

**Smart Library**

Name Students:

- Naif M. Alqubalee

- Basel Adel

- Mahphoud Alajem

- Omer Ahmed

- Ahmed Jaron

Content ………………………………………………………………………………… 1

ـــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــ

Introduction ……………………………………………………………………………………… 2

Problem statement …………………………………………………………………………….... 2.1

Project scope ………………………………………………………………………………………. 2.2

Actors ………………………………………………………………………………………………… 2.3

Requirements ………………………………………………………………………………………………. 3

Functional Requirements ………………………………………………………………………………… 3.1

Non-Functional Requirements ………………………………………………………………………... 3.2

System analysis and design UML diagrams……………………………………..... 4

Use Case Diagram & table ……………………………………………………………………………….. 4.1

Class Diagram ………………………………………………………………………………..................... 4.2

ER Diagram ………………………………………………………………………………………….............. 4.3

ERD to Schema …………………………………………………………………………………….............. 4.4

Project Code & Forms……………………………………………………………………….. 9

First Interface (Form 1) …………………………………………………………………………………….. 9

Sign in Student (Form 2) ……………………………………………………………………………………10

Sign in Admin (Form 3) ……………………………………………………………………………………..12

Sign up Student (Form 4) ………………………………………………………………………………….14

Main Student Page (Form 5) …………………………………………………………………………….16

Main Admin Page (Form 6) ……………………………………………………………………………….29

Add Book (Form 7) ……………………………………………………………………………………………52

Update Book (Form 8) ………………………………………………………………………………………54

2. Introduction

* The University Library Management App is designed to provide students with a simple way to access and download the university's book collection. This desktop application aims to enhance the student experience by making it easier to find and view books that related to their studies.
* Key Features
* Simple User Interface: The app welcomes users with a main page that allows them to choose their role—either as a student or an admin. Each role has a specific experience designed to meet their specific needs.
* Student Login: Students can log in using their university Name(optional). Once logged in, they are presented with a main page that features various filters such as level, department, teacher, and download options. These filters help students quickly find the books they need from database.
* Admin Access: Administrators can log in using their Name(optional) and Password(required). After a successful login, admins have access to additional functionalities, including the ability to add or delete or update books from the database. This ensures that the library's resources are always up to date.
* Book Management: The app provides dedicated forms for adding and update books. Admins can enter book details, such as the book name, link, teacher's name, and department, to upload new resources.

This application not only streamlines the process of finding and managing books but also fosters a more organized and efficient library system at the university. By integrating user-friendly forms and efficient filtering options, the app aims to enhance the educational experience for all users.

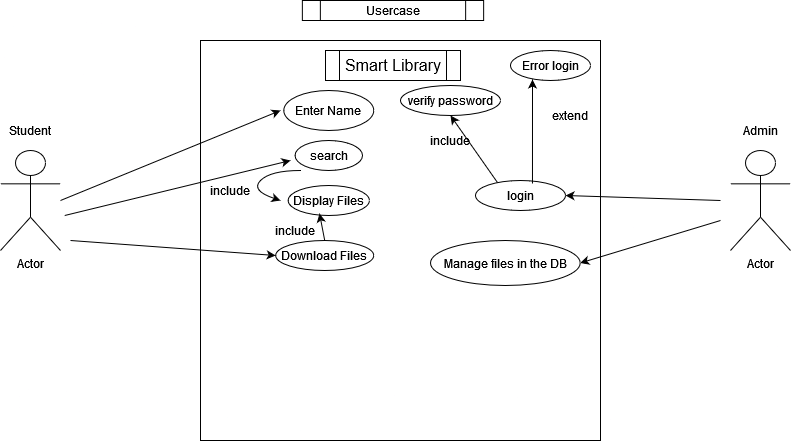
3. Requirements

* The library project is an important step towards improving the learning experience at the university by identifying various external sources that help students during their studies at the college. Through functional and non-functional requirements, the project can provide an effective platform for students to access study materials easily.
* 3.1 Functional Requirements:

1. Bring, delete or Update files: Ability to easily bring, delete or modify files to the system from the database.
2. Upload files: This allows the admin to easily upload information about new files to the system so that they can be added to the database.
3. File display interfaces: Provides convenient display interfaces to enable users and administrators to see the available files. These interfaces include details such as names, departments, etc.
4. Ability to filtering files: Users and Admin should be able to filter files according to certain criteria such as teacher name, department, or level. This helps in facilitating quick and efficient access to the required files.
5. Search Feature: Provides a search feature that allows users or Admin to enter keywords to quickly find files.

* 3.2 Non-Functional Requirements:

1. Professional display interface design: User interfaces should be professionally designed, making it easy for users to interact with the system.
2. Response: The system response must be fast, even when there are a large number of users at the same time.
3. Speed: File upload and download speeds should be high to ensure that there are no delays that affect the user experience.



| **Use Case ID** | **Use Case Name** | **Actors** | **Description** | **Preconditions** | **Postconditions** |
| --- | --- | --- | --- | --- | --- |
| UC1 | Student Login | Student | Allows students to log in using their credentials to access library features such as filtering and downloads. | The student must have valid credentials. The system must be connected to the student database. | The student is logged in and redirected to their main page. |
| UC2 | Admin Login | Admin | Enables administrators to log in to manage library resources. | The admin must have a valid username and password. | The admin is logged in and redirected to the admin dashboard. |
| UC3 | View Book List | Student, Admin | Displays a list of available books, with options to filter based on criteria like level, department, or teacher. | The user must be logged in. | The user sees a filtered list of books based on selected criteria. |
| UC4 | Search Books | Student, Admin | Allows users to search for books using keywords. | The user must be logged in. | The user sees a list of books matching the search criteria. |
| UC5 | Upload Book | Admin | Enables the admin to upload details of new books, such as name, link, teacher, and department. | The admin must be logged in. Valid book details must be provided. | The new book details are stored in the database and visible to users. |
| UC6 | Update Book Details | Admin | Allows the admin to modify existing book details, including name, link, teacher, and department. | The admin must be logged in. The book must exist in the database. | The updated book details are saved and reflected in the system. |
| UC7 | Delete Book | Admin | Allows the admin to delete a book from the library. | The admin must be logged in. The book must exist | The selected book is removed from the database. |
| UC8 | Filter Books | Student, Admin | Provides options to filter books by department, level, or teacher for easy access. | The user must be logged in. | The user sees a filtered list of books based on the selected criteria. |
| UC9 | View User Profile | Student, Admin | Allows users to view their profile details, such as name and access level. | The user must be logged in. | The user sees their profile information. |
| UC10 | Log Out | Student, Admin | Ends the user session and logs them out of the system. | The user must be logged in. | The user is logged out, and the session is terminated. |

teach

Relationships in DB

has

M:1

1:1

1:1

Depts

Books

Teachers

1:M

N:M

M:N

In

Figure ERD

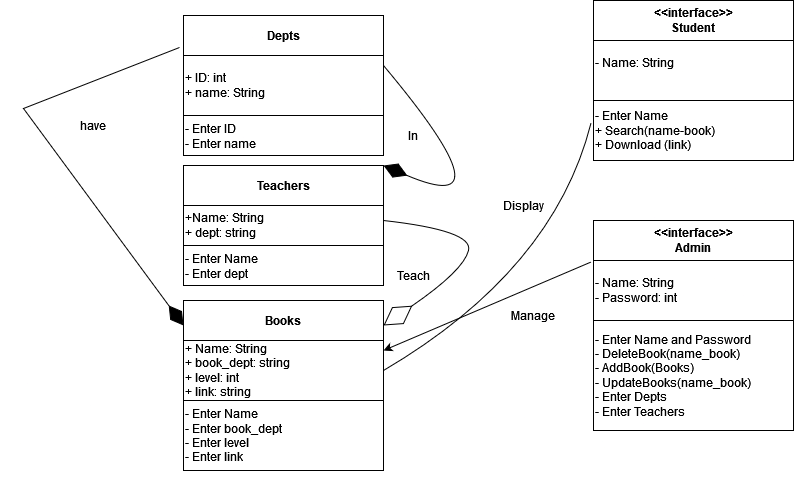


Figure Class Diagram

Books

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Dept | Level | Link | Teachers ID | Depts ID |

Teachers

|  |  |  |
| --- | --- | --- |
| ID | Name | Dept |

Depts

|  |  |  |
| --- | --- | --- |
| ID | Name |  |

In

|  |  |  |
| --- | --- | --- |
| Teachers ID | Depts ID |  |

|  |  |
| --- | --- |
| Depts ID | Books ID |

Has

Schema From ERD

Project Code:

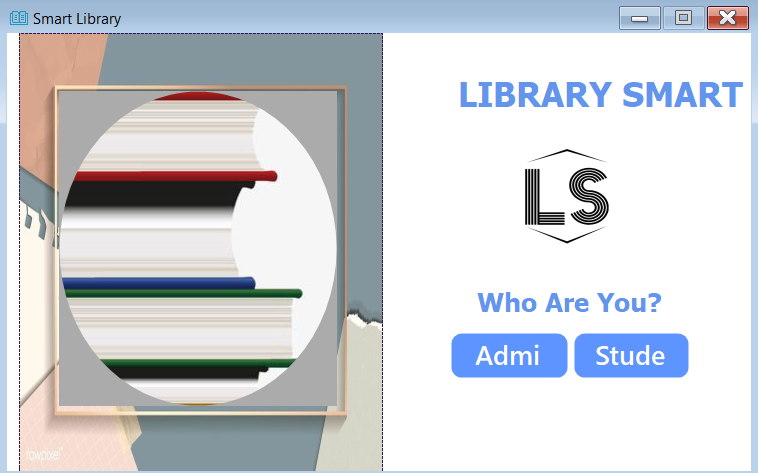


Figure Form 1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Smart\_Library

{

public partial class first\_form : Form

{

public first\_form()

{

InitializeComponent();

}

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

{

secondForm sAdminForm = new secondForm();

sAdminForm.Show();

this.Hide();

}

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

{

second\_pass\_form\_for\_students sStudentForm = new second\_pass\_form\_for\_students();

sStudentForm.Show();

this.Hide();

}

private void guna2HtmlLabel1\_Click(object sender, EventArgs e)

{

}

}

}

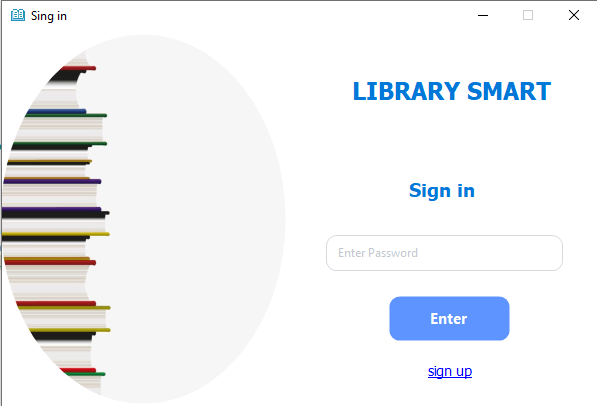


Figure Form 2 (Student)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Smart\_Library

{

public partial class second\_pass\_form\_for\_students : Form

{

smart\_libraryEntities Smart\_Library = new smart\_libraryEntities();

user user = new user();

public second\_pass\_form\_for\_students()

{

InitializeComponent();

}

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

{

char c = e.KeyChar;

if (char.IsDigit(c))

{

e.Handled = false;

}

else if (c == 8)

{

e.Handled = false;

}

else

{

e.Handled = true;

}

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

try

{

int pass = Convert.ToInt32(txtPass.Text);

var password = Smart\_Library.users.Where(b => b.password == pass).Select(b => b.password);

if (password.Contains(pass))

{

main\_student sd = new main\_student();

sd.Show();

this.Hide();

}

else

{

MessageBox.Show("your password wrong");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void linkSignUp\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

signUp sp = new signUp();

sp.Show();

this.Hide();

}

}

}

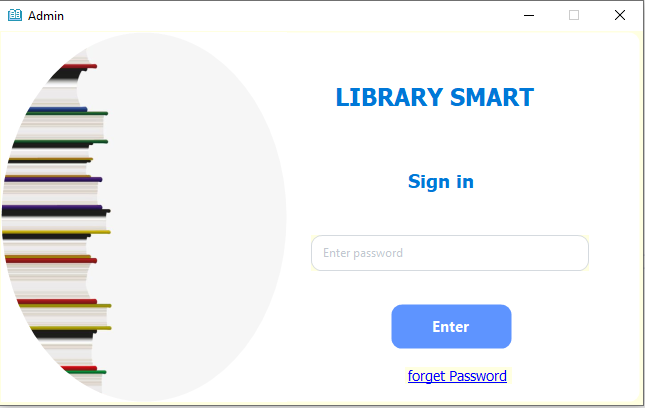


Figure 3 (Admin)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Microsoft.VisualBasic;

namespace Smart\_Library

{

public partial class secondForm : Form

{

smart\_libraryEntities Smart\_Library = new smart\_libraryEntities();

user user = new user();

public secondForm()

{

InitializeComponent();

}

private void secondForm\_Load(object sender, EventArgs e)

{

txtBoxPassword.Focus();

}

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

{

char c = e.KeyChar;

if (char.IsDigit(c))

{

e.Handled = false;

}

else if (c == 8)

{

e.Handled = false;

}

else

{

e.Handled = true;

}

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

try

{

main\_admin md = new main\_admin();

int pass = int.Parse(txtBoxPassword.Text);

var password = Smart\_Library.users.Where(b => b.password == pass).Select(b => b.password);

if (password.Contains(pass))

{

md.Show();

this.Hide();

}

else

{

MessageBox.Show("wrong Password!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void linkSignUP\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

try

{

MessageBox.Show("change password successfully");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void linkFrogetPass\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

int DefaultPass = 2468;

string defaultNumber = Interaction.InputBox("Check Default Number", "InputBox");

if (int.Parse(defaultNumber) == DefaultPass)

{

string newName = Interaction.InputBox("Enter New Name", "InputBox");

string pass = Interaction.InputBox("Enter New Password", "InputBox");

user u = new user

{

id = Smart\_Library.users.Count() + 1,

username = newName,

password = int.Parse(pass),

};

Smart\_Library.users.Add(u);

Smart\_Library.SaveChanges();

MessageBox.Show("successfully");

}

}

}

}

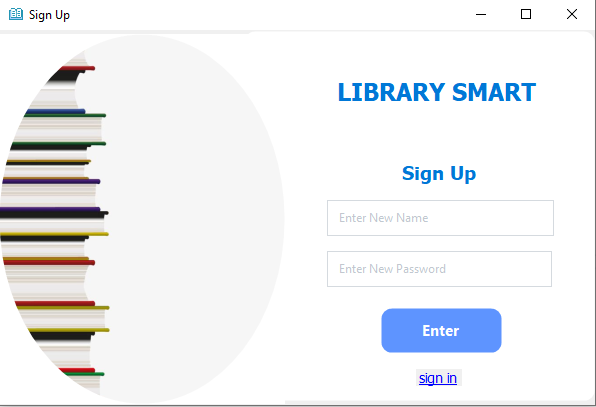


Figure Form 4 (Student)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Smart\_Library

{

public partial class signUp : Form

{

public static string name;

smart\_libraryEntities smart\_Library = new smart\_libraryEntities();

user user = new user();

public signUp()

{

InitializeComponent();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

second\_pass\_form\_for\_students sd = new second\_pass\_form\_for\_students();

sd.Show();

this.Hide();

}

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

{

try

{

int count = smart\_Library.users.Count();

user ur = new user()

{

id = count + 1,

username = txtName.Text,

password = Convert.ToInt32(txtPass.Text)

};

smart\_Library.users.Add(ur);

smart\_Library.SaveChanges();

MessageBox.Show("Add new user successfully");

this.Hide();

second\_pass\_form\_for\_students sd = new second\_pass\_form\_for\_students();

sd.Show();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

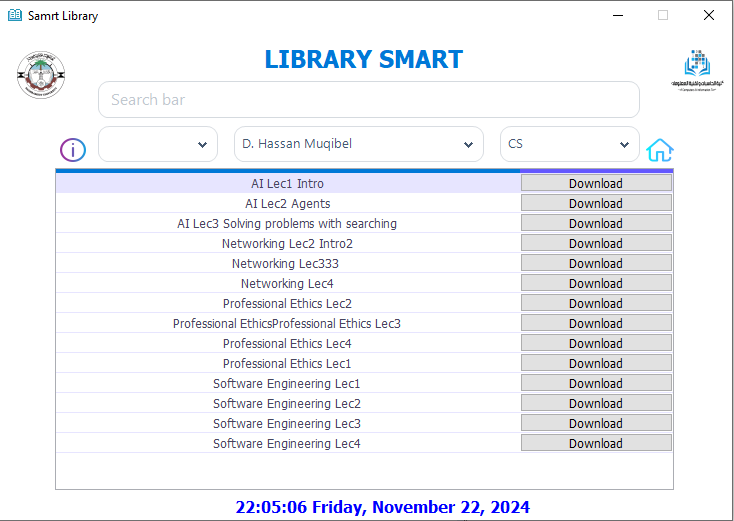


Figure Form 5 (Main Student Page)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.Entity.Migrations;

using System.Globalization; // Add this at the top of your file

using System.Net;

using System.IO;

using System.Net.Http;

namespace Smart\_Library

{

public partial class main\_student : Form

{

smart\_libraryEntities smartLibrary = new smart\_libraryEntities();

info\_books books = new info\_books();

info\_majors majors = new info\_majors();

info\_teachers teachers = new info\_teachers();

public main\_student()

{

InitializeComponent();

var bookNames = smartLibrary.info\_books.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super }).ToList();

dgvDisplayBooksG.DataSource = bookNames;

}

void btnDownload()

{

// Create a button column for downloading

DataGridViewButtonColumn downloadButtonColumn = new DataGridViewButtonColumn();

downloadButtonColumn.HeaderText = "Download";

downloadButtonColumn.Text = "Download"; // Text displayed on the button

downloadButtonColumn.Width = 100;

downloadButtonColumn.UseColumnTextForButtonValue = true; // Use the text for all rows

downloadButtonColumn.DefaultCellStyle.ForeColor = Color.White; // Set the background color for the Download button

dgvDisplayBooksG.Columns.Add(downloadButtonColumn);

}

private void LoadMajors()

{

var majors = smartLibrary.info\_majors.Select(m => m.name\_major).ToList();

cBoxMajordgv.DataSource = majors;

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

private void LoadTeachers()

{

var teachers = smartLibrary.info\_teachers.Select(t => t.name\_of\_teacher).ToList();

cBoxTeachersdgv.DataSource = teachers;

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

void DataLoad()

{

dgvDisplayBooksG.Columns[0].HeaderText = "Name Books";

// Hide the links column

dgvDisplayBooksG.Columns["id"].Visible = false;

dgvDisplayBooksG.Columns["book\_major"].Visible = false;

dgvDisplayBooksG.Columns["level"].Visible = false;

dgvDisplayBooksG.Columns["links"].Visible = false;

dgvDisplayBooksG.Columns["super"].Visible = false;

btnDownload();

// Center-align text in the first two columns

foreach (DataGridViewColumn column in dgvDisplayBooksG.Columns)

{

column.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

// Set row height to create a gap effect

dgvDisplayBooksG.RowTemplate.Height = 50; // Adjust this value as needed

// Set the font size for the DataGridView cells

dgvDisplayBooksG.DefaultCellStyle.Font = new Font(dgvDisplayBooksG.DefaultCellStyle.Font.FontFamily, 9f);

// Set column widths

int totalWidth = dgvDisplayBooksG.Width; // Get the total width of the DataGridView

dgvDisplayBooksG.Columns[1].Width = (int)(totalWidth \* 0.75); // 75% for Name of Book

dgvDisplayBooksG.Columns[5].Width = (int)(totalWidth \* 0.7);

}

private void DownloadBook(string url, string bookTitle) // string username, string password

{

using (var client = new WebClient())

{

try

{

// Set Basic Authentication header

//client.Credentials = new NetworkCredential(username, password);

// Create a safe filename by removing invalid characters

string safeFileName = string.Join("\_", bookTitle.Split(Path.GetInvalidFileNameChars()));

// Specify the local file path where you want to save the downloaded file

string localFilePath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments), $"{safeFileName}.pdf"); // Download the file

//Download file

client.DownloadFile(url, localFilePath);

// message Done

MessageBox.Show("Download completed: " + localFilePath, "Download", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("An error occurred while downloading the book: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void main\_student\_Load(object sender, EventArgs e)

{

DataLoad();

LoadMajors();

LoadTeachers();

dgvDisplayBooksG.CellContentClick += dgvDisplayBooksG\_CellContentClick;

// time

timer1.Start();

}

private void dgvDisplayBooksG\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// Check if the clicked cell is in the Download column

if (e.RowIndex >= 0 && e.ColumnIndex == 2) // Adjust the index if needed

{

// Get the link from the hidden links column

string link = dgvDisplayBooksG.Rows[e.RowIndex].Cells["links"].Value.ToString();

string bookTitle = dgvDisplayBooksG.Rows[e.RowIndex].Cells["name\_of\_book"].Value.ToString();

if (!string.IsNullOrEmpty(link))

{

DownloadBook(link, bookTitle);

}

else

{

MessageBox.Show("No download link available.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

// filtering by teacher

private void FilterBooksByTeacher(string selectedTeacher)

{

int selectedTeacherId = GetTeacherIdFromName(selectedTeacher);

var filteredBooks = smartLibrary.info\_books

.Where(b => b.super == selectedTeacherId)

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

// get teacher id from name

private int GetTeacherIdFromName(string teacherName)

{

var teacher = smartLibrary.info\_teachers.FirstOrDefault(t => t.name\_of\_teacher == teacherName);

return teacher?.id\_teacher ?? 0;

}

// filtering by major

// filtering according majors

private void FilterBooksByMajors(string selectedMajor)

{

int selectedMajorId = GetMajorIdFromName(selectedMajor);

var filteredBooks = smartLibrary.info\_books

.Where(b => b.book\_major == selectedMajorId)

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

// // get major id from name

private int GetMajorIdFromName(string MajorName)

{

var major = smartLibrary.info\_majors.FirstOrDefault(t => t.name\_major == MajorName);

return major?.id\_mejors ?? 0;

}

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

{

if (cBoxMajordgv.SelectedItem != null)

{

string selectedMajor = cBoxMajordgv.SelectedItem.ToString();

FilterBooksByMajors(selectedMajor);

}

}

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

{

string selectedLevel = cBoxleveldgv.Text;

var filteredBooks = smartLibrary.info\_books

.Where(b => b.level == selectedLevel)

.Select(l => new { l.id, l.name\_of\_book, l.book\_major, l.level, l.links, l.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

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

{

if (cBoxTeachersdgv.SelectedItem != null)

{

string selectedTeacher = cBoxTeachersdgv.SelectedItem.ToString();

FilterBooksByTeacher(selectedTeacher);

}

}

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

{

string searchText = txtSearchMainS.Text.ToLower(); // Get the search text

var filteredBooks = smartLibrary.info\_books

.Where(b => b.name\_of\_book.ToLower().Contains(searchText))

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks; // Update the DataGridView

// Set row height to create a gap effect

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

private void RefreshDataGridView()

{

// Retrieve the latest data from the database

var bookNames = smartLibrary.info\_books.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super }).ToList();

// Update the DataGridView

dgvDisplayBooksG.DataSource = bookNames;

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

RefreshDataGridView();

}

private void timer1\_Tick(object sender, EventArgs e)

{

CultureInfo culture = new CultureInfo("en-US");

labTime.Text = $"{DateTime.Now.ToString("HH:mm:ss", culture)} {DateTime.Now.ToString("dddd, MMMM dd, yyyy", culture)}";

labTime.ForeColor = Color.Blue;

}

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

{

LinkLabel linkUniver = new LinkLabel();

linkUniver.Text = "https://hu.edu.ye/";

//

Label labUniver = new Label()

{

Text = "Hadramout Unversity",

Font = new Font("Arial", 20, FontStyle.Bold),

};

MessageBox.Show("\t\t\t"+ labUniver.Text + "\n"+"المكتبة الذكية تعتبر ثورة جديدة في تقديم المصادر العلمية للطلاب و اعضاء الهيئة التدريسية." +

" تسعى المكتبة الالكترونية على تبسيط عملية البحث عن الكتب والمحاضرات على صفحة واحدة بسيطة التصميم،" +

" يتوفر ضمن النظام وسائل البحث المتنوعة للكتب التي تساعد في ايجاد الكتاب او المحاضرة المراده." +

"ونتمنى ان يتم تطوير هذا المشروع في المستقبل لتحقيق الرؤية التاريخية للجامعة" +

" التي قد اوصت بتطوير التعليم و البحث العلمي." +

" :هذه المكتبة أيضًا تساهم بشكل فعال في تعزيز بيئة التعلم الحالية بطريقة تلبي متطلبات الطلاب والأكاديميين بطريقة مبتكرة ومريحة." +

"موقع الجامعة" + linkUniver.Text);

}

}

}

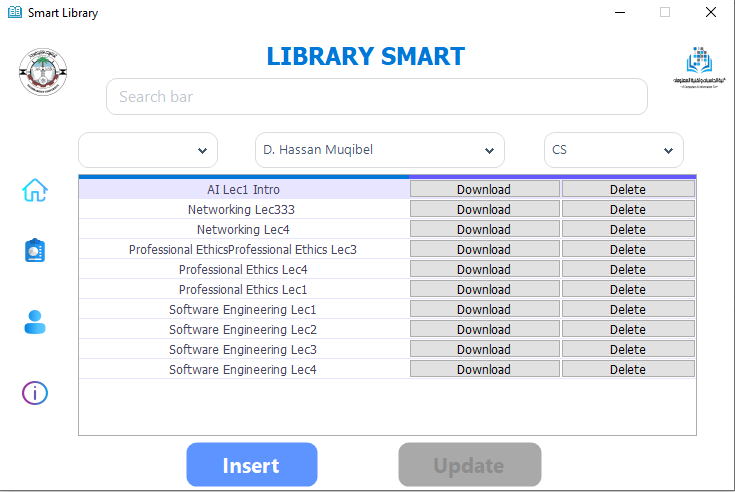


Figure Form 6 (Main Admin Page)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Data.SqlClient;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.Entity.Migrations;

using System.Net;

using System.IO;

using System.Net.Http;

using System.Data.Entity;

using MigraDoc.DocumentObjectModel;

using MigraDoc.Rendering;

using PdfSharp.Pdf;

using MigraDoc.DocumentObjectModel.Fields;

using Color = MigraDoc.DocumentObjectModel.Color;

using MigraDoc.DocumentObjectModel.Tables;

using iTextSharp.text.pdf;

using iTextSharp.text;

using Document = iTextSharp.text.Document;

namespace Smart\_Library

{

public partial class main\_admin : Form

{

smart\_libraryEntities smartLibrary = new smart\_libraryEntities();

info\_books books = new info\_books();

info\_majors majors = new info\_majors();

info\_teachers teachers = new info\_teachers();

user User = new user();

Document Document = new Document();

public static string nameOfTeacher;

public main\_admin()

{

InitializeComponent();

var bookNames = smartLibrary.info\_books.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super }).ToList();

dgvDisplayBooksG.DataSource = bookNames;

dgvDisplayBooksG.CellContentClick += dgvDisplayBooksG\_CellContentClick;

}

private void LoadTeachers()

{

var teachers = smartLibrary.info\_teachers.Select(t => t.name\_of\_teacher).ToList();

cBoxTeachersdgv.DataSource = teachers;

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

// load users

private void LoadUsers()

{

var users = smartLibrary.users.Select(d => new { d.id, d.username }).ToList();

dgvDisplayBooksG.DataSource = users;

// btn to delete user

DataGridViewButtonColumn deleteButtonColumn = new DataGridViewButtonColumn

{

HeaderText = "Delete User",

Text = "Delete",

UseColumnTextForButtonValue = true,

Width = 100

};

deleteButtonColumn.UseColumnTextForButtonValue = true; // Use the text for all rows

dgvDisplayBooksG.Columns.Add(deleteButtonColumn);

}

private void LoadMajors()

{

var majors = smartLibrary.info\_majors.Select(m => m.name\_major).ToList();

cBoxMajordgv.DataSource = majors;

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

void btnDownload()

{

// Create a button column for downloading

DataGridViewButtonColumn downloadButtonColumn = new DataGridViewButtonColumn();

downloadButtonColumn.HeaderText = "Download";

downloadButtonColumn.Text = "Download"; // Text displayed on the button

downloadButtonColumn.Width = 100;

downloadButtonColumn.UseColumnTextForButtonValue = true; // Use the text for all rows

dgvDisplayBooksG.Columns.Add(downloadButtonColumn);

}

private void DownloadBook(int rowIndex)

{

try

{

// Get the book name and download URL from the selected row

string bookName = dgvDisplayBooksG.Rows[rowIndex].Cells["name\_of\_book"].Value?.ToString(); // Ensure null safety

string downloadUrl = dgvDisplayBooksG.Rows[rowIndex].Cells["links"].Value?.ToString();

if (string.IsNullOrWhiteSpace(bookName) || string.IsNullOrWhiteSpace(downloadUrl))

{

MessageBox.Show("Book name or download link is missing.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Convert Google Drive link to direct download link if applicable

if (downloadUrl.Contains("/d/") && downloadUrl.Contains("/view"))

{

string fileId = downloadUrl.Split(new string[] { "/d/", "/view" }, StringSplitOptions.None)[1];

downloadUrl = $"https://drive.google.com/uc?export=download&id={fileId}";

}

// Check if the link is valid

Uri uriResult;

bool isValidUrl = Uri.TryCreate(downloadUrl, UriKind.Absolute, out uriResult) && (uriResult.Scheme == Uri.UriSchemeHttp || uriResult.Scheme == Uri.UriSchemeHttps);

if (!isValidUrl)

{

MessageBox.Show("Invalid download URL.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Set the destination file path in the Downloads folder

string downloadsPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments));

string filePath = Path.Combine(downloadsPath, $"{bookName}.pdf");

// Ensure file does not overwrite

if (File.Exists(filePath))

{

MessageBox.Show($"The file '{filePath}' already exists. Rename the book or delete the existing file.", "File Exists", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Download the file

using (WebClient client = new WebClient())

{

client.DownloadFile(downloadUrl, filePath);

MessageBox.Show($"Book '{bookName}' downloaded successfully to {downloadsPath}.", "Download Complete", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (WebException webEx)

{

MessageBox.Show($"Download failed: {webEx.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (IOException ioEx)

{

MessageBox.Show($"File I/O error: {ioEx.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while downloading the book: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

void btnDelete()

{

// Create a button column for deleting

DataGridViewButtonColumn deleteButtonColumn = new DataGridViewButtonColumn();

deleteButtonColumn.HeaderText = "Delete";

deleteButtonColumn.Text = "Delete"; // Text displayed on the button

deleteButtonColumn.Width = 100;

deleteButtonColumn.UseColumnTextForButtonValue = true; // Use the text for all rows

dgvDisplayBooksG.Columns.Add(deleteButtonColumn);

}

private void DeleteBook(int rowIndex)

{

try

{

// Get the book ID from the selected row

int bookId = (int)dgvDisplayBooksG.Rows[rowIndex].Cells["id"].Value;

// Find the book in the database

var bookToDelete = smartLibrary.info\_books.FirstOrDefault(b => b.id == bookId);

if (bookToDelete != null)

{

// Delete the book from the database

smartLibrary.info\_books.Remove(bookToDelete);

smartLibrary.SaveChanges();

// Refresh the DataGridView

MessageBox.Show("Book deleted successfully.", "Delete", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshDataGridView();

}

else

{

MessageBox.Show("Failed to delete the book. The book was not found.");

RefreshDataGridView();

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while deleting the book: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

void ColFalse()

{

// Hide the links column

dgvDisplayBooksG.Columns["id"].Visible = false;

dgvDisplayBooksG.Columns["book\_major"].Visible = false;

dgvDisplayBooksG.Columns["level"].Visible = false;

dgvDisplayBooksG.Columns["links"].Visible = false;

dgvDisplayBooksG.Columns["super"].Visible = false;

}

void DataLoad()

{

dgvDisplayBooksG.Columns[1].HeaderText = "Name Books";

// Hide the links column

ColFalse();

btnDownload();

btnDelete();

// Center-align text in the first two columns

foreach (DataGridViewColumn column in dgvDisplayBooksG.Columns)

{

column.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

// Set the font size for the DataGridView cells

dgvDisplayBooksG.DefaultCellStyle.Font = new System.Drawing.Font(dgvDisplayBooksG.DefaultCellStyle.Font.FontFamily, 9f);

// set column w

getWidthFromDataGV(0.52, 0.23, 0.24);

}

// set column widths

void getWidthFromDataGV(double sizeNameBook, double sizeBtnDownload, double sizeBtnDelete)

{

int totalWidth = dgvDisplayBooksG.Width; // Get the total width of the DataGridView

dgvDisplayBooksG.Columns[1].Width = (int)(totalWidth \* sizeNameBook); // 55% for Name of Book

dgvDisplayBooksG.Columns[5].Width = (int)(totalWidth \* sizeBtnDownload); // 20% for btnDownloads

dgvDisplayBooksG.Columns[6].Width = (int)(totalWidth \* sizeBtnDelete); // 20% for btnDeletes

}

private void dgvDisplayBooks\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

// main form for admin Load

private void main\_admin\_Load(object sender, EventArgs e)

{

DataLoad();

LoadTeachers();

LoadMajors();

btnUpdatedgv.Enabled = false; // Initially disable the update button

dgvDisplayBooksG.CellContentClick += dgvDisplayBooksG\_CellContentClick;

dgvDisplayBooksG.SelectionChanged += dgvDisplayBooksG\_SelectionChanged; // Attach selection

}

private void dgvDisplayBooksG\_SelectionChanged(object sender, EventArgs e)

{

// Enable the Update button if any row is selected

btnUpdatedgv.Enabled = dgvDisplayBooksG.SelectedRows.Count > 0;

}

// get name teacher through super

private string getNameTeacherThroughSuper(int super)

{

var teacher = smartLibrary.info\_teachers.FirstOrDefault(t => t.id\_teacher == super);

return teacher?.name\_of\_teacher ?? "Unknown";

}

// filtering according teacher

private void FilterBooksByTeacher(string selectedTeacher)

{

int selectedTeacherId = GetTeacherIdFromName(selectedTeacher);

var filteredBooks = smartLibrary.info\_books

.Where(b => b.super == selectedTeacherId)

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

// get teacher id from name

private int GetTeacherIdFromName(string teacherName)

{

var teacher = smartLibrary.info\_teachers.FirstOrDefault(t => t.name\_of\_teacher == teacherName);

return teacher?.id\_teacher ?? 0;

}

// filtering according majors

private void FilterBooksByMajors(string selectedMajor)

{

int selectedMajorId = GetMajorIdFromName(selectedMajor);

var filteredBooks = smartLibrary.info\_books

.Where(b => b.book\_major == selectedMajorId)

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

// // get major id from name

private int GetMajorIdFromName(string MajorName)

{

var major = smartLibrary.info\_majors.FirstOrDefault(t => t.name\_major == MajorName);

return major?.id\_mejors ?? 0;

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

if (dgvDisplayBooksG.Columns.Count > 1)

{

dgvDisplayBooksG.Columns.Clear();

}

RefreshDataGridView();

btnInsertdgv.Enabled = true;

btnUpdatedgv.Enabled = false;

}

private void RefreshDataGridView()

{

if(dgvDisplayBooksG.Columns.Count <= 2)

{

var bookNames = smartLibrary.info\_books.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super }).ToList();

// Update the DataGridView

dgvDisplayBooksG.DataSource = bookNames;

ColFalse();

btnDownload();

btnDelete();

// set column w

getWidthFromDataGV(0.52, 0.23, 0.24);

}

// Retrieve the latest data from the database

}

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

{

string searchText = txtSearchMainA.Text.ToLower(); // Get the search text

var filteredBooks = smartLibrary.info\_books

.Where(b => b.name\_of\_book.ToLower().Contains(searchText))

.Select(b => new { b.id, b.name\_of\_book, b.book\_major, b.level, b.links, b.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks; // Update the DataGridView

// Set row height to create a gap effect

dgvDisplayBooksG.RowTemplate.Height = 20; // Adjust this value as needed

}

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

{

string selectedLevel = cBoxleveldgv.Text;

var filteredBooks = smartLibrary.info\_books

.Where(b => b.level == selectedLevel)

.Select(l => new { l.id, l.name\_of\_book, l.book\_major, l.level, l.links, l.super })

.ToList();

dgvDisplayBooksG.DataSource = filteredBooks;

}

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

{

if (cBoxMajordgv.SelectedItem != null)

{

string selectedMajor = cBoxMajordgv.SelectedItem.ToString();

FilterBooksByMajors(selectedMajor);

}

}

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

{

if (cBoxTeachersdgv.SelectedItem != null)

{

string selectedTeacher = cBoxTeachersdgv.SelectedItem.ToString();

FilterBooksByTeacher(selectedTeacher);

}

}

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

{

var insert = new form\_add\_data\_to\_db();

insert.Show();

}

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

{

if (dgvDisplayBooksG.SelectedRows.Count > 0) // Check if a book is selected

{

string name\_major = "";

var selectedRow = dgvDisplayBooksG.SelectedRows[0];

int bookId = (int)selectedRow.Cells["id"].Value; // Get the book ID

string bookName = selectedRow.Cells["name\_of\_book"].Value.ToString();

int bookMajor = (int)selectedRow.Cells["book\_major"].Value; // Assuming this is an integer

if (bookMajor == 1)

{

name\_major = "CS";

}

else

{

name\_major = "IT";

}

string level = selectedRow.Cells["level"].Value.ToString();

int super = (int)selectedRow.Cells["super"].Value;

string links = selectedRow.Cells["links"].Value.ToString();

string nameTearcher = getNameTeacherThroughSuper(super);

// Create an instance of the update form and pass the selected book's details

var updateForm = new form\_update\_data\_db(bookId, bookName, name\_major, level, nameTearcher, super, links);

updateForm.ShowDialog(); // Show the form modally

}

else

{

MessageBox.Show("Please select a book to update.", "No Selection", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void main\_admin\_Click(object sender, EventArgs e)

{

}

private void dgvDisplayBooksG\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// Check if the clicked cell is in a valid column and row

if (e.RowIndex >= 0 && e.ColumnIndex >= 0 && e.ColumnIndex < dgvDisplayBooksG.Columns.Count)

{

// Check if the clicked cell is in the Download column

if (dgvDisplayBooksG.Columns[e.ColumnIndex].HeaderText == "Download")

{

DownloadBook(e.RowIndex);

}

else if (dgvDisplayBooksG.Columns[e.ColumnIndex].HeaderText == "Delete")

{

// Confirm deletion

var confirmResult = MessageBox.Show("Are you sure you want to delete this book?",

"Confirm Delete",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if (confirmResult == DialogResult.Yes)

{

DeleteBook(e.RowIndex);

}

}

else if(dgvDisplayBooksG.Columns[e.ColumnIndex].HeaderText == "Delete User")

{

var confirmResult = MessageBox.Show("Are you sure you want to delete this User?",

"Confirm Delete",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

if(confirmResult == DialogResult.Yes)

{

DeleteUser(e.RowIndex);

}

}

}

}

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

{

if(dgvDisplayBooksG.Rows.Count > 0)

{

SaveFileDialog save = new SaveFileDialog();

save.Filter = "PDF (\*.pdf)|\*.pdf";

save.FileName = "Result";

bool ErrorMessage = false;

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

{

if (File.Exists(save.FileName))

{

try

{

}

catch (Exception ex)

{

ErrorMessage = true;

MessageBox.Show("Unable to wride data in disk" + ex.Message);

}

}

if (!ErrorMessage)

{

try

{

PdfPTable pTable = new PdfPTable(dgvDisplayBooksG.Columns.Count);

pTable.DefaultCell.Padding = 2;

pTable.WidthPercentage = 100;

pTable.HorizontalAlignment = Element.ALIGN\_LEFT;

foreach(DataGridViewColumn col in dgvDisplayBooksG.Columns)

{

PdfPCell pCell = new PdfPCell(new Phrase(col.HeaderText));

pTable.AddCell(pCell);

}

foreach(DataGridViewRow row in dgvDisplayBooksG.Rows)

{

foreach(DataGridViewCell dcell in row.Cells)

{

pTable.AddCell(dcell.Value.ToString());

}

}

// greate pdf doc

using (FileStream fs = new FileStream(save.FileName, FileMode.Create))

{

string sl = "Smart Library";

Document doc = new Document();

doc.Open();

doc.AddTitle(sl);

doc.Add(pTable);

doc.Close();

fs.Close();

}

MessageBox.Show("Report Export Successfully");

}

catch(Exception ex)

{

MessageBox.Show("Error while Explorting Report" + ex.Message);

}

}

}

}

else

{

MessageBox.Show("We can't Explort Report , Not Found Books");

}

}

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

{

// clear dgv

dgvDisplayBooksG.Columns.Clear();

LoadUsers();

int totalWidth = dgvDisplayBooksG.Width; // Get the total width of the DataGridView

dgvDisplayBooksG.Columns[2].Width = (int)(totalWidth \* 0.2); // 55% for Name of Book

btnInsertdgv.Enabled = false;

btnUpdatedgv.Enabled = false;

}

private void DeleteUser(int rowIndex)

{

try

{

// Get the book ID from the selected row

int bookId = (int)dgvDisplayBooksG.Rows[rowIndex].Cells["id"].Value;

// Find the book in the database

var UserToDelete = smartLibrary.users.FirstOrDefault(b => b.id == bookId);

if (UserToDelete != null)

{

// Delete the book from the database

smartLibrary.users.Remove(UserToDelete);

smartLibrary.SaveChanges();

// Refresh the DataGridView

MessageBox.Show("user deleted successfully.", "Delete", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshDataGridView();

}

else

{

MessageBox.Show("Failed to delete the user. The user was not found.");

RefreshDataGridView();

}

}

catch (Exception ex)

{

MessageBox.Show($"An error occurred while deleting the user: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

LinkLabel linkUniver = new LinkLabel();

linkUniver.Text = "https://hu.edu.ye/";

//

Label labUniver = new Label()

{

Text = "Hadramout Unversity",

Font = new System.Drawing.Font("Arial", 20, FontStyle.Bold),

};

MessageBox.Show("\t\t\t" + labUniver.Text + "\n" + "المكتبة الذكية تعتبر ثورة جديدة في تقديم المصادر العلمية للطلاب و اعضاء الهيئة التدريسية." +

" تسعى المكتبة الالكترونية على تبسيط عملية البحث عن الكتب والمحاضرات على صفحة واحدة بسيطة التصميم،" +

" يتوفر ضمن النظام وسائل البحث المتنوعة للكتب التي تساعد في ايجاد الكتاب او المحاضرة المراده." +

"ونتمنى ان يتم تطوير هذا المشروع في المستقبل لتحقيق الرؤية التاريخية للجامعة" +

" التي قد اوصت بتطوير التعليم و البحث العلمي." +

" :هذه المكتبة أيضًا تساهم بشكل فعال في تعزيز بيئة التعلم الحالية بطريقة تلبي متطلبات الطلاب والأكاديميين بطريقة مبتكرة ومريحة." +

"موقع الجامعة" + linkUniver.Text);

}

}

}

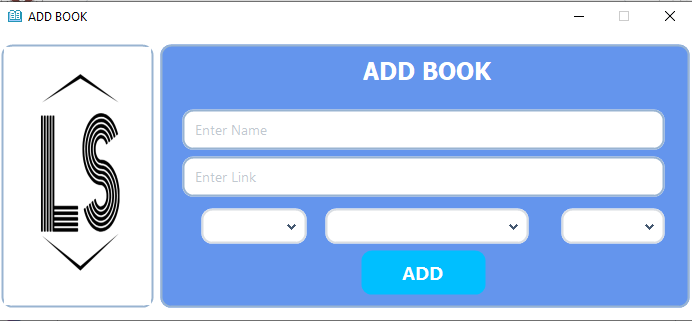


Figure Form 7 Add Book

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Smart\_Library;

namespace Smart\_Library

{

public partial class form\_add\_data\_to\_db : Form

{

smart\_libraryEntities smartLibrary = new smart\_libraryEntities();

info\_books books = new info\_books();

public form\_add\_data\_to\_db()

{

InitializeComponent();

}

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

{

}

int getMajor(int major)

{

string selectedBookMajorId = cBoxMajors.SelectedItem.ToString(); // Cast to int

if (selectedBookMajorId == "CS")

{

major = 1;

}

else if(selectedBookMajorId == "IT")

{

major = 2;

}

else

{

major = 0;

}

return major;

}

// get super on book

private int GetTeacherIdFromName(string teacherName)

{

var teacher = smartLibrary.info\_teachers.FirstOrDefault(t => t.name\_of\_teacher == teacherName);

return teacher?.id\_teacher ?? 0;

}

// get count for id

int getCount(int count)

{

count = smartLibrary.info\_books.Count(b => b.name\_of\_book == count.ToString());

return count;

}

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

{

try

{

// get major

int major = 0;

// count of ID

int totalID = smartLibrary.info\_books.Count();

// Get the selected string value from the ComboBox for level

string selectedLevel = cBoxLevels.SelectedItem?.ToString(); // Get selected level as string

books = new info\_books()

{

id = totalID + 2,

name\_of\_book = txtNameBooks.Text,

book\_major = getMajor(major),

level = selectedLevel,

links = txtlink.Text,

super = GetTeacherIdFromName(cboxTeachers.Text)

};

//smartLibrary.info\_books.Add(book);

//int check = smartLibrary.SaveChanges();

// Add the new book to the database

MessageBox.Show(books.id.ToString());

using (var context = new smart\_libraryEntities())

{

context.info\_books.Add(books);

context.SaveChanges(); // This will insert the new book and auto-generate the ID

}

MessageBox.Show("Book added successfully!");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

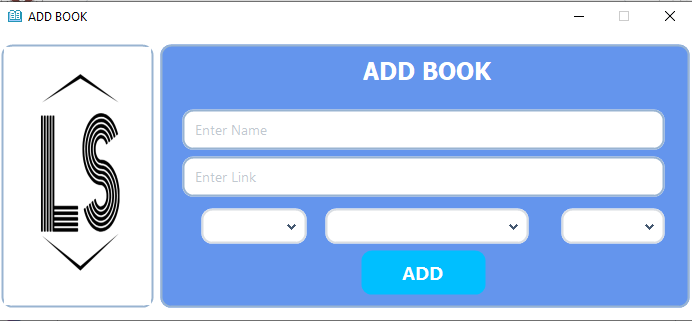


Figure Form 8 (Update Book)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.Entity;

namespace Smart\_Library

{

public partial class form\_update\_data\_db : Form

{

smart\_libraryEntities smartLibrary = new smart\_libraryEntities();

info\_books books = new info\_books();

int bookID;

int IDMajor;

string superr;

public form\_update\_data\_db(int id, string name, string major, string level, string teacher, int super, string links)

{

InitializeComponent();

bookID = id;

txtNameBooks.Text = name; // Assuming you have a TextBox for book name

cBoxMajors.Text = major; // Assuming you have a ComboBox for book major

cboxTeachers.Text = teacher;

cBoxLevels.SelectedItem = level; // Assuming you have a ComboBox for level

txtlink.Text = links; // Assuming you have a TextBox for links

string selectedItem = cboxTeachers.SelectedItem.ToString();

superr = super.ToString();

}

void getTeachersNames()

{

var teachers = smartLibrary.info\_teachers.Select(t => t.name\_of\_teacher);

cboxTeachers.DataSource = teachers;

}

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

{

// Update the book's details

var bookToUpdate = smartLibrary.info\_books.Find(bookID);

if (bookToUpdate != null)

{

if (cBoxMajors.Text == "CS")

{

IDMajor = 1;

}

else

{

IDMajor = 2;

}

bookToUpdate.name\_of\_book = txtNameBooks.Text;

bookToUpdate.book\_major = IDMajor; // Cast to int

bookToUpdate.level = cBoxLevels.SelectedItem.ToString(); // Get selected level as string

bookToUpdate.super = int.Parse(superr);

bookToUpdate.links = txtlink.Text;

smartLibrary.Entry(bookToUpdate).State = EntityState.Modified;

smartLibrary.SaveChanges(); // Save changes to the database

MessageBox.Show("Book updated successfully!", "Update", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close(); // Close the update form

}

else

{

MessageBox.Show("Book not found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}