Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 3

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Варіант - 17

Виконав
Студент групи ІА-31: Перевірив:
Лисенко В. Є Мягкий М. Ю.

Київ 2025

Зміст

1. **Мета:**

Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

2. **Хід роботи:**

Мій варіант:

17. **System activity monitor** (iterator, command, abstract factory, bridge, visitor,

SOA) Монітор активності системи повинен зберігати і запам'ятовувати статистику використовуваних компонентів системи, включаючи навантаження на процесор, обсяг займаної оперативної пам'яті, натискання клавіш на клавіатурі, дії миші (переміщення, натискання), відкриття вікон і зміна вікон; будувати звіти про використання комп'ютера за різними критеріями (% часу перебування у веб-браузері, середнє навантаження на процесор по годинах, середній час роботи комп'ютера по днях і т.д.); правильно поводитися з «простоюванням» системи – відсутністю користувача.

1) Ознайомитись з короткими теоретичними відомостями.
2) Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу

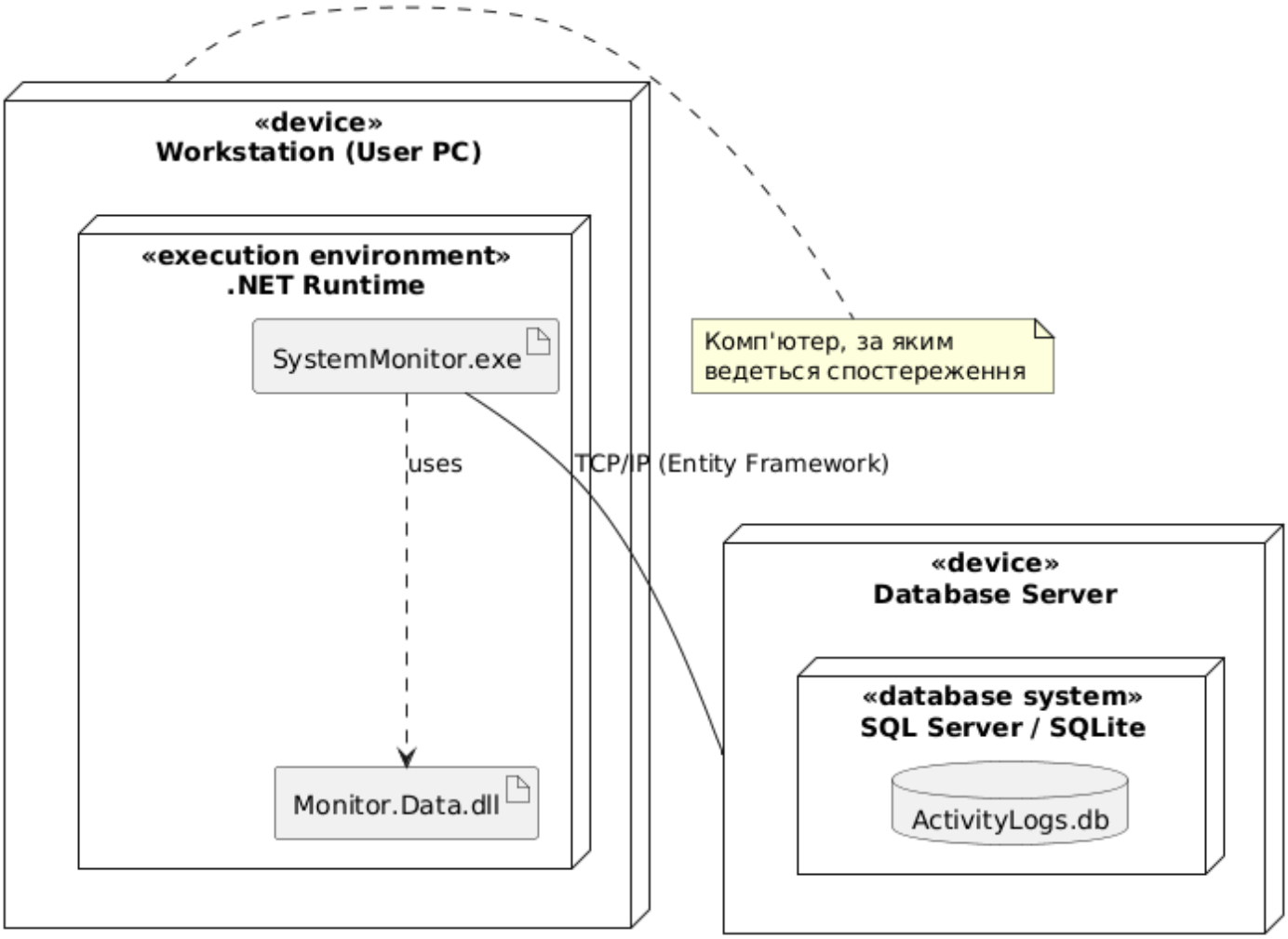

Рис.1 – Діаграма розгортання

На діаграмі розгортання (Рис. 1) зображено фізичну архітектуру системи "System Activity Monitor". Система реалізована як Desktop-застосунок (Rich Client), що розгортається безпосередньо на робочій станції користувача (Workstation).

- Клієнтський вузол: Персональний комп'ютер під керуванням Windows OS, де запускається виконуваний файл SystemMonitor.exe у середовищі .NET Runtime.
- Вузол даних: Система керування базами даних (наприклад, MS SQL Server або локальний файл SQLite), що зберігає історію активності (ActivityLogs). Взаємодія між додатком та БД відбувається через ORM Entity Framework.

3) Розробити діаграму компонентів для проєктованої системи

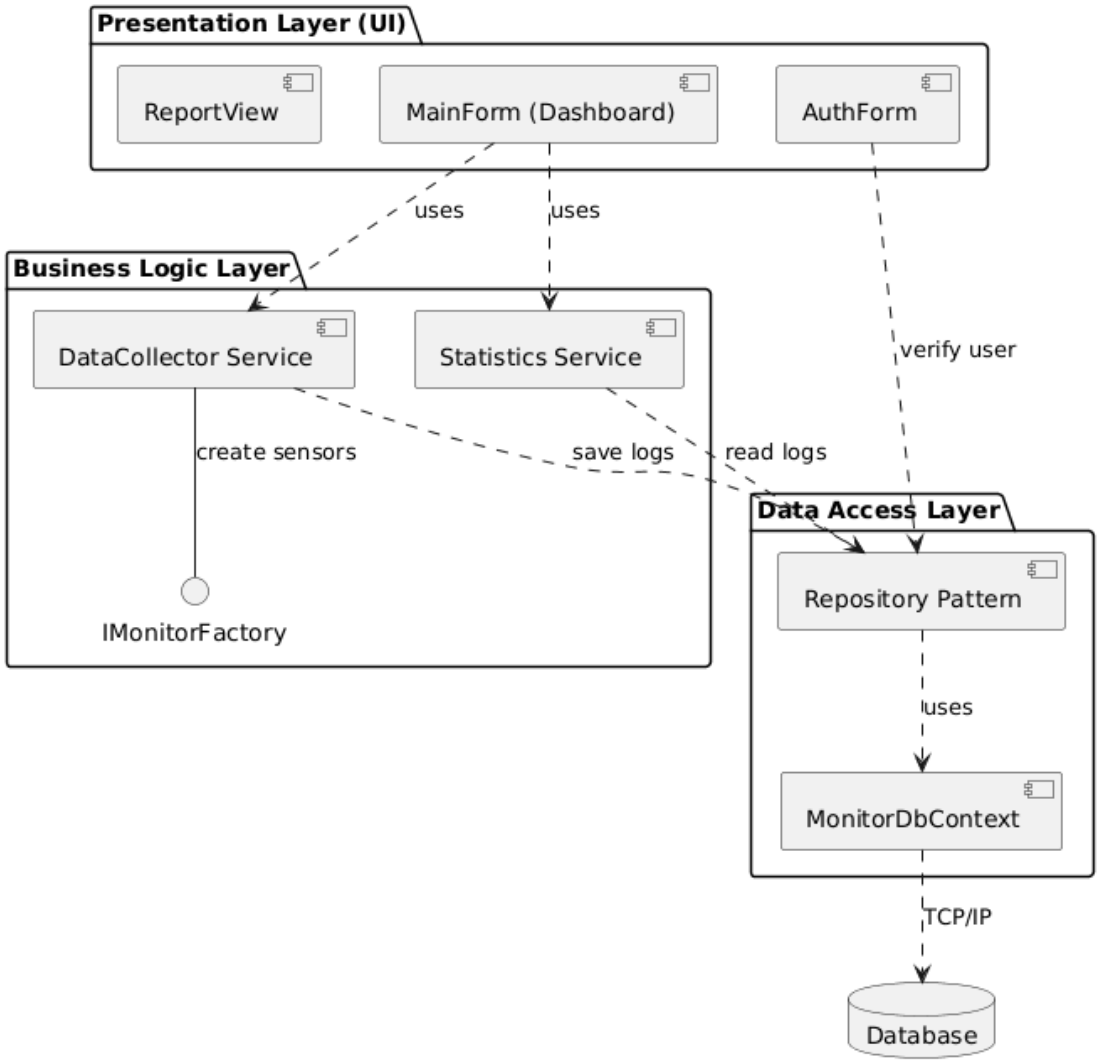

Рис.2 – Діаграма компонентів

На діаграмі компонентів (рис.2) зображено логічну трирівневу архітектуру системи:

1. Presentation Layer: Відповідає за взаємодію з користувачем (форми авторизації, дашборд моніторингу, перегляд звітів).
2. Business Logic Layer: Містить сервіс збору даних DataCollector (який опитує сенсори через фабрику) та сервіс аналітики Statistics Service.
3. Data Access Layer: Реалізує патерн Repository для абстрагування роботи з базою даних через MonitorDbContext.

4) Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі

Табл. 1 Сценарій запис показників ресурсів

Елемент	Опис
Передумови	Моніторинг запущено, таймер спрацював (тік).
Постумови	У базі даних створено новий запис про навантаження.
Взаємодіючі сторони	Системний Таймер, Модуль збору даних.
Короткий опис	Система зчитує поточний стан обладнання та зберігає його.
Основний потік	1. Таймер ініціює подію оновлення. 2. Система запитує відсоток завантаження CPU. 3. Система запитує обсяг вільної RAM. 4. Дані формуються в об'єкт ResourceSnapshot. 5. Об'єкт передається в репозиторій для збереження.
Винятки	Помилка доступу до лічильників продуктивності (запис в лог помилок).

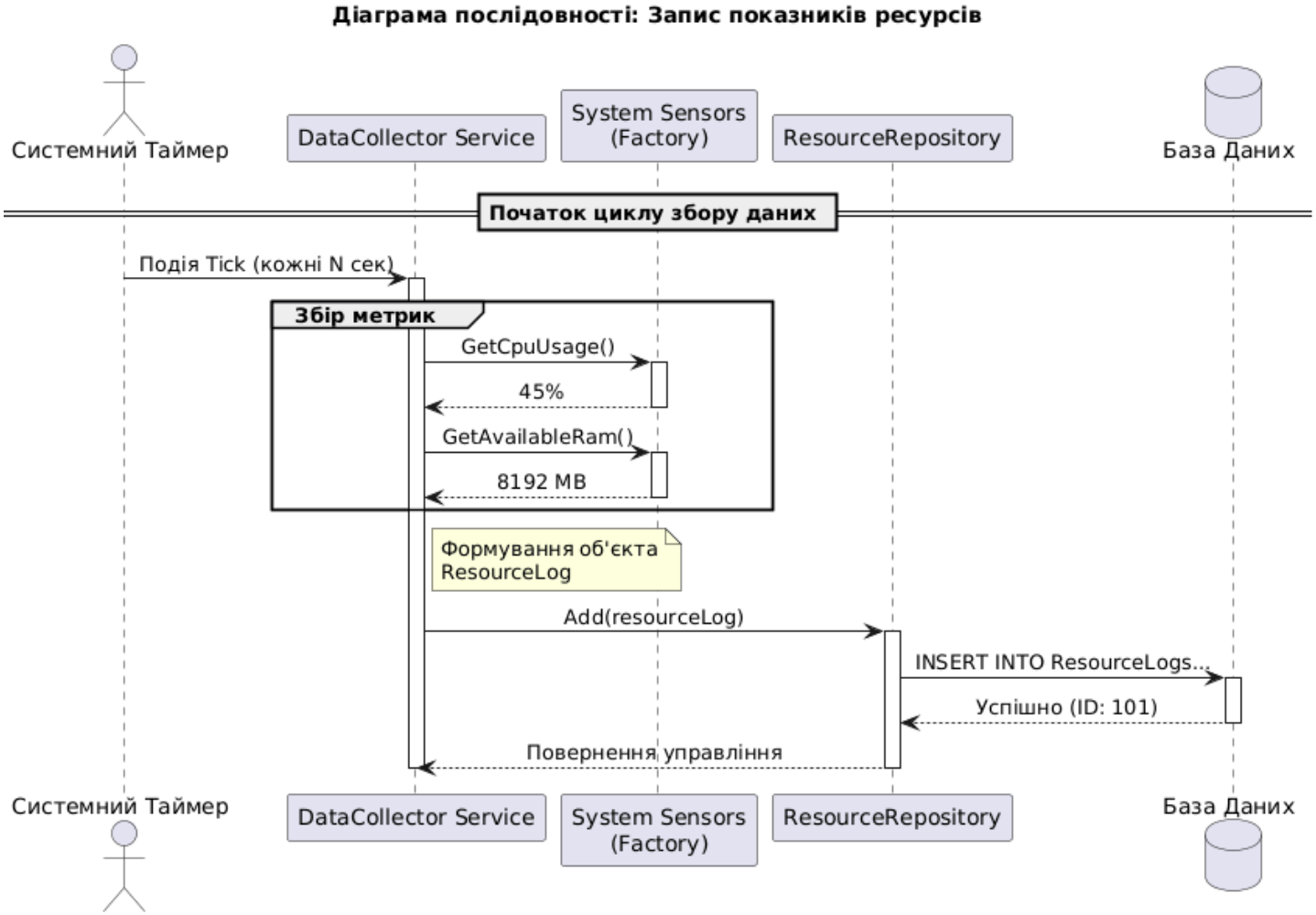

Рис. 3. Діаграма послідовності «Запис показників ресурсів»

Табл. 2 Сценарій визначення простою (Idle State)

Елемент	Опис
Передумови	Моніторинг активний.
Постумови	Статус сесії змінюється на "Active" або "Idle".
Взаємодіючі сторони	Користувач, Input Monitor.
Короткий опис	Система перевіряє, як довго не було дій мишею чи клавіатурою.
Основний потік	1. Система слухає глобальні хуки (hooks) клавіатури/миші. 2. Якщо подія сталася -> скинути таймер простою. 3. Якщо таймер перевищив поріг (наприклад, 5 хв) -> створити подію IdleStarted. 4. При наступній дії -> створити подію IdleEnded.

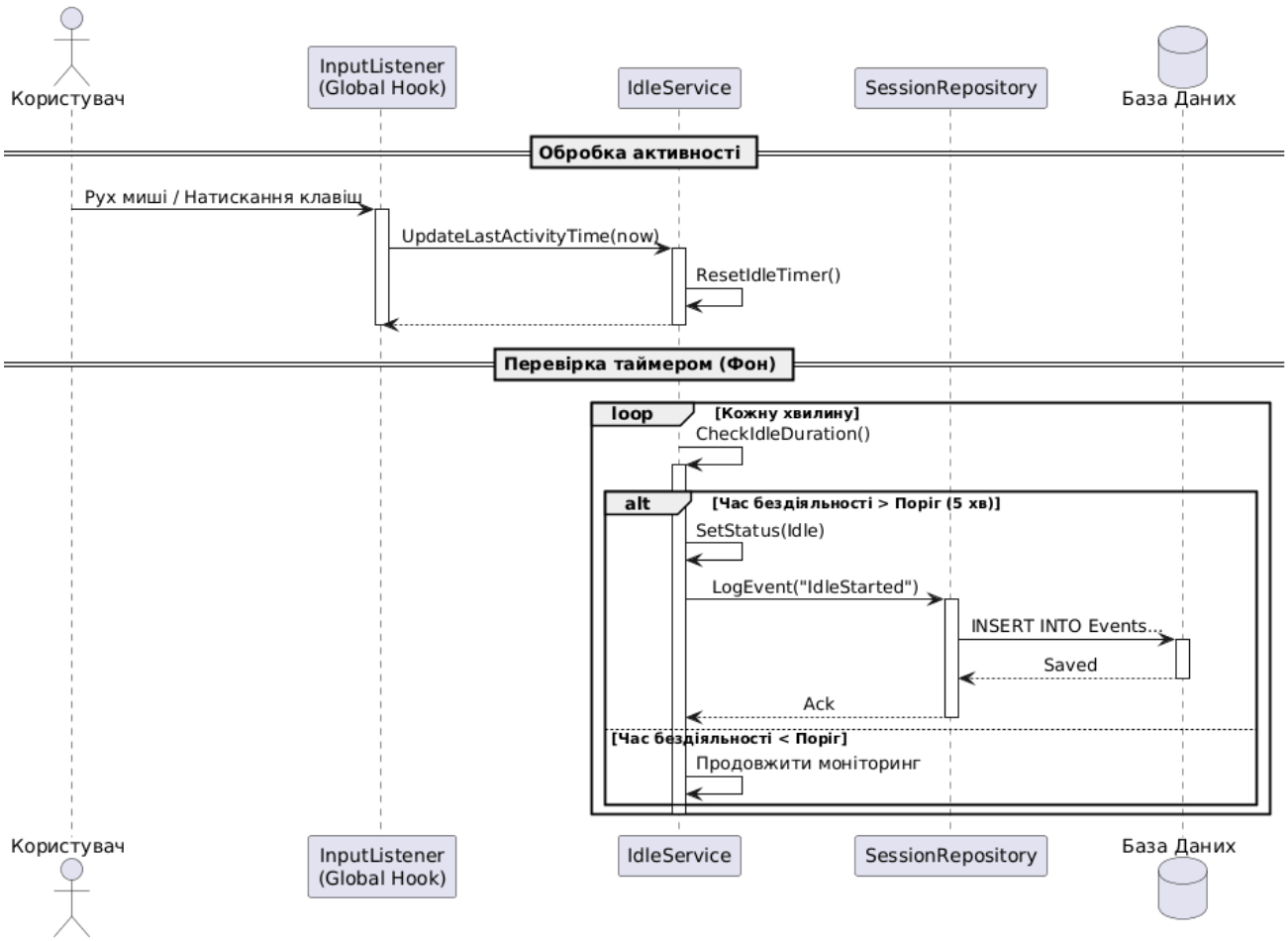

Рис. 4 Діаграма послідовності «Визначення простою»

5) На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

Програмна реалізація

MonitorDbContext.cs:

```
using Microsoft.EntityFrameworkCore;
using SystemActivityMonitor.Data.Entities;
using System.Security.Cryptography;
using System.Text;
namespace SystemActivityMonitor.Data
{
    public class MonitorDbContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<Session> Sessions { get; set; }
        public DbSet<ResourceLog> ResourceLogs { get; set; }

        private readonly string _dbPath = "system_monitor.db";

        protected override void OnConfiguring(DbContextOptionsBuilder options)
        {
            options.UseSqlite($"Data Source={_dbPath}");
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>().HasData(
                new User
                {
                    Id = Guid.Parse("11111111-1111-1111-1111-111111111111"),
                    Username = "admin",
                    PasswordHash = HashPassword("admin123"),
                    Role = "Admin",
                    CreatedAt = DateTime.UtcNow
                },
                new User
                {
                    Id = Guid.Parse("22222222-2222-2222-2222-222222222222"),
                    Username = "user",
                    PasswordHash = HashPassword("user123"),
```

```
                    Role = "User",
                    CreatedAt = DateTime.UtcNow
                }
            );
        }
        public static string HashPassword(string password)
        {
            using (var sha256 = SHA256.Create())
            {
                var bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
                var builder = new StringBuilder();
                foreach (var b in bytes)
                {
                    builder.Append(b.ToString("x2"));
                }
                return builder.ToString();
            }
        }
    }
}
```

LoginWindow.xaml

```
<Window x:Class="SystemActivityMonitor.UI.LoginWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="System Monitor - Вхід" Height="300" Width="350"
        WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
    <Grid Margin="20">
        <StackPanel VerticalAlignment="Center">
            <TextBlock Text="Вхід у систему" FontSize="20" FontWeight="Bold"
                       HorizontalAlignment="Center" Margin="0,0,0,20"/>

            <TextBlock Text="Логін:" Margin="0,5,0,0"/>
            <TextBox x:Name="txtUsername" Height="25" Margin="0,5,0,10"/>

            <TextBlock Text="Пароль:" Margin="0,5,0,0"/>
            <PasswordBox x:Name="txtPassword" Height="25" Margin="0,5,0,20"/>

            <Button Content="Увійти" Height="35" Click="BtnLogin_Click" Background="#007ACC" Foreground="White"
FontWeight="Bold"/>

            <TextBlock HorizontalAlignment="Center" Margin="0,15,0,0">
                <Hyperlink Click="BtnGoToRegister_Click">
                    Немає акаунту? Зареєструватися
```

```
                </Hyperlink>
            </TextBlock>

            <TextBlock x:Name="lblStatus" Text="" Foreground="Red"
                       HorizontalAlignment="Center" Margin="0,10,0,0"/>
        </StackPanel>
    </Grid>
</Window>
```

LoginWindow.xaml.cs

```
using System.Linq;
using System.Windows;
using SystemActivityMonitor.Data;

namespace SystemActivityMonitor.UI
{
    public partial class LoginWindow : Window
    {
        public LoginWindow()
        {
            InitializeComponent();

            using (var db = new MonitorDbContext())
            {
                db.Database.EnsureCreated();
            }
        }

        private void BtnLogin_Click(object sender, RoutedEventArgs e)
        {
            string username = txtUsername.Text;
            string password = txtPassword.Password;

            if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))
            {
                lblStatus.Text = "Введіть логін та пароль!";
                return;
            }

            string inputHash = MonitorDbContext.HashPassword(password);

            using (var db = new MonitorDbContext())
            {
                var user = db.Users.FirstOrDefault(u => u.Username == username && u.PasswordHash == inputHash);
```

```
                if (user != null)
                {
                    MainWindow main = new MainWindow(user.Username, user.Role);
                    main.Show();
                    this.Close();
                }
                else
                {
                    lblStatus.Text = "Невірний логін або пароль";
                }
            }
        }
        private void BtnGoToRegister_Click(object sender, RoutedEventArgs e)
        {
            RegisterWindow reg = new RegisterWindow();
            reg.Show();
            this.Close();
        }
    }
}
```

MainWindow.cs

```
<Window x:Class="SystemActivityMonitor.UI.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="System Activity Monitor - Головна" Height="450" Width="800" WindowStartupLocation="CenterScreen">
    <Grid>
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
            <TextBlock x:Name="txtWelcome" Text="Вітаємо в системі!" FontSize="24" FontWeight="Bold"
HorizontalAlignment="Center"/>
            <TextBlock x:Name="txtRole" Text="Ваша роль: ..." FontSize="16" Foreground="Gray" HorizontalAlignment="Center"
Margin="0,10,0,0"/>

            <TextBlock Text="Тут скоро будуть графіки активності..." Margin="0,50,0,0" FontStyle="Italic"/>
        </StackPanel>
    </Grid>
</Window>
```

MainWindow.xaml.cs:

```
using System.Windows;
namespace SystemActivityMonitor.UI
{
```

```
    public partial class MainWindow : Window
    {
        public MainWindow(string username, string role)
        {
            InitializeComponent();
            txtWelcome.Text = $"Вітаємо, {username}!";
            txtRole.Text = $"Рівень доступу: {role}";
        }
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

RegisterWindow.xaml

```
<Window x:Class="SystemActivityMonitor.UI.RegisterWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Реєстрація" Height="350" Width="350" WindowStartupLocation="CenterScreen">
    <Grid Margin="20">
        <StackPanel VerticalAlignment="Center">
            <TextBlock Text="Новий користувач" FontSize="20" FontWeight="Bold" HorizontalAlignment="Center" Margin="0,0,0,20"/>
            <TextBlock Text="Придумайте логін:"/>
            <TextBox x:Name="txtRegUsername" Height="25" Margin="0,5,0,10"/>
            <TextBlock Text="Придумайте пароль:"/>
            <PasswordBox x:Name="txtRegPassword" Height="25" Margin="0,5,0,20"/>
            <Button Content="Створити акаунт" Height="35" Click="BtnRegister_Click" Background="#28a745" Foreground="White"
FontWeight="Bold"/>
            <Button Content="Скасувати" Margin="0,10,0,0" Click="BtnCancel_Click" Background="Transparent" BorderThickness="0"
Foreground="Gray"/>
        </StackPanel>
    </Grid>
</Window>
```

RegisterWindow.xaml.cs

```
using System;
using System.Linq;
using System.Windows;
using SystemActivityMonitor.Data;
using SystemActivityMonitor.Data.Entities;

namespace SystemActivityMonitor.UI
```

```
{
    public partial class RegisterWindow : Window
    {
        public RegisterWindow()
        {
            InitializeComponent();
        }

        private void BtnRegister_Click(object sender, RoutedEventArgs e)
        {
            string username = txtRegUsername.Text;
            string password = txtRegPassword.Password;

            if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))
            {
                MessageBox.Show("Будь ласка, заповніть всі поля!");
                return;
            }

            using (var db = new MonitorDbContext())
            {
                if (db.Users.Any(u => u.Username == username))
                {
                    MessageBox.Show("Такий користувач вже існує!");
                    return;
                }

                var newUser = new User
                {
                    Username = username,
                    PasswordHash = MonitorDbContext.HashPassword(password),
                    Role = "User",
                    CreatedAt = DateTime.UtcNow
                };

                db.Users.Add(newUser);
                db.SaveChanges();
            }

            MessageBox.Show("Акаунт успішно створено!");

            LoginWindow login = new LoginWindow();
            login.Show();
            this.Close();
```

```
        }

        private void BtnCancel_Click(object sender, RoutedEventArgs e)
        {
            LoginWindow login = new LoginWindow();
            login.Show();
            this.Close();
        }
    }
}
```

App.xaml:

```
<Application x:Class="SystemActivityMonitor.UI.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:SystemActivityMonitor.UI"
             StartupUri="LoginWindow.xaml">
    <Application.Resources>
    </Application.Resources>
</Application>
```

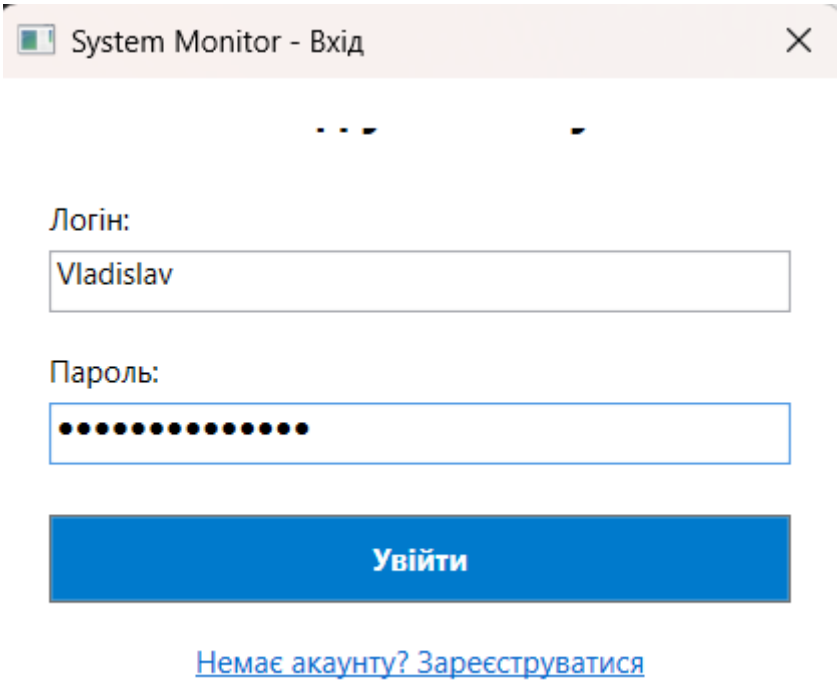

Рис. 5 Форма Авторизації

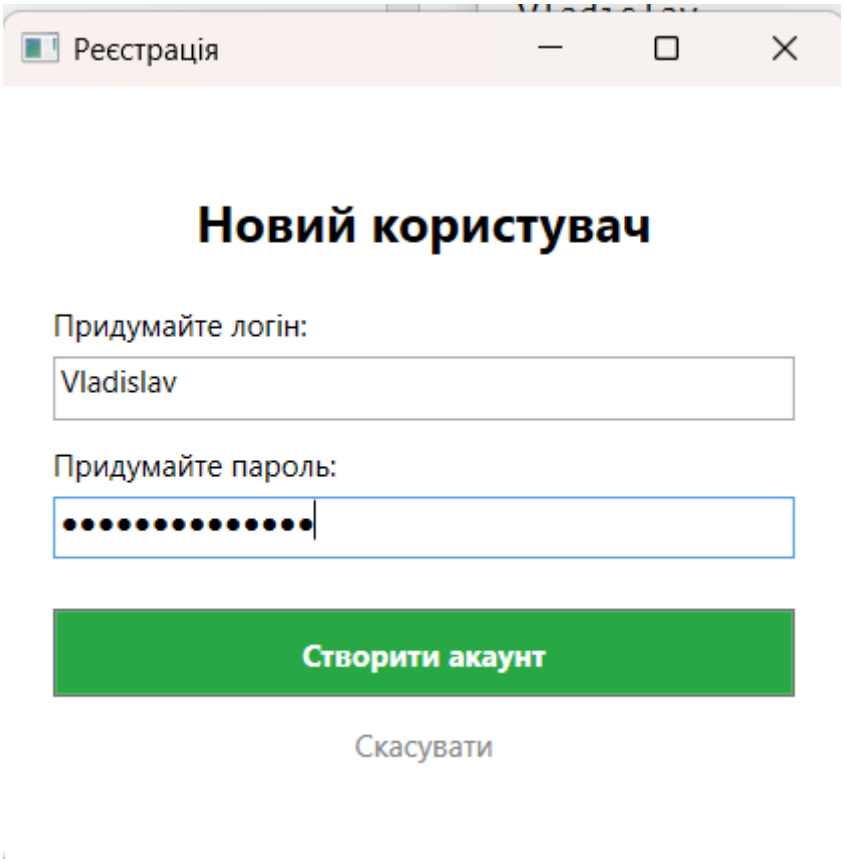

Рис. 6 Форма Реєстрації

Id	Username	PasswordHash	Role	SettingsJson	CreatedAt
Фільтр	Фільтр	Фільтр	Фільтр	Фільтр	Фільтр
11111111-1111-1111-1111-111111111111	admin	240be518fabd2724ddb6f04eeb1da5967448…	Admin	NULL	2025-12-11 19:03:09.415
22222222-2222-2222-2222-222222222222	user	e606e38b0d8c19b24cf0ee3808183162ea7c…	User	NULL	2025-12-11 19:03:09.415
6C709AAE-1894-4CFC-B924-89426C23B836	Vladislav	2052a537688420f2375cc3a708448ac628e5…	User	NULL	2025-12-11 19:19:13.638

Рис. 7 Таблиця SQLite

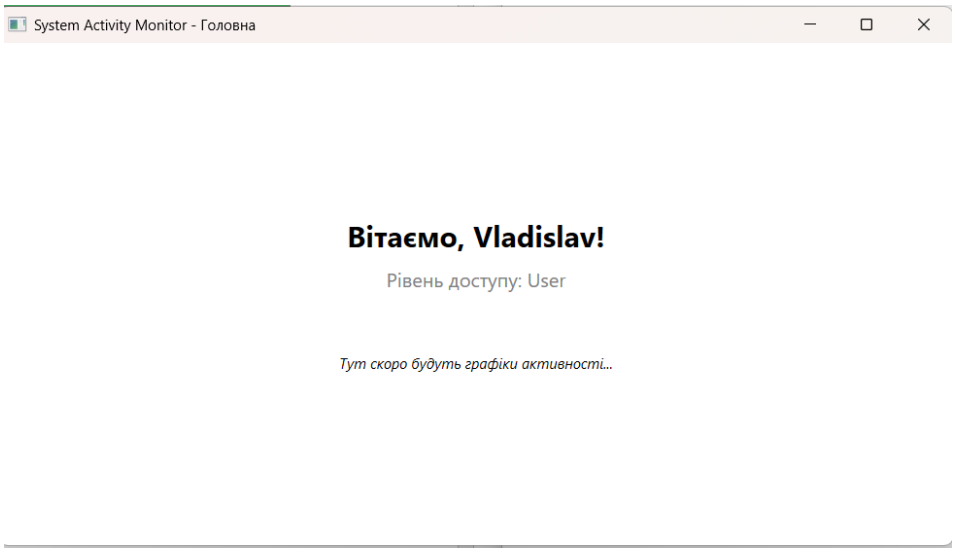

Рис. 8 Успішна авторизація

3. Висновок

У ході виконання лабораторної роботи було успішно досягнуто поставленої мети: опановано створення діаграм розгортання та компонентів для проєктованої системи, що дозволило відобразити її фізичну структуру, розміщення програмних модулів на різних вузлах і характер їхньої взаємодії. Також було засвоєно побудову діаграм взаємодії, зокрема діаграм послідовностей, сформованих на основі сценаріїв із попередньої роботи. Це дало змогу краще зрозуміти логіку обміну повідомленнями між об'єктами та послідовність виконання операцій, що є важливим етапом у моделюванні ПЗ та підготовці до його практичної реалізації.

4. Контрольні питання

1. Що собою становить діаграма розгортання?

Діаграма розгортання — це UML-схема, яка показує фізичну конфігурацію системи: на яких пристроях або середовищах виконання розташовані компоненти програмного забезпечення та як ці елементи взаємодіють у реальних умовах.

2. Які бувають види вузлів на діаграмі розгортання?

На діаграмі розгортання розрізняють два типи вузлів:

- Апаратні вузли — конкретні фізичні пристрої (сервери, ПК, смартфони).
- Програмні вузли — платформи або середовища виконання (ОС, віртуальні машини, контейнери).

3. Які бувають зв'язки на діаграмі розгортання?

Основні зв'язки:

- Асоціація — показує фізичне або логічне з'єднання між вузлами.
- Залежність — відображає, що один вузол чи розміщений на ньому компонент потребує ресурси або функції іншого.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів зазвичай присутні:
компоненти, інтерфейси, порти, залежності між компонентами, пакети та підсистеми.

5. Що становлять собою зв'язки на діаграмі компонентів?

Зв'язки відображають, як один компонент використовує інтерфейси іншого або залежить від нього, тобто показують функціональні та структурні взаємозв'язки між компонентами системи.

6. Які бувають види діаграм взаємодії?

До діаграм взаємодії належать:

- діаграми послідовностей,
- діаграми комунікацій,
- діаграми часу,
- діаграми огляду взаємодії.

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей використовується для відображення порядку передачі повідомлень і викликів між об’єктами під час виконання конкретного сценарію або функціонального процесу.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

На ній можуть бути присутні: актори, об’єкти, лінії життя, повідомлення (синхронні та асинхронні), а також області активації, що показують виконання операцій.

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Вони деталізують кожен варіант використання, показуючи покрокову взаємодію між учасниками сценарію та системою, яка реалізує цей варіант.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Об’єкти, що взаємодіють на діаграмі послідовностей, є конкретними екземплярами класів, а повідомлення між ними відповідають викликам методів, описаних у діаграмі класів.