InfoAssignSubject.getAssign(students[j].getID(), currentAssignSubject.getMaLop());

for (int i = 0; i < infoAssigns.Count; i++)

{

ListViewItem li = new ListViewItem()

{

Text = students[j].getID(),

};

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = students[j].getFullName().getFullName() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infoAssigns[i].getHocKi().ToString() });

li.SubItems.Add(new ListViewItem.ListViewSubItem() { Text = infoAssigns[i].getYear().ToString() });

lvStudentLearn.Items.Add(li);

}

}

clearText();

}

* **KeyPress nhập điểm:**

private void keyPressInputValueMark(object sender, KeyPressEventArgs e)

{

if (e.KeyChar != ',' && (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30)))

{

e.Handled = true;

}

}

* **Textchanged nhập điểm:**

private void inputValueMark(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtKTDK.Text) || string.IsNullOrEmpty(txtKTHP.Text))

{

lblSum.Text = "Tổng điểm: 0";

return;

}

double ptA = double.Parse(txtKTDK.Text);

double ptB = double.Parse(txtKTHP.Text);

double result = ptA \* (cbbPercentKTDK.SelectedIndex + 1) / 10 + ptB \* (cbbPercentKTHP.SelectedIndex + 1) / 10;

lblSum.Text = "Tổng điểm: " + result;

}

* **Click lưu thông tin sinh viên**

private void btnSave\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtName.Text) || string.IsNullOrEmpty(txtPhone.Text) || string.IsNullOrEmpty(txtAddress.Text) || string.IsNullOrEmpty(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa nhập đủ thông tin!");

return;

}

if (Person.isExistPersonID(txtID.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Mã sinh viên đã được sử dụng!");

txtID.Focus();

return;

}

if (!Person.isPhoneValid(txtPhone.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Số điện thoại không hợp lệ!");

txtPhone.Focus();

return;

}

if (!Person.isEmailValid(txtEmail.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Email không hợp lệ!");

txtEmail.Focus();

return;

}

Student student = new Student(currentAcct.getID(), txtPhone.Text, txtName.Text, txtAddress.Text, txtEmail.Text, new Date(dtpBirth.Value));

if (currentAcct.updateAccount(student, txtID.Text))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Cập nhật thông tin thành công!");

}

currentAcct = new Account(currentAcct.getName());

}

* **Click chấm điểm:**

private void btnMark\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(txtKTDK.Text) || string.IsNullOrEmpty(txtKTHP.Text))

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Điểm không thể để trống!");

return;

}

int percentKTDK = (cbbPercentKTDK.SelectedIndex + 1) \* 10;

List<MarkSubject> pointSubs = MarkSubject.getAllPoint();

for (int i = 0; i < pointSubs.Count; i++)

{

pointSubs[i].updateMarkSubject(pointSubs[i].getKTDK(), pointSubs[i].getKTHP(), percentKTDK);

}

Student student = (Student)Person.getInfo("ID", currentAcct.getID());

InfoAssignSubject infoAssign = InfoAssignSubject.getInfo(student.getID(), currentAssignSubject.getMaLop(), student.getHocKi(), student.getYear());

if (MarkSubject.isExistsMarkSubject(currentAcct.getID(), currentAssignSubject.getMaLop(), infoAssign.getHocKi(), infoAssign.getYear()))

{

MarkSubject mark = MarkSubject.getInfo(currentAssignSubject.getMaLop(), student.getID(), infoAssign.getHocKi(), infoAssign.getYear());

;if (!mark.updateMarkSubject(double.Parse(txtKTDK.Text), double.Parse(txtKTHP.Text), percentKTDK))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Lưu điểm thất bại!");

}

else

{

clearText();

lvStudentLearn.SelectedItems.Clear();

}

}

else

{

MarkSubject mark = new MarkSubject(currentAcct.getID(), currentAssignSubject.getMaLop(), double.Parse(txtKTDK.Text), double.Parse(txtKTHP.Text), percentKTDK, infoAssign.getHocKi(), infoAssign.getYear());

if (mark.insertMark())

{

clearText();

lvStudentLearn.SelectedItems.Clear();

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Lưu điểm thất bại!");

}

}

}

* **Chọn sinh viên trong lớp học chính (làm cố vấn):**

private void lvMainClass\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvMainClass.SelectedIndices.Count != 0)

{

lvStudentLearn.SelectedIndices.Clear();

string ID = lvMainClass.Items[lvMainClass.SelectedIndices[0]].Text;

string nameAcct = (string) Account.getField("TenTK", "ID", ID);

currentAcct = new Account(nameAcct);

gbInfo.Enabled = true;

gbMark.Enabled = false;

Student student = (Student) Person.getInfo("ID", ID);

txtName.Text = student.getFullName().getFullName();

txtAddress.Text = student.getAddress();

txtEmail.Text = student.getEmail();

txtID.Text = ID;

txtPhone.Text = student.getPhone();

dtpBirth.Value = student.getBirth().getDate();

}

else

{

currentAcct = null;

gbInfo.Enabled = false;

clearText();

}

}

* **Chọn sinh viên trong lớp môn học đang dạy:**

private void lvStudentLearn\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvStudentLearn.SelectedIndices.Count != 0)

{

lvMainClass.SelectedIndices.Clear();

btnKickStudent.Enabled = true;

ListViewItem li = lvStudentLearn.SelectedItems[0];

string nameAcct = (string)Account.getField("TenTK", "ID", lvStudentLearn.Items[lvStudentLearn.SelectedIndices[0]].Text);

currentAcct = new Account(nameAcct);

Student student = (Student)Person.getInfo("ID", currentAcct.getID());

txtName.Text = student.getFullName().getFullName();

txtAddress.Text = student.getAddress();

txtEmail.Text = student.getEmail();

txtID.Text = student.getID();

txtPhone.Text = student.getPhone();

dtpBirth.Value = student.getBirth().getDate();

gbInfo.Enabled = true;

gbMark.Enabled = true;

InfoAssignSubject infoAssign = InfoAssignSubject.getInfo(student.getID(), currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

MarkSubject mark = MarkSubject.getInfo(currentAssignSubject.getMaLop(), currentAcct.getID(), infoAssign.getHocKi(), infoAssign.getYear());

if (mark == null)

{

btnDeleteMark.Enabled = false;

txtKTDK.Clear();

txtKTHP.Clear();

}

else

{

if (infoAssign.getHocKi() != student.getHocKi() || infoAssign.getYear() != student.getYear())

{

btnDeleteMark.Enabled = false;

}

else

{

btnDeleteMark.Enabled = true;

}

int percent = mark.getPercentKTDK();

cbbPercentKTDK.SelectedIndex = percent / 10 - 1;

txtKTDK.Text = mark.getKTDK().ToString();

txtKTHP.Text = mark.getKTHP().ToString();

}

if (infoAssign.getHocKi() != student.getHocKi() || infoAssign.getYear() != student.getYear())

{

btnMark.Enabled = false;

}

else

{

btnMark.Enabled = true;

}

}

else

{

clearText();

btnDeleteMark.Enabled = false;

btnKickStudent.Enabled = false;

currentAcct = null;

gbInfo.Enabled = false;

gbMark.Enabled = false;

}

}

* **KeyPress nhập mã sinh viên:**

private void txtID\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

{

if (txtID.Text == "SV-")

{

e.Handled = true;

}

}

else if (char.IsLetter(e.KeyChar) || char.IsSymbol(e.KeyChar) || (!char.IsLetterOrDigit(e.KeyChar) && e.KeyChar > 30))

{

e.Handled = true;

}

}

* **Click loại sinh viên khỏi lớp:**

private void btnKickStudent\_Click(object sender, EventArgs e)

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn loại sinh viên khỏi lớp " + teacher.getClass().getName() +"?") != DialogResult.Yes)

{

return;

}

if (lvMainClass.SelectedIndices.Count != 0)

{

if (!InfoAssignStudent.kickStudent(txtID.Text))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Loại sinh viên khỏi lớp thất bại!");

}

else

{

loadMainClassStudents();

}

}

else

{

ListViewItem li = lvStudentLearn.SelectedItems[0];

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(txtID.Text, currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

if (!infoAssignSubject.deleteAssignSubject())

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Loại sinh viên khỏi lớp thất bại!");

}

else

{

loadClassTeach();

}

}

}

* **Click xoá điểm sinh viên:**

private void btnDeleteMark\_Click(object sender, EventArgs e)

{

ListViewItem li = lvStudentLearn.SelectedItems[0];

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn có chắc chắn muốn xoá điểm của sinh viên " + txtID.Text + " - HK" + li.SubItems[2].Text + " - Năm " + li.SubItems[3].Text + " - Mã lớp: " + currentAssignSubject.getMaLop() +"?") != DialogResult.Yes)

{

return;

}

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(txtID.Text, currentAssignSubject.getMaLop(), int.Parse(li.SubItems[2].Text), int.Parse(li.SubItems[3].Text));

if (!MarkSubject.deleteMarkSubject(txtID.Text, currentAssignSubject.getMaLop(), infoAssignSubject.getHocKi(), infoAssignSubject.getYear()))

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá điểm sinh viên thất bại!");

}

else

{

loadClassTeach();

}

}

* **ItemChecked sinh viên thuộc lớp chính:**

private void lvMainClass\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvMainClass.CheckedIndices.Count != 0 || lvStudentLearn.CheckedIndices.Count != 0)

{

btnKickStudent.Enabled = true;

}

else

{

btnKickStudent.Enabled = false;

}

}

* **ItemChecked sinh viên thuộc lớp đang dạy:**

private void lvStudentLearn\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (lvMainClass.CheckedIndices.Count != 0 || lvStudentLearn.CheckedIndices.Count != 0)

{

btnKickStudent.Enabled = true;

}

else

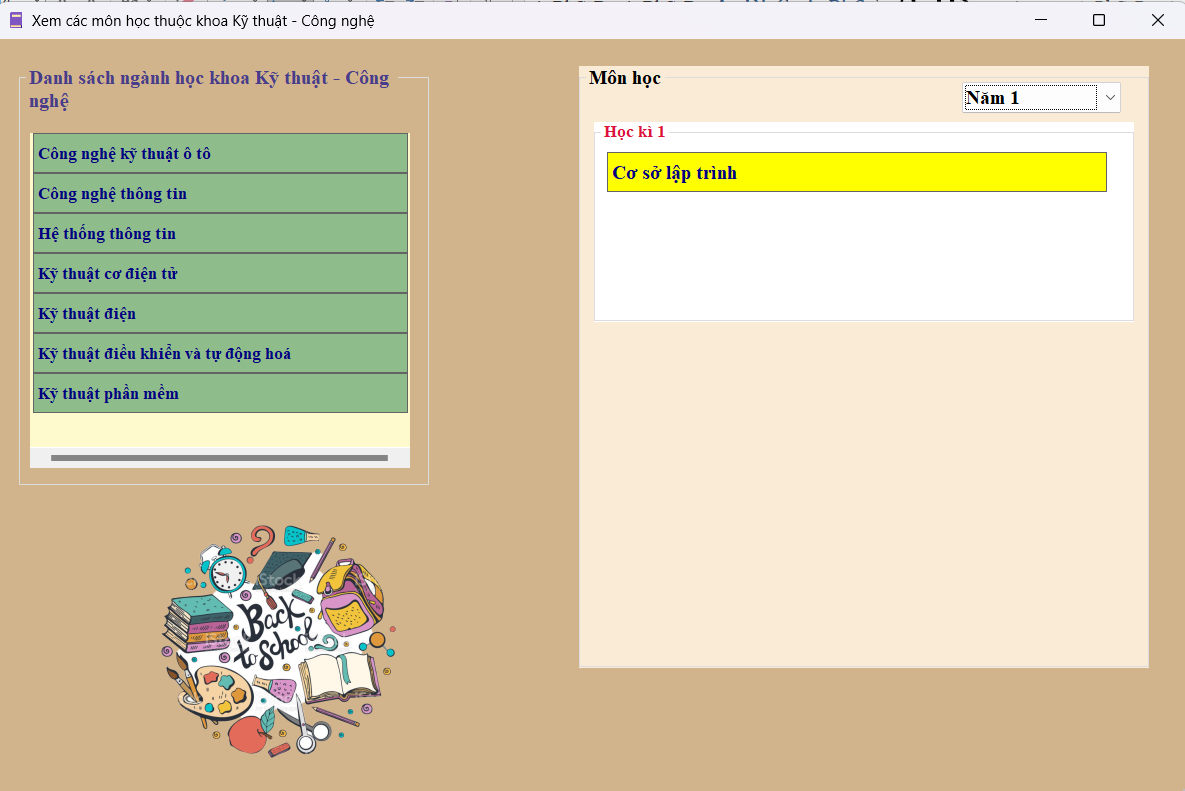
{

btnKickStudent.Enabled = false;

}

}

1. **Form xem các môn học thuộc khoa (Giảng viên) / ngành (Sinh viên):**
2. **Giao diện:**



1. **Các sự kiện – Event Handlers (kèm code):**

* **SelectedIndexChanged Combobox ngành:**

private void selectMajor(object sender, EventArgs e)

{

currentMajor = Major.getInfo(((Control)sender).Text);

currentYear = 1;

loadSubjects(1, true);

if (cbbYear.Items.Count != 0)

{

cbbYear.SelectedIndex = 0;

}

else

{

currentYear = 0;

}

}

* **SelectedIndexChanged Combox năm:**

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

if (currentYear == cbbYear.SelectedIndex + 1)

{

return;

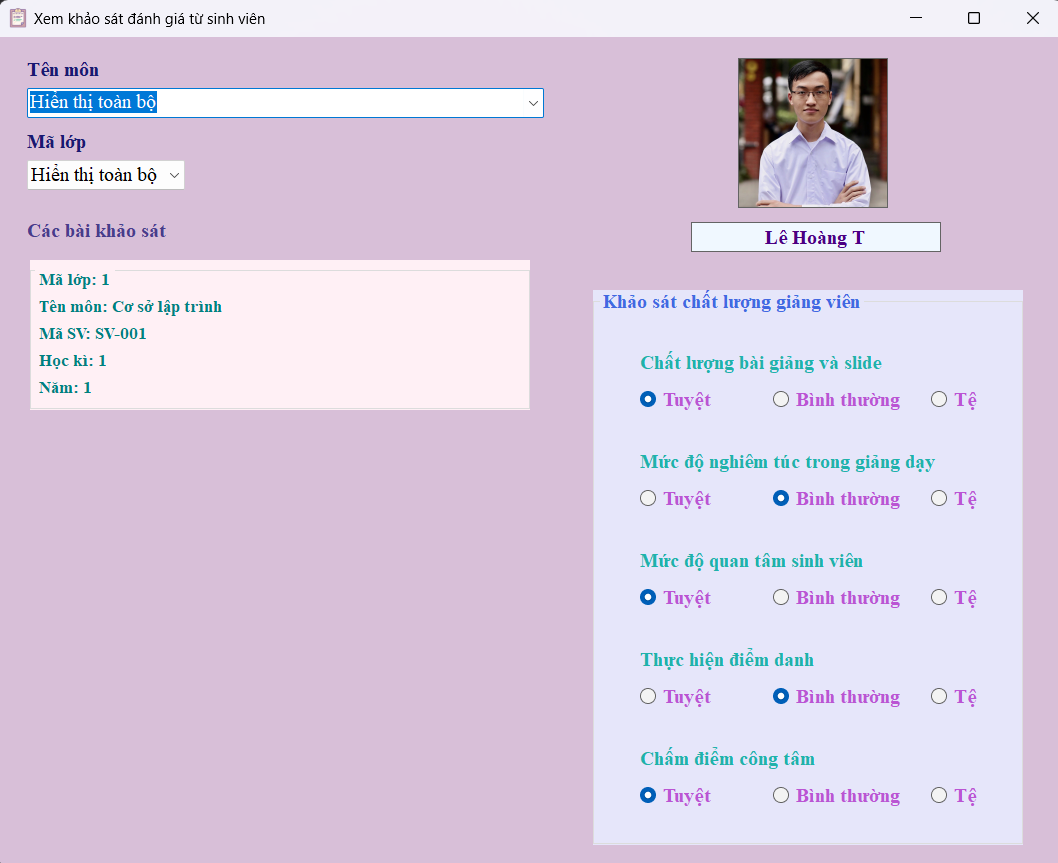
}

currentYear = cbbYear.SelectedIndex + 1;

loadSubjects(cbbYear.SelectedIndex + 1);

}

1. **Form xem các khảo sát:**
2. **Giao diện**

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **SelectedIndexChanged Combobox ngành:**

private void cbbNameSubject\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbMaLop.Items.Clear();

cbbMaLop.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < infoAssignTeachers.Count; i++)

{

if (cbbNameSubject.SelectedIndex != 0 && infoAssignTeachers[i].getNameSubject() != cbbNameSubject.Text)

{

infoAssignTeachers.RemoveAt(i);

i--;

continue;

}

cbbMaLop.Items.Add(infoAssignTeachers[i].getMaLop());

}

cbbMaLop.SelectedIndex = 0;

}

* **SelectedIndexChanged Combobox mã lớp:**

private void cbbMaLop\_SelectedIndexChanged(object sender, EventArgs e)

{

fpnSurvey.Controls.Clear();

List<Survey> lstSurvey = Teacher.getAllSurvey(teacher.getID());

for (int i = 0; i < lstSurvey.Count; i++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstSurvey[i].getMaLop());

if (cbbNameSubject.SelectedIndex != 0 && infoAssignTeacher.getNameSubject() != cbbNameSubject.Text)

{

lstSurvey.RemoveAt(i);

i--;

continue;

}

if (cbbMaLop.SelectedIndex!= 0 && lstSurvey[i].getMaLop().ToString() != cbbMaLop.Text)

{

lstSurvey.RemoveAt(i);

i--;

continue;

}

fpnSurvey.Controls.Add(getGroupSurvey(lstSurvey[i]));

}

clearSurvey();

}

* **Click xem khảo sát:**

private void selectSurvey(object sender, EventArgs e)

{

Survey survey = (Survey)((Control)sender).Tag;

if (survey == null)

{

return;

}

Student student = (Student)Person.getInfo("ID", survey.getMaSV());

lblNameStudent.Text = student.getFullName().getFullName();

object img = Account.getField("AnhDaiDien", "ID", student.getID());

if (img != null)

{

ptrAvt.Image = Avatar.ConvertBytesToImage((byte[])img);

}

else

{

ptrAvt.Image = Image.FromFile("Assets/Imgs/No Image.png");

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

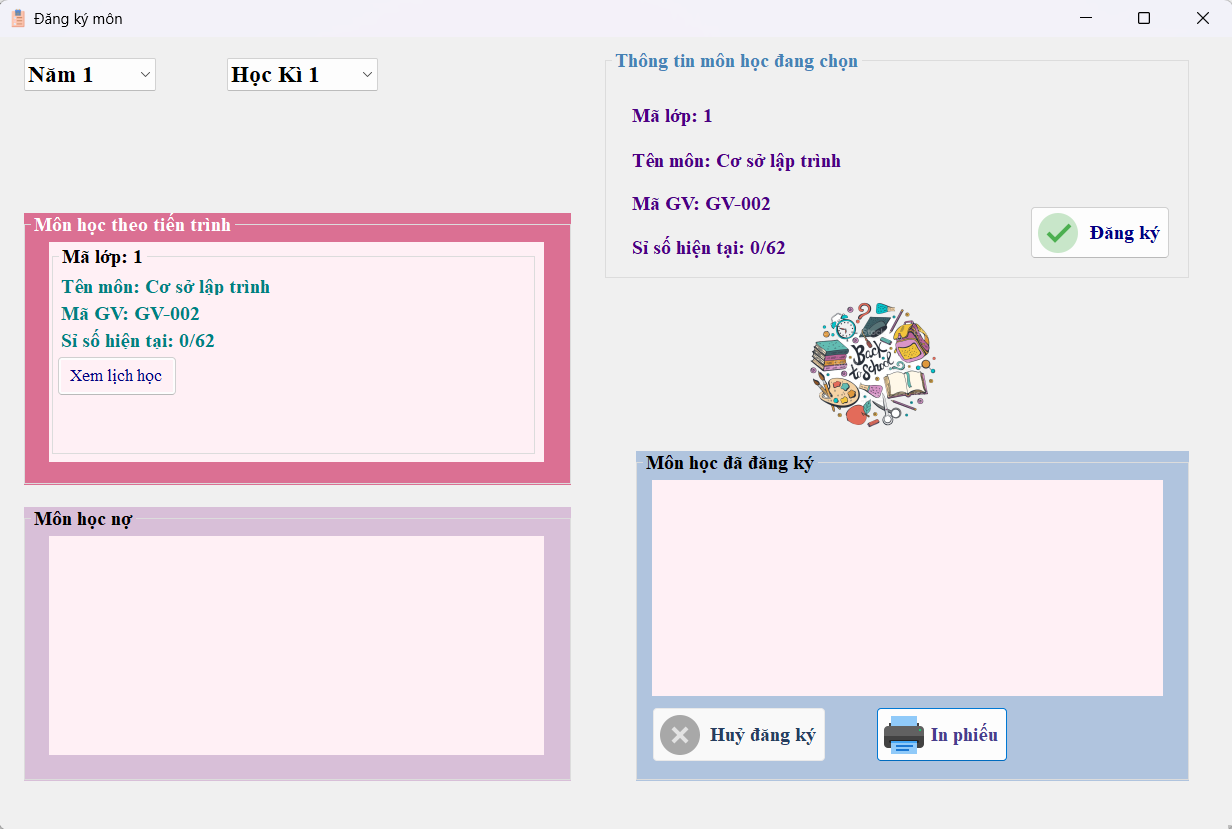
}

}

}

}

1. **Form đăng ký môn:**
2. **Giao diện:**



1. **Các sự kiện – Event Handlers (kèm code):**

* **SelectedIndexChanged Combobox học kì:**

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

int maxHocki;

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

maxHocki = student.getHocKi();

}

else

{

maxHocki = 2;

}

for (int i = 1; i <= maxHocki; i++)

{

cbbHocKi.Items.Add("Học Kì " + i);

}

cbbHocKi.SelectedIndex = cbbHocKi.Items.Count - 1;

}

* **SekectedIndexChanged Combobox học kì:**

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbYear.SelectedIndex == -1)

{

return;

}

loadSubject();

loadSubjectAssigned();

}

* **Click xem lịch học:**

private void selectDate(object sender, EventArgs e)

{

int maLop = (int)((Control)sender).Tag;

frmDateAssign frm = new frmDateAssign(maLop);

frm.ShowDialog();

}

* **Click chọn môn theo tiến trình để đăng ký:**

private void selectAssignTeacher(object sender, EventArgs e)

{

InfoAssignTeacher infoAssignTeacher = (InfoAssignTeacher)((Control)sender).Tag;

if (infoAssignTeacher.getMaxCount() == InfoAssignSubject.getStudentsAssignSubject(infoAssignTeacher.getMaLop()).Count())

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Sỉ số đã đạt tối đa!!");

return;

}

currentInfoAssignTeacher = infoAssignTeacher;

if (cbbHocKi.SelectedIndex + 1 != student.getHocKi() || cbbYear.SelectedIndex + 1 != student.getYear())

{

btnAssignSubject.Enabled = false;

}

else

{

btnAssignSubject.Enabled = true;

}

lblMaLop.Text = "Mã lớp: " + currentInfoAssignTeacher.getMaLop().ToString();

lblNameSubject.Text = "Tên môn: " + currentInfoAssignTeacher.getNameSubject();

lblMaGV.Text = "Mã GV: " + currentInfoAssignTeacher.getMaGV();

lblCurrentCount.Text = "Sỉ số hiện tại: " + InfoAssignSubject.getStudentsAssignSubject(currentInfoAssignTeacher.getMaLop()).Count() + "/" + currentInfoAssignTeacher.getMaxCount();

btnCancelAssignSubject.Enabled = false;

}

* **Click chọn môn đã đăng ký:**

private void selectAssignedTeacher(object sender, EventArgs e)

{

InfoAssignTeacher infoAssignTeacher = (InfoAssignTeacher)((Control)sender).Tag;

InfoAssignSubject infoAssignSubject = InfoAssignSubject.getInfo(student.getID(), infoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (infoAssignSubject.getYear() != cbbYear.SelectedIndex + 1 || infoAssignSubject.getHocKi() != cbbHocKi.SelectedIndex + 1)

{

return;

}

currentInfoAssignTeacher = infoAssignTeacher;

if (cbbHocKi.SelectedIndex + 1 != student.getHocKi() || cbbYear.SelectedIndex + 1 != student.getYear())

{

btnCancelAssignSubject.Enabled = false;

}

else

{

btnCancelAssignSubject.Enabled = true;

}

lblMaLop.Text = "Mã lớp: " + currentInfoAssignTeacher.getMaLop().ToString();

lblNameSubject.Text = "Tên môn: " + currentInfoAssignTeacher.getNameSubject();

lblMaGV.Text = "Mã GV: " + currentInfoAssignTeacher.getMaGV();

lblCurrentCount.Text = "Sỉ số hiện tại: " + InfoAssignSubject.getStudentsAssignSubject(currentInfoAssignTeacher.getMaLop()).Count() + "/" + currentInfoAssignTeacher.getMaxCount();

btnAssignSubject.Enabled = false;

}

* **Click đăng ký môn:**

private void btnAssignSubject\_Click(object sender, EventArgs e)

{

if (currentInfoAssignTeacher == null)

{

return;

}

if (InfoAssignSubject.isSameDateAssignSubject(student.getID(), currentInfoAssignTeacher.getMaLop()))

{

MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Lịch học bị trùng!");

return;

}

InfoAssignTeacher CurrentInfoAssignTeacher = InfoAssignTeacher.getInfo(currentInfoAssignTeacher.getMaLop());

List<InfoAssignSubject> infoAssignSubjects = InfoAssignSubject.getAllAssignSubject();

for (int i = 0; i < infoAssignSubjects.Count; i++)

{

if (!InfoAssignSubject.isExistsAssignSubject(student.getID(), infoAssignSubjects[i].getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1))

{

continue;

}

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignSubjects[i].getMaLop());

if (infoAssignTeacher.getNameSubject() == CurrentInfoAssignTeacher.getNameSubject())

{

if (MessageInfo.makeMessage("Question", "Hỏi lại cho chắc", "Bạn đã đăng ký môn " + currentInfoAssignTeacher.getNameSubject() + ", Bạn có chắc muốn thay thế lớp mới không?") != DialogResult.Yes)

{

return;

}

infoAssignSubjects[i].deleteAssignSubject();

break;

}

}

InfoAssignSubject info = new InfoAssignSubject(student.getID(), currentInfoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

info.addAssignSubject();

loadSubject();

loadRelearn();

loadSubjectAssigned();

}

* **Click huỷ đăng môn:**

private void btnCancelAssignSubject\_Click(object sender, EventArgs e)

{

InfoAssignSubject info = InfoAssignSubject.getInfo(student.getID(), currentInfoAssignTeacher.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (!info.deleteAssignSubject())

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Xoá đăng ký thất bại!");

return;

}

loadSubject();

loadRelearn();

loadSubjectAssigned();

btnCancelAssignSubject.Enabled = false;

}

* **Click in phiếu đăng ký môn:**

private void btnPrint\_Click(object sender, EventArgs e)

{

if (infoAssignedSubjects.Count == 0)

{

MessageInfo.makeMessage("Warning", "Cảnh báo", "Bạn chưa đăng ký môn nào để in phiếu!");

return;

}

List<string> lstContent = new List<string>();

string nameFile = "Phiếu đăng ký môn học HK" + (cbbHocKi.SelectedIndex + 1) + " - Năm " + (cbbYear.SelectedIndex + 1) + " - " + student.getFullName().getFullName() + " - " + student.getID();

lstContent.Add(getTextAlignCenter(nameFile, 76));

lstContent.Add("----------------------------------------------------------------------------");

if (File.Exists(nameFile + ".txt"))

{

int i = 1;

while (File.Exists(nameFile + " (" + i + ").txt"))

{

i++;

}

nameFile += " (" + i + ").txt";

}

else

{

nameFile += ".txt";

}

lstContent.Add("| Mã lớp | Tên môn | Mã GV |");

lstContent.Add("----------------------------------------------------------------------------");

for (int j = 0; j < infoAssignedSubjects.Count; j++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(infoAssignedSubjects[j].getMaLop());

string line = "|" + getTextAlignCenter(infoAssignTeacher.getMaLop().ToString(), 12) + "|";

line += getTextAlignCenter(infoAssignTeacher.getNameSubject(), 49) + "|";

line += getTextAlignCenter(infoAssignTeacher.getMaGV(), 11) + "|";

lstContent.Add(line);

lstContent.Add("----------------------------------------------------------------------------");

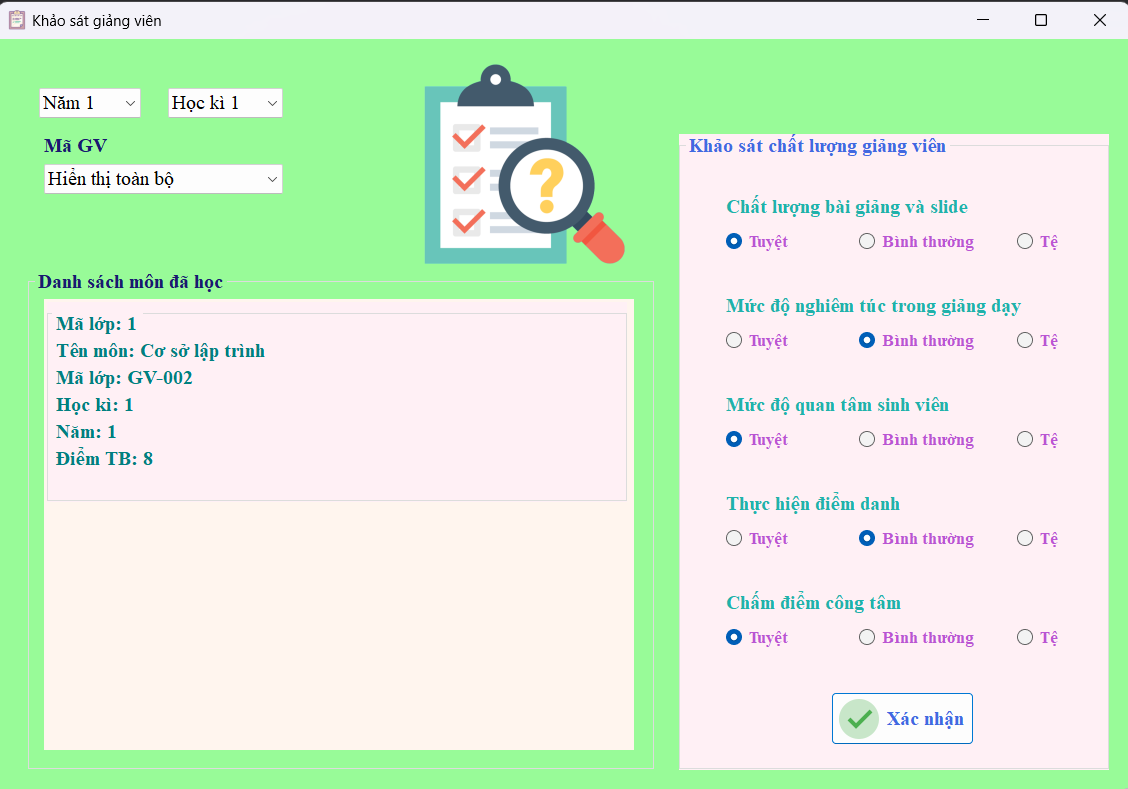
}

File.WriteAllLines(nameFile, lstContent);

MessageInfo.makeMessage("Information", "Chúc mừng", "In phiếu thành công!");

}

1. **Form khảo sát giảng viên:**
2. **Giao diện:**

****

1. **Các sự kiện – Event Handlers (kèm code):**

* **Click chọn môn để xem:**

private void selectAssignSubject(object sender, EventArgs e)

{

currentInfoAssignSubject = (InfoAssignSubject)((Control)sender).Tag;

MarkSubject mark = MarkSubject.getInfo(currentInfoAssignSubject.getMaLop(), student.getID(), currentInfoAssignSubject.getHocKi(), currentInfoAssignSubject.getYear());

if (mark == null || cbbYear.SelectedIndex + 1 != student.getYear() || cbbHocKi.SelectedIndex + 1 != student.getHocKi())

{

gbSurvey.Enabled = false;

clearSurvey();

return;

}

clearSurvey();

gbSurvey.Enabled = true;

Survey survey = Survey.getSurvey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (survey == null)

{

return;

}

int[] content = survey.getContent();

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

int value = int.Parse(rdo.Tag.ToString());

if (value == content[index])

{

RadioButton radio = (RadioButton)rdo;

radio.Checked = true;

break;

}

}

}

}

}

}

* **Click xác nhận (gửi, cập nhật) khảo sát:**

private void btnAccept\_Click(object sender, EventArgs e)

{

if (currentInfoAssignSubject == null)

{

return;

}

int[] content = new int[5];

foreach (Control pn in gbSurvey.Controls)

{

if (pn is Panel)

{

int index = int.Parse(pn.Tag.ToString()) - 1;

foreach (Control rdo in pn.Controls)

{

if (rdo is RadioButton)

{

RadioButton radio = (RadioButton)rdo;

if (radio.Checked)

{

content[index] = int.Parse(rdo.Tag.ToString());

break;

}

}

}

}

}

Survey survey = Survey.getSurvey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1);

if (survey == null)

{

survey = new Survey(student.getID(), currentInfoAssignSubject.getMaLop(), cbbHocKi.SelectedIndex + 1, cbbYear.SelectedIndex + 1, content);

if (survey.insertSurvey())

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Đã gửi khảo sát!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Gửi khảo sát thất bại!");

}

}

else

{

if (survey.updateSurvey(content))

{

MessageInfo.makeMessage("Information", "Chúc mừng", "Cập nhật khảo sát thành công!");

}

else

{

MessageInfo.makeMessage("Error", "Rất tiếc", "Cập nhật khảo sát thất bại!");

}

}

}

* **SelectedIndexChanged Combobox Năm:**

private void cbbYear\_SelectedIndexChanged(object sender, EventArgs e)

{

cbbHocKi.Items.Clear();

int maxHocki;

if (cbbYear.SelectedIndex + 1 == student.getYear())

{

maxHocki = student.getHocKi();

}

else

{

maxHocki = 2;

}

for (int i = 1; i <= maxHocki; i++)

{

cbbHocKi.Items.Add("Học kì " + i);

}

cbbHocKi.SelectedIndex = cbbHocKi.Items.Count - 1;

gbSurvey.Enabled = false;

clearSurvey();

}

* **SelectedIndexChanged Combobox Học kì:**

private void cbbHocKi\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cbbYear.SelectedIndex == -1)

{

return;

}

lstInfoAssigns = student.getInfoAssignSubject();

cbbMaGV.Items.Clear();

cbbMaGV.Items.Add("Hiển thị toàn bộ");

for (int i = 0; i < lstInfoAssigns.Count; i++)

{

if (cbbYear.SelectedIndex + 1 != lstInfoAssigns[i].getYear())

{

continue;

}

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstInfoAssigns[i].getMaLop());

cbbMaGV.Items.Add(infoAssignTeacher.getMaGV());

}

cbbMaGV.SelectedIndex = 0;

gbSurvey.Enabled = false;

clearSurvey();

}

* **SelectedIndexChanged Combobox mã giảng viên để lọc:**

private void cbbMaGV\_SelectedIndexChanged(object sender, EventArgs e)

{

lstInfoAssigns = student.getInfoAssignSubject();

for (int i = 0; i < lstInfoAssigns.Count; i++)

{

InfoAssignTeacher infoAssignTeacher = InfoAssignTeacher.getInfo(lstInfoAssigns[i].getMaLop());

if (cbbYear.SelectedIndex + 1 != lstInfoAssigns[i].getYear() || cbbMaGV.SelectedIndex != 0 && infoAssignTeacher.getMaGV() != cbbMaGV.Text)

{

lstInfoAssigns.RemoveAt(i);

i--;

}

}

gbSurvey.Enabled = false;

clearSurvey();

loadInfoAssignSubject();

}

1. **Kế hoạch thực hiện tiếp theo:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TT** | **Nội dung** | **Công cụ** | **Sản phẩm** | **Người thực hiện** |
|  |  |  |  |  |
|  |  |  |  |  |