**Лабораторна робота №12. Створення багатовіконних застосунків. Частина 2.**

**Виконавець:** Лазоренко М.Д.

**Група виконавця:** КН-922в

**Мета роботи:** набути навички у створенні багатовіконних застосунків з більш розвиненим функціоналом.

**Індивідуальні завдання**

**Варіант №9**

**Завдання 1**

Додати до проекту форму ресстрації користувачів умовної системи із використанням реестраційної форми згідно до власного варіанту. Перед занесенням ресстраційних даних до відповідної таблиці, виконати перевірку (валідацію) реестраційних полів:

* UserName : лише числа та символи латиниці кількістю не більше 15 i щоб ім'я було унікальним (раніше не було в таблиці БД);
* Password : числа, символи латиниці та обов'язково спеціальні символи довжиною не менше 12, та його коректне повторення у відповідному полі;
* Email : коректне значения електронної поштової адреси

В разі наявності помилок повертатися до форми і відображати їх навпроти відповідних полів.

|  |  |
| --- | --- |
| № варіанту | Шаблон реєстраційної форми |
| 1 | 2 |
| 9. |  |

Валідацію полів форми зробити за допомогою регулярних виразів.

**Текст програми:**

***RegForm.cs:***

using Lab12;

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab12

{

public partial class RegForm : Form

{

public RegForm()

{

InitializeComponent();

}

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

{

// Очищення попередніх повідомлень про помилки

lblErrorUsername.Text = "";

lblErrorPassword.Text = "";

lblErrorConfirmPassword.Text = "";

lblErrorEmail.Text = "";

// Отримання значень

string username = txtUsername.Text.Trim();

string password = txtPassword.Text.Trim();

string confirmPassword = txtConfirmPassword.Text.Trim();

string email = txtEmail.Text.Trim();

// Логіка валідації

bool isValid = true; // Загальний прапорець для перевірки

// Перевірка Username

if (!ValidateUsername(username))

{

lblErrorUsername.Text = "Некоректне ім'я користувача! (Макс 15 символів, лише літери та цифри)";

isValid = false;

}

// Перевірка чи користувач вже існує

if (UserExists(username))

{

lblErrorUsername.Text = "Користувач з таким ім'ям вже існує!";

isValid = false;

}

// Перевірка Password

if (!ValidatePassword(password))

{

lblErrorPassword.Text = "Пароль має бути 12+ символів, з літерами, цифрами та спецсимволами.";

isValid = false;

}

// Перевірка збігу паролів

if (password != confirmPassword)

{

lblErrorConfirmPassword.Text = "Паролі не співпадають!";

isValid = false;

}

// Перевірка Email

if (!ValidateEmail(email))

{

lblErrorEmail.Text = "Некоректний формат Email!";

isValid = false;

}

// Якщо помилок немає, відобразити успішну реєстрацію

if (isValid)

{

// Зберегти користувача

SaveUser(username, password, email);

MessageBox.Show("Реєстрація успішна!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Перехід до форми авторизації

LogForm loginForm = new LogForm();

loginForm.Show();

this.Close();

}

}

private bool ValidateUsername(string username)

{

// Перевірка: лише латиниця та цифри, довжина до 15 символів

return Regex.IsMatch(username, @"^[a-zA-Z0-9]{1,15}$");

}

private bool ValidatePassword(string password)

{

// Перевірка: довжина не менше 12, містить букви, цифри, спеціальні символи

return Regex.IsMatch(password, @"^(?=.\*[a-zA-Z])(?=.\*\d)(?=.\*[!@#$%^&\*]).{12,}$");

}

private bool ValidateEmail(string email)

{

// Простий регулярний вираз для перевірки email

return Regex.IsMatch(email, @"^\S+@\S+**\.**\S+$");

}

private bool UserExists(string username)

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 1)

{

string storedUsername = data[0];

if (storedUsername == username)

{

return true;

}

}

}

}

return false;

}

private void SaveUser(string username, string password, string email)

{

// Зберегти користувача в users.txt

string usersFile = "users.txt";

string userLine = $"{username};{password};{email}";

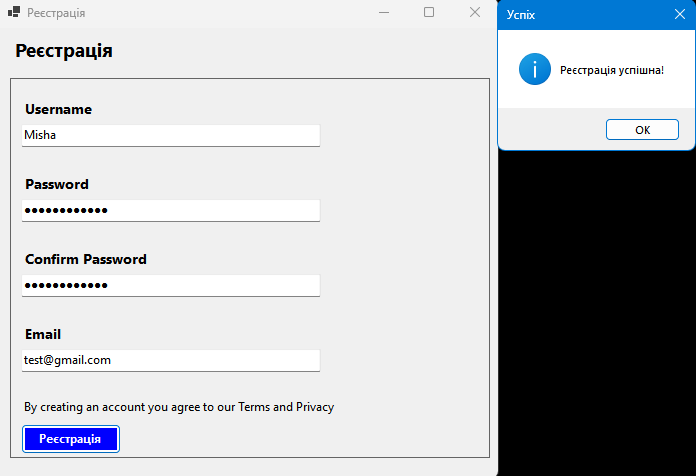
File.AppendAllLines(usersFile, new string[] { userLine });

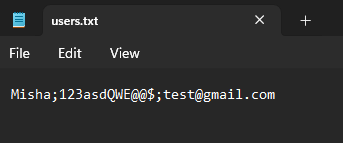
}

}

}

***Результат роботи програми:***

******

******

Програма реалізує форму реєстрації користувача з перевіркою введених даних. Вона складається з класу RegForm, який є частиною Windows Forms програми. Ключові функції програми описані нижче:

1. **Ініціалізація форми:**
   * При створенні об'єкта форми (RegForm) ініціалізуються елементи інтерфейсу, включаючи текстові поля для введення імені користувача, пароля, підтвердження пароля та електронної пошти.
2. **Обробка події натискання кнопки реєстрації (btnRegister\_Click):**
   * Спочатку очищаються всі попередні повідомлення про помилки.
   * Збираються введені дані з текстових полів: ім'я користувача, пароль, підтвердження пароля та email.
3. **Валідація введених даних:**
   * Перевіряється, чи відповідають введені дані вимогам:
     + Ім'я користувача: лише латинські літери та цифри, максимальна довжина — 15 символів.
     + Пароль: має бути не менше 12 символів, містити літери, цифри та спеціальні символи.
     + Підтвердження пароля: має співпадати з введеним паролем.
     + Email: має бути у коректному форматі.
   * Якщо перевірка не пройдена, виводяться відповідні повідомлення про помилки для кожного поля.
4. **Перевірка наявності користувача:**
   * Програма перевіряє, чи вже існує користувач з таким ім'ям у файлі users.txt. Якщо такий користувач вже є, виводиться повідомлення про помилку.
5. **Збереження користувача:**
   * Якщо всі перевірки пройдені успішно, дані користувача зберігаються у файл users.txt у форматі username;password;email.
6. **Успішна реєстрація:**
   * Якщо реєстрація пройшла успішно, виводиться повідомлення про успіх, а потім форма переходить до форми авторизації (LoginForm).

**Завдання 2**

Додати до проекту форму авторизації користувачів умовної системи із використанням дизайну реестраційної форми з минулого завдання відповідного варіанту. Авторизаційна форма має складатися з двох полів: Username та Password, до яких користувач має ввести відповідну інформацію. У випадку коректного вводу система мас підтвердити авторизацію користувача, у протилежному випадку — повернутися до назад форми та видати повідомлення про помилку в позиції згідно до свого варіанта

|  |  |
| --- | --- |
| № варіанту | Відносна локація (сірий відтінок) повідомлень про помилки |
| 1 | 2 |
| 9. |  |

**Текст програми:**

***LogForm.cs:***

using Lab12;

using System;

using System.IO;

using System.Windows.Forms;

namespace Lab12

{

public partial class LogForm : Form

{

public LogForm()

{

InitializeComponent();

}

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

{

// Очищення попередніх повідомлень про помилки

lblErrorMessage.Text = "";

// Отримання значень

string username = txtUsername.Text.Trim();

string password = txtPassword.Text.Trim();

// Перевірка чи заповнені поля

if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))

{

lblErrorMessage.Text = "Будь ласка, заповніть всі поля.";

return;

}

// Перевірка авторизації

if (AuthenticateUser(username, password))

{

MessageBox.Show("Авторизація успішна!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Відкрити профільну форму

ProfileForm profileForm = new ProfileForm(username);

profileForm.Show();

this.Hide();

}

else

{

lblErrorMessage.Text = "Невірний логін або пароль.";

}

}

private bool AuthenticateUser(string username, string password)

{

// Перевірка користувача в файлі users.txt

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 2)

{

string storedUsername = data[0];

string storedPassword = data[1];

if (storedUsername == username && storedPassword == password)

{

return true;

}

}

}

}

return false;

}

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

{

// Відкрити реєстраційну форму

RegForm registerForm = new RegForm();

registerForm.Show();

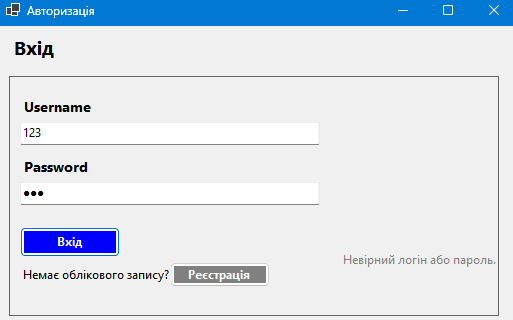
this.Hide();

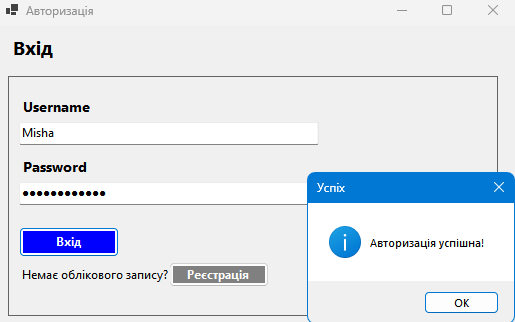
}

}

}

***Результат роботи програми:***





Програма складається з класу LoginForm, який є частиною Windows Forms програми і забезпечує функціонал для авторизації користувача, а також надає можливість переходу до форми реєстрації.

1. **Ініціалізація форми**:
   * При створенні об'єкта форми (LoginForm) ініціалізуються елементи інтерфейсу, включаючи текстові поля для введення імені користувача та пароля.
2. **Обробка події натискання кнопки "Авторизація"** (btnLogin\_Click):
   * Спочатку очищаються всі попередні повідомлення про помилки.
   * Збираються введені дані з текстових полів: ім'я користувача та пароль.
3. **Перевірка заповненості полів**:
   * Програма перевіряє, чи були заповнені поля для введення імені користувача та пароля. Якщо будь-яке з полів не заповнене, виводиться відповідне повідомлення про помилку.
4. **Авторизація користувача**:
   * Виконується перевірка введених даних з тими, що зберігаються у файлі users.txt. Якщо ім'я користувача та пароль збігаються з даними в файлі, користувач вважається авторизованим.
   * Якщо авторизація пройшла успішно, відображається повідомлення про успіх і відкривається нова форма профілю користувача (ProfileForm).
   * Якщо дані невірні, виводиться повідомлення про помилку "Невірний логін або пароль".
5. **Перехід до форми реєстрації**:
   * Користувач має можливість перейти до форми реєстрації, натискаючи кнопку "Реєстрація". У цьому випадку відкривається форма реєстрації (RegForm).

**Завдання 3**

Додати до проекту профільну форму користувача умовної системи із використанням дизайну реестраційної та авторизаційної форм з минулих завдань із розташованими на ній ресстраційними (профільними) параметрами користувача, яка повинна мати можливість змінювати ці параметри. Особливості зміни параметрів визначаються згідно до власного варіанту

|  |  |
| --- | --- |
| № варіанту | Особливості зміни реєстраційних параметрів |
| 1 | 2 |
| 9. | Кожен профільний параметр змінюється на власній окремій формі |

**Текст програми:**

***ProfileForm.cs:***

using Lab12;

using System;

using System.IO;

using System.Windows.Forms;

namespace Lab12

{

public partial class ProfileForm : Form

{

private string username;

public ProfileForm(string username)

{

this.username = username;

InitializeComponent();

lblUsername.Text = username;

// Завантажити дані користувача

LoadUserData();

}

private void LoadUserData()

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 3)

{

string storedUsername = data[0];

if (storedUsername == username)

{

lblEmail.Text = data[2];

return;

}

}

}

}

}

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

{

// Відкрити форму зміни пароля

ChangePasswordForm changePasswordForm = new ChangePasswordForm(username);

changePasswordForm.ShowDialog();

}

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

{

// Відкрити форму зміни Email

ChangeEmailForm changeEmailForm = new ChangeEmailForm(username);

changeEmailForm.ShowDialog();

// Оновити Email на профільній формі після зміни

LoadUserData();

}

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

{

// Повернутися до форми авторизації

LoginForm loginForm = new LoginForm();

loginForm.Show();

this.Close();

}

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

{

// Відкрити форму відправки повідомлення

MessageSendForm messageSendForm = new MessageSendForm(username);

messageSendForm.Show();

}

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

{

// Відкрити форму отримання повідомлень

MessageReceiveForm messageReceiveForm = new MessageReceiveForm(username);

messageReceiveForm.Show();

}

}

}

***ChangeEmailForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab12

{

public partial class ChangeEmailForm : Form

{

private string username;

public ChangeEmailForm(string username)

{

this.username = username;

InitializeComponent();

LoadCurrentEmail();

}

private void LoadCurrentEmail()

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data[0] == username && data.Length >= 3)

{

txtCurrentEmail.Text = data[2];

break;

}

}

}

}

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

{

lblErrorNewEmail.Text = "";

string newEmail = txtNewEmail.Text.Trim();

if (!ValidateEmail(newEmail))

{

lblErrorNewEmail.Text = "Некоректний формат Email!";

return;

}

UpdateEmail(newEmail);

MessageBox.Show("Email успішно змінено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

private bool ValidateEmail(string email)

{

return Regex.IsMatch(email, @"^\S+@\S+**\.**\S+$");

}

private void UpdateEmail(string newEmail)

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

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

{

var data = lines[i].Split(';');

if (data[0] == username && data.Length >= 3)

{

data[2] = newEmail;

lines[i] = string.Join(";", data);

break;

}

}

File.WriteAllLines(usersFile, lines);

}

}

}

}

***ChangePasswordForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab12

{

public partial class ChangePasswordForm : Form

{

private string username;

public ChangePasswordForm(string username)

{

this.username = username;

InitializeComponent();

}

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

{

lblErrorOldPassword.Text = "";

lblErrorNewPassword.Text = "";

lblErrorConfirmPassword.Text = "";

string oldPassword = txtOldPassword.Text.Trim();

string newPassword = txtNewPassword.Text.Trim();

string confirmPassword = txtConfirmPassword.Text.Trim();

if (!ValidateOldPassword(oldPassword))

{

lblErrorOldPassword.Text = "Невірний старий пароль!";

return;

}

if (!ValidatePassword(newPassword))

{

lblErrorNewPassword.Text = "Пароль має бути 12+ символів, з літерами, цифрами та спецсимволами.";

return;

}

if (newPassword != confirmPassword)

{

lblErrorConfirmPassword.Text = "Паролі не співпадають!";

return;

}

// Оновити пароль

UpdatePassword(newPassword);

MessageBox.Show("Пароль успішно змінено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

private bool ValidateOldPassword(string oldPassword)

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 2)

{

if (data[0] == username && data[1] == oldPassword)

{

return true;

}

}

}

}

return false;

}

private bool ValidatePassword(string password)

{

// Така ж перевірка як і раніше

return Regex.IsMatch(password, @"^(?=.\*[a-zA-Z])(?=.\*\d)(?=.\*[!@#$%^&\*]).{12,}$");

}

private void UpdatePassword(string newPassword)

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

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

{

var data = lines[i].Split(';');

if (data[0] == username)

{

data[1] = newPassword;

lines[i] = string.Join(";", data);

break;

}

}

File.WriteAllLines(usersFile, lines);

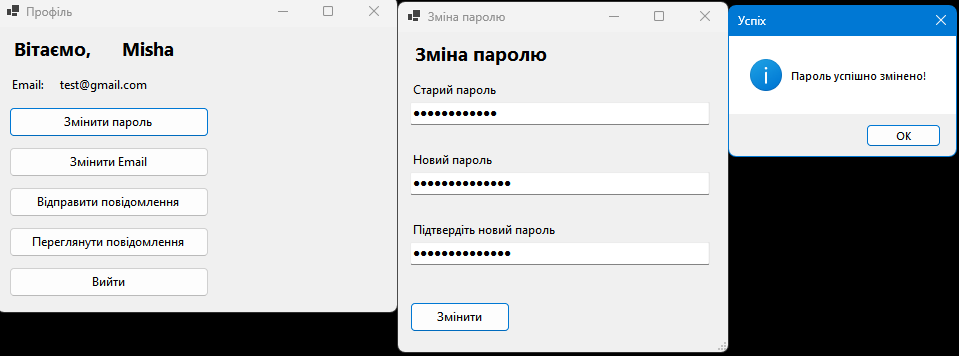
}

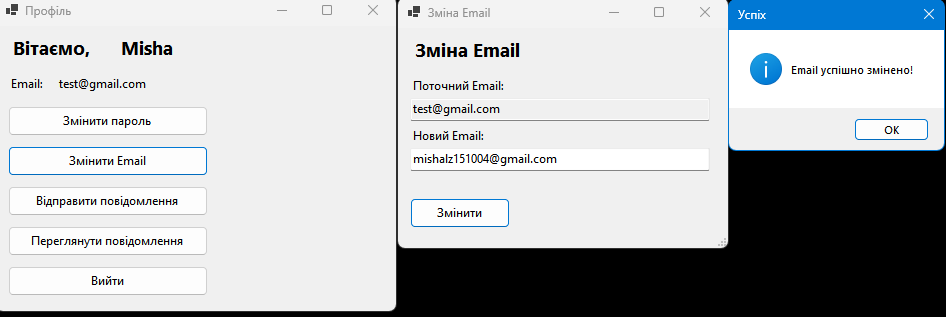
}

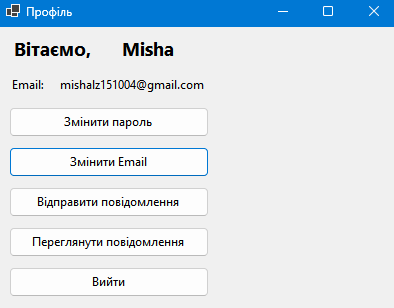
}

}

***Результат роботи програми:***





****

Програма складається з кількох форм, які надають користувачу можливість переглядати та редагувати свій профіль, змінювати дані та взаємодіяти з іншими користувачами через систему повідомлень. Ось короткий опис кожної форми:

**1. ProfileForm**

* Основні функції:
  + Відображення профілю користувача, включаючи ім'я та електронну пошту.
  + Можливість змінити пароль або електронну пошту.
  + Користувач може вийти зі свого профілю та повернутися на форму авторизації.
  + Користувач може відправляти і отримувати повідомлення.
* Ключові методи:
  + LoadUserData(): Завантажує дані користувача з файлу users.txt, зокрема його електронну пошту.
  + btnChangePassword\_Click: Відкриває форму для зміни пароля.
  + btnChangeEmail\_Click: Відкриває форму для зміни електронної пошти.
  + btnLogout\_Click: Повертає користувача до форми авторизації.
  + btnSendMessage\_Click: Відкриває форму для відправлення повідомлень.
  + btnViewMessages\_Click: Відкриває форму для перегляду отриманих повідомлень.

**2. ChangeEmailForm**

* Основні функції:
  + Дозволяє користувачу змінити свою електронну пошту.
  + Перевірка коректності введеного нового email (формат перевіряється за допомогою регулярного виразу).
  + Оновлення email в файлі users.txt.
* Ключові методи:
  + LoadCurrentEmail(): Завантажує поточну електронну пошту користувача з файлу users.txt.
  + btnChangeEmail\_Click: Перевіряє правильність введеного email, оновлює його в файлі, і виводить повідомлення про успіх.

**3. ChangePasswordForm**

* Основні функції:
  + Дозволяє користувачу змінити свій пароль.
  + Перевірка правильності введеного старого пароля.
  + Перевірка нового пароля на відповідність вимогам (мінімум 12 символів, наявність літер, цифр та спецсимволів).
  + Оновлення пароля в файлі users.txt.
* Ключові методи:
  + ValidateOldPassword(): Перевіряє правильність старого пароля, порівнюючи його з даними в файлі users.txt.
  + ValidatePassword(): Перевіряє новий пароль на відповідність вимогам.
  + UpdatePassword(): Оновлює пароль в файлі users.txt.
  + btnChangePassword\_Click: Обробляє натискання кнопки для зміни пароля, перевіряє введені дані, оновлює пароль і виводить повідомлення про успіх.

**Завдання 4**

Додати до проекту форму відправки повідомлень користувачів умовної системи із використанням дизайну форм минулих завдань, яка повинна мати поля ТО для визначення користувача-отримувача повідомлення, THEME та TEXT для визначення теми i тексту повідомлення відповідно. Повідомлення мають зберігатись у відповідному файлі повідомлень для форми отримання цих повідомлень.

**Текст програми:**

***MsgSendForm.cs:***

using System;

using System.IO;

using System.Windows.Forms;

using System.Xml.Linq;

namespace Lab12

{

public partial class MsgSendForm : Form

{

private string senderUsername;

public MsgSendForm(string senderUsername)

{

this.senderUsername = senderUsername;

InitializeComponent();

}

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

{

lblErrorMessage.Text = "";

string recipient = txtTo.Text.Trim();

string theme = txtTheme.Text.Trim();

string text = txtText.Text.Trim();

if (string.IsNullOrEmpty(recipient) || string.IsNullOrEmpty(text))

{

lblErrorMessage.Text = "Будь ласка, заповніть всі обов'язкові поля.";

return;

}

if (!UserExists(recipient))

{

lblErrorMessage.Text = "Користувача не знайдено.";

return;

}

SaveMessage(recipient, theme, text);

MessageBox.Show("Повідомлення надіслано!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

private bool UserExists(string username)

{

string usersFile = "users.txt";

if (File.Exists(usersFile))

{

var lines = File.ReadAllLines(usersFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 1 && data[0] == username)

{

return true;

}

}

}

return false;

}

private void SaveMessage(string recipient, string theme, string text)

{

string messagesFile = "messages.txt";

string messageLine = $"{recipient};{senderUsername};{theme};{text}";

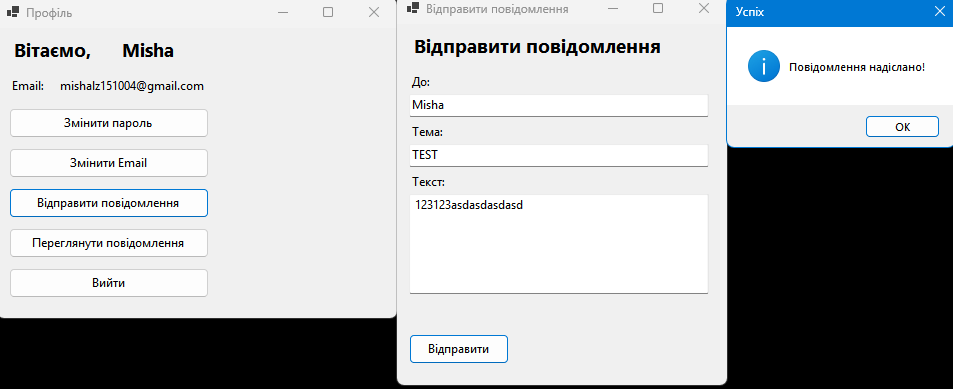
File.AppendAllLines(messagesFile, new string[] { messageLine });

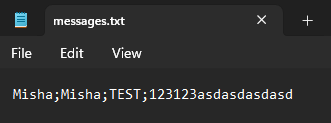
}

}

}

***Результат роботи програми:***

******



Програма реалізує функціональність відправлення повідомлень у вигляді форми, де користувач може вказати отримувача, тему та текст повідомлення. Вона перевіряє правильність введених даних, наявність отримувача в системі, а також зберігає повідомлення у файл.

**Клас MsgSendForm:**

1. **Конструктор:**
   * Конструктор приймає ім'я користувача-відправника і ініціалізує форму.
2. **Метод btnSend\_Click:**
   * Викликається при натисканні кнопки "Відправити".
   * Перевіряє, чи заповнені обов'язкові поля (отримувач та текст повідомлення).
   * Якщо отримувача не знайдено в системі, виводиться повідомлення про помилку.
   * Якщо дані правильні, повідомлення зберігається в файл та виводиться повідомлення про успішне відправлення.
3. **Метод UserExists:**
   * Перевіряє, чи існує користувач з вказаним ім'ям в системі. Для цього програма шукає ім'я користувача в файлі users.txt.
4. **Метод SaveMessage:**
   * Зберігає повідомлення у файл messages.txt у вигляді рядка, який містить отримувача, відправника, тему та текст повідомлення.

**Завдання 5**

Додати до проекту форму отримання повідомлень користувачів умовної системи із використанням дизайну форм минулих завдань, яка повинна мати перелік повідомлень для поточного користувача із можливістю перегляду кожного повідомлення в окремій формі.

**Текст програми:**

***MsgReceiveForm.cs:***

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

using UserMessage = Lab12.Message;

namespace Lab12

{

public partial class MsgReceiveForm : Form

{

private string username;

private List<UserMessage> messages = new List<UserMessage>(); // Ініціалізація списку

public MsgReceiveForm(string username)

{

this.username = username;

InitializeComponent();

LoadMessages();

}

private void LoadMessages()

{

string messagesFile = "messages.txt";

if (File.Exists(messagesFile))

{

var lines = File.ReadAllLines(messagesFile);

foreach (var line in lines)

{

var data = line.Split(';');

if (data.Length >= 4)

{

string recipient = data[0];

string sender = data[1];

string theme = data[2];

string text = data[3];

if (recipient == username)

{

messages.Add(new UserMessage

{

Sender = sender,

Theme = theme,

Text = text

});

}

}

}

}

lstMessages.Items.Clear();

foreach (var message in messages)

{

lstMessages.Items.Add($"{message.Sender}: {message.Theme}");

}

}

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

{

int index = lstMessages.SelectedIndex;

if (index >= 0 && index < messages.Count)

{

MsgDetailsForm messageDetailsForm = new MsgDetailsForm(messages[index]);

messageDetailsForm.ShowDialog();

}

}

}

}

***MsgDetailsForm.cs:***

using System;

using System.Windows.Forms;

using System.Xml.Linq;

namespace Lab12

{

public partial class MsgDetailsForm : Form

{

private Message message;

public MsgDetailsForm(Message message)

{

this.message = message;

InitializeComponent();

lblSender.Text = $"Від: {message.Sender}";

lblTheme.Text = $"Тема: {message.Theme}";

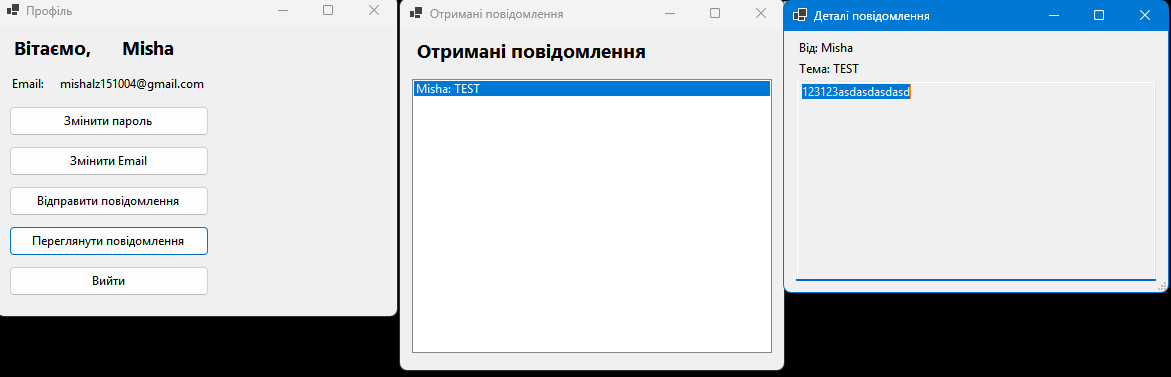
txtText.Text = message.Text;

}

}

}

***Результат роботи програми:***

****

Ці два класи, MsgReceiveForm та MsgDetailsForm, разом реалізують інтерфейс для отримання та перегляду повідомлень у програмі. Ось детальний опис кожної частини:

**1. Клас MsgReceiveForm**

Цей клас відповідає за відображення списку отриманих повідомлень для конкретного користувача.

**Основні компоненти:**

* **username**: Ім'я користувача, для якого відображаються повідомлення.
* **messages**: Список повідомлень, що зберігаються у вигляді об'єктів класу UserMessage.

**Методи:**

1. **MsgReceiveForm(string username)**:
   * Конструктор, який приймає ім'я користувача та ініціалізує форму. Після цього викликається метод LoadMessages() для завантаження повідомлень.
2. **LoadMessages()**:
   * Завантажує повідомлення з файлу messages.txt. Для кожного повідомлення перевіряється, чи є отримувачем вказаний користувач.
   * Якщо користувач є отримувачем, повідомлення додається в список messages.
   * Після завантаження повідомлень, список повідомлень виводиться в елемент lstMessages, відображаючи відправника та тему повідомлення.
3. **lstMessages\_SelectedIndexChanged**:
   * Цей метод викликається, коли користувач вибирає повідомлення зі списку. Він відкриває нову форму MsgDetailsForm, що містить повні деталі обраного повідомлення.

**2. Клас MsgDetailsForm**

Ця форма відображає деталі обраного повідомлення, включаючи відправника, тему та текст повідомлення.

**Основні компоненти:**

* **message**: Об'єкт повідомлення, який містить всі необхідні дані (відправник, тема, текст).

**Методи:**

1. **MsgDetailsForm(Message message)**:
   * Конструктор, що приймає об'єкт повідомлення та ініціалізує форму.
   * Встановлює значення елементів на формі: виводить відправника, тему та текст повідомлення у відповідні компоненти на формі (наприклад, lblSender, lblTheme, txtText).

**Завдання 6**

Додати до проекту форму заставку із логотипом i назвою умовної системи, з’являється на початку функціонування цієї системи.

***Текст програми:***

using System;

using System.Drawing;

using System.IO;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab12

{

public partial class SplashScreenForm : Form

{

public SplashScreenForm()

{

InitializeComponent();

this.Load += SplashScreenForm\_Load;

}

private async void SplashScreenForm\_Load(object? sender, EventArgs e)

{

// Конвертація byte[] у Image

try

{

byte[] imageBytes = Properties.Resources.pic; // Замініть на правильну назву ресурсу

using (MemoryStream ms = new MemoryStream(imageBytes))

{

this.pictureBoxLogo.Image = Image.FromStream(ms);

}

}

catch (Exception ex)

{

MessageBox.Show($"Не вдалося завантажити зображення: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

await Task.Delay(3000); // Затримка 3 секунди

LoginForm loginForm = new LoginForm();

loginForm.Show();

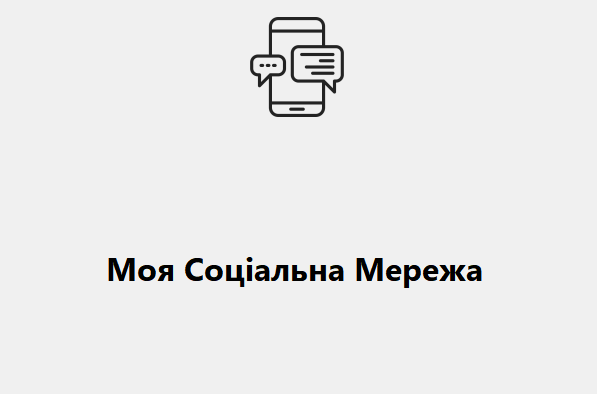
this.Hide();

}

}

}

***Результат роботи програми:***



Цей клас реалізує форму стартового екрану (Splash Screen), яка відображається при запуску програми і містить логотип або інше зображення, а також затримку перед переходом до основної форми (в даному випадку — форми входу).

**Основні компоненти:**

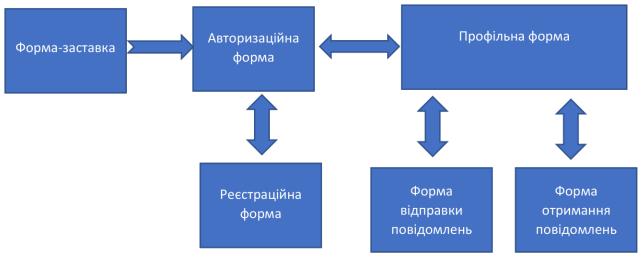
1. **pictureBoxLogo** — елемент управління для відображення зображення на стартовому екрані.
2. **SplashScreenForm\_Load** — подія, що обробляє завантаження форми, в якій виконується завантаження зображення та встановлення його в pictureBoxLogo.

**Основні методи:**

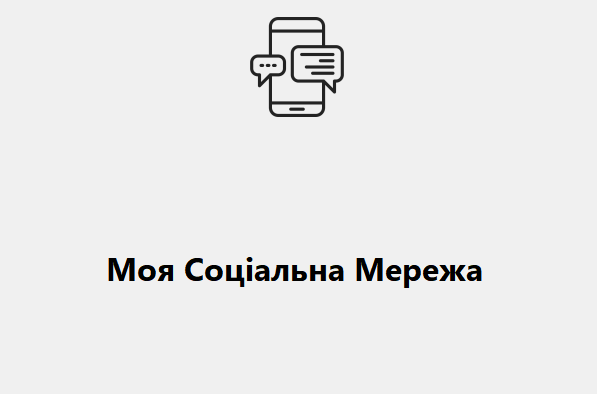
1. **Конструктор SplashScreenForm()**:
   * Ініціалізує форму та підключає обробник події завантаження форми.
2. **Метод SplashScreenForm\_Load(object? sender, EventArgs e)**:
   * Цей метод викликається при завантаженні форми.
   * Спочатку намагається завантажити зображення з ресурсів програми (масив байтів Properties.Resources.pic).
   * Зображення конвертується з формату byte[] у об'єкт Image за допомогою потоку пам'яті (MemoryStream).
   * Якщо виникає помилка при завантаженні зображення, виводиться повідомлення про помилку.
   * Після завантаження зображення формується затримка на 3 секунди за допомогою Task.Delay(3000), щоб користувач міг побачити логотип на екрані.
   * Після затримки відкривається форма входу LoginForm, а стартова форма (SplashScreen) ховається.

**Завдання 7**

Об'єднати зроблені форми за сценарієм функціонування умовної системи, який характеризусться наступною схемою.

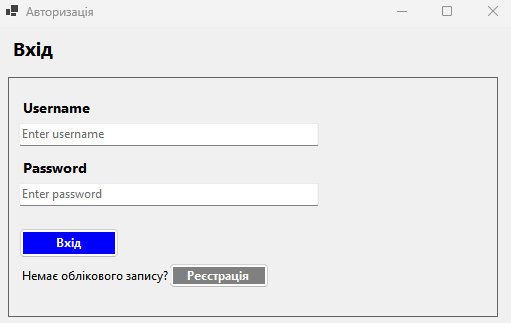


***Результат:***



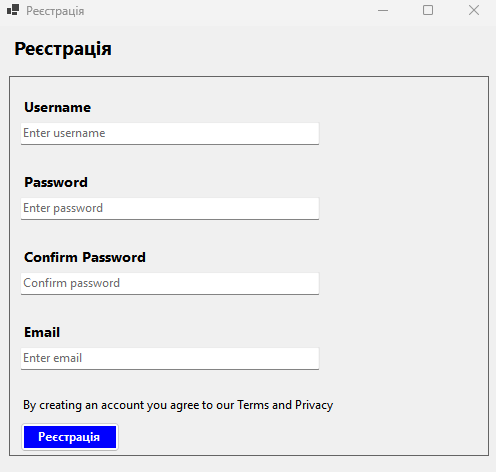
****

****

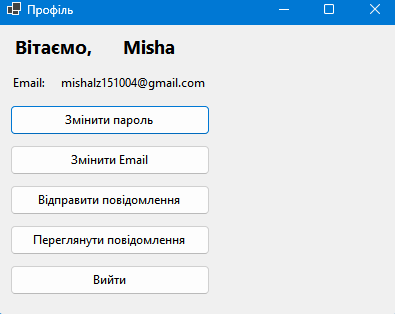
****

****

****

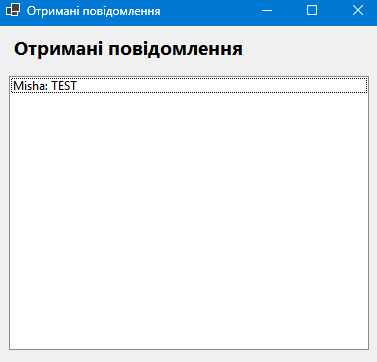
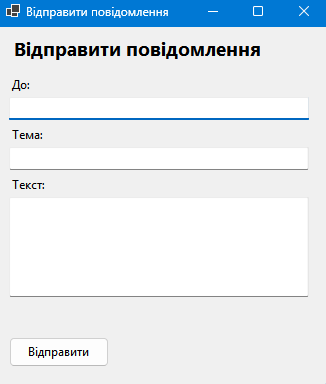
****

****

****

****

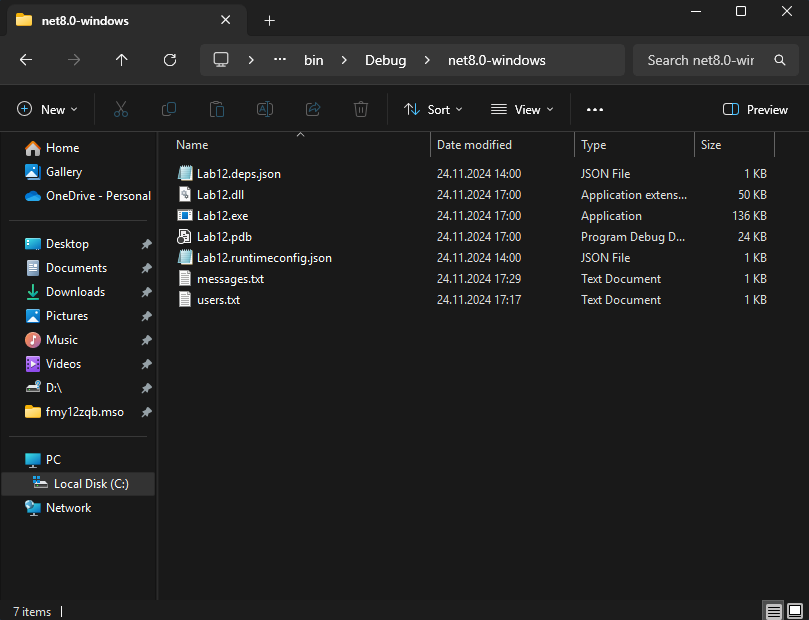
****

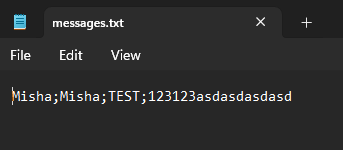
****

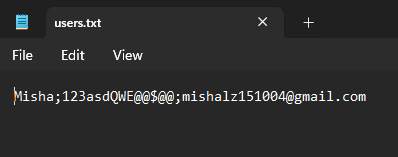
**Завдання 8**

Уся інформація, що потрібна для функціонування системи мас зберігатися в текстових aбo в двійкових файлах за бажанням розробника.

***Результат:***







**Висновок**

Під час виконання лабораторної роботи №12 було набуто навичок створення багатовіконних застосунків з розширеним функціоналом. У процесі роботи було реалізовано функції для відправлення та отримання повідомлень, організовано збереження даних у файлах, а також розроблено стартовий екран із завантаженням зображення. Робота дозволила закріпити навички роботи з графічним інтерфейсом, файлами та ресурсами програми. Завдання виконано повністю, і поставлена мета досягнута.