RECURSOLVE - Temă proiect "Combinatorică"

Nume Elev: Bobea Alin-Gabriel (Denumit în continuare 'elevul')

Profesor Coordonator: Petre Florin

Școala: Colegiul Național "Nicolae Iorga", Vălenii de Munte

SECȚIUNEA 1. Descrierea proiectului

RecurSolve este o aplicație inovatoare și captivantă proiectată pentru a facilita învățarea și explorarea recursivității/backtracking-ului într-un mod interactiv și distractiv. Cu o interfață intuitivă și o gamă diversă de caracteristici, RecurSolve aduce lumea complexă a recursivității la îndemâna utilizatorilor de toate nivelele de experiență în programare și matematică.

SECȚIUNEA 2. Principii și idei folosite la crearea acestui program

Având în vedere regulamentul și cerințele prezentate, elevul a respectat toate principiile enunțate. Acesta afirmă că toate materialele sunt create de el, cu anumite excepții:

-Diverse ecuații matematice
-Anumite poze și alte fișiere de tip "placeholder"
-Diverse idei de programe, design și cromatică
-API-ul folosit pentru asistentul din pagina "acasă" (deținut de OpenAI)

SECȚIUNEA 3. Conținutul programului

SECȚIUNEA 3.1 Pagini principale și meniuri

! Se va prezenta mai întâi codul XAML al paginii, urmat de codul C# atribuit acesteia. !

1. Startup-ul

La deschidere, utilizatorul privește o animație.

```
<Window x:Class="proiectoicd2025.MainWindow"
```

```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        Title="Recursolve" Height="200" Width="520" Loaded="Window_Loaded"
Background="#FF290303"

        WindowStyle="None"

        BorderBrush="Black"

        BorderThickness="1"

        AllowsTransparency="True"

        WindowStartupLocation="CenterScreen">

    <Grid Margin="0,0,-2,0">

        <StackPanel Orientation="Horizontal" HorizontalAlignment="Left"
VerticalAlignment="Top" Name="TitleStackPanel" Margin="23,32,0,0">

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter1" Text="R" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter2" Text="E" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter3" Text="C" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter4" Text="U" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter5" Text="R" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter6" Text="S" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter7" Text="O" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter8" Text="L" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter9" Text="V" Opacity="0"/>

            <TextBlock FontSize="50" FontWeight="Bold" Foreground="White"
x:Name="Letter10" Text="E" Opacity="0"/>

        </StackPanel>
```

```
        <TextBlock FontSize="30" FontStyle="Italic" Foreground="LightGray"
HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,120,0,0"
                   Opacity="0" x:Name="Motto" Text="Solving problems, one step at a
time."/>
    </Grid>
</Window>
--
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Animation;

namespace proiectoicd2025
{
    public partial class MainWindow : Window
    {
        Window login;
        public MainWindow()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            Storyboard titleAnimation = new Storyboard();
            double delay = 0;
```

```
            TextBlock[] letters = { Letter1, Letter2, Letter3, Letter4, Letter5,
Letter6, Letter7, Letter8, Letter9, Letter10 };
            foreach (TextBlock letter in letters)
            {
                AnimateLetter(letter, delay, titleAnimation);
                delay += 100;
            }

            AnimateMotto(Motto, delay + 500, titleAnimation);

            FadeOutEverything(titleAnimation, delay + 2500);

            titleAnimation.Completed += pornesteLogin;

            titleAnimation.Begin();
        }

        private void AnimateLetter(TextBlock letter, double delay, Storyboard
storyboard)
        {
            TranslateTransform moveTransform = new TranslateTransform { X = 500 };
            letter.RenderTransform = moveTransform;

            DoubleAnimation moveAnimation = new DoubleAnimation
            {
                From = 500,
                To = 0,
```

```
                Duration = TimeSpan.FromSeconds(0.8),
                BeginTime = TimeSpan.FromMilliseconds(delay),
                EasingFunction = new QuadraticEase { EasingMode =
EasingMode.EaseOut }
            };

            DoubleAnimation fadeAnimation = new DoubleAnimation
            {
                From = 0,
                To = 1,
                Duration = TimeSpan.FromSeconds(0.5),
                BeginTime = TimeSpan.FromMilliseconds(delay)
            };

            Storyboard.SetTarget(moveAnimation, letter);
            Storyboard.SetTargetProperty(moveAnimation, new
PropertyPath("RenderTransform.X"));

            Storyboard.SetTarget(fadeAnimation, letter);
            Storyboard.SetTargetProperty(fadeAnimation, new
PropertyPath(TextBlock.OpacityProperty));

            storyboard.Children.Add(moveAnimation);
            storyboard.Children.Add(fadeAnimation);
        }

        private void AnimateMotto(TextBlock motto, double delay, Storyboard
storyboard)
        {
```

```
            DoubleAnimation fadeInAnimation = new DoubleAnimation

            {

                From = 0,

                To = 1,

                Duration = TimeSpan.FromSeconds(1),

                BeginTime = TimeSpan.FromMilliseconds(delay),

                EasingFunction = new QuadraticEase { EasingMode =
EasingMode.EaseOut }

            };



            Storyboard.SetTarget(fadeInAnimation, motto);

            Storyboard.SetTargetProperty(fadeInAnimation, new
PropertyPath(TextBlock.OpacityProperty));



            storyboard.Children.Add(fadeInAnimation);

        }



        private void FadeOutEverything(Storyboard storyboard, double delay)

        {

            foreach (var element in new[] { Letter1, Letter2, Letter3, Letter4,
Letter5, Letter6, Letter7, Letter8, Letter9, Letter10, Motto })

            {

                DoubleAnimation fadeOutAnimation = new DoubleAnimation

                {

                    From = 1,

                    To = 0,

                    Duration = TimeSpan.FromSeconds(1.5),

                    BeginTime = TimeSpan.FromMilliseconds(delay),

                    EasingFunction = new QuadraticEase { EasingMode =
```

```
EasingMode.EaseInOut }
                };


                Storyboard.SetTarget(fadeOutAnimation, element);
                Storyboard.SetTargetProperty(fadeOutAnimation, new
PropertyPath(TextBlock.OpacityProperty));
                storyboard.Children.Add(fadeOutAnimation);
            }
        }


        private void pornesteLogin(object sender, EventArgs e)
        {
            this.Hide();
            login = new login();
            login.Show();
        }


    }
}
```

2. Pagina de logare

După ce animație se termină, utilizatorului îi este deschisă pagina de logare.

```
<Window x:Class="proiectoicd2025.login"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Login" Height="600" Width="700"
        MinHeight="600" MinWidth="700"
```

```
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        Background="Transparent"
        ResizeMode="CanResizeWithGrip">

    <Window.Resources>
        <Style x:Key="HighlightButtonStyle" TargetType="Button">
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="5">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
```

```
                    <Setter Property="Foreground" Value="Goldenrod"/>
                </Trigger>
                <Trigger Property="IsPressed" Value="True">
                    <Setter Property="Foreground" Value="Goldenrod"/>
                </Trigger>
            </Style.Triggers>
        </Style>
        <Style x:Key="LoginButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#39b359"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="18"/>
            <Setter Property="BorderBrush" Value="Transparent"/>
            <Setter Property="Padding" Value="5"/>
            <Setter Property="HorizontalContentAlignment" Value="Center"/>
            <Setter Property="VerticalContentAlignment" Value="Center"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="Margin" Value="0,10,0,5"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="5">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
```

```
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="DarkGoldenrod"/>
                </Trigger>
                <Trigger Property="IsPressed" Value="True">
                    <Setter Property="Background" Value="Goldenrod"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Border Background="#FF290303" CornerRadius="8" Padding="0">
        <Grid>
            <Border Height="40" Background="#FF1E1E1E" VerticalAlignment="Top">
                <Grid MouseDown="TitleBar_MouseDown">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition/>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="Auto"/>
                    </Grid.ColumnDefinitions>

                    <TextBlock Text="" Foreground="White" VerticalAlignment="Center"
Margin="10,0,0,0" FontWeight="Bold" Height="40"/>
```

```
                    <Button Grid.Column="1" Content="–" Width="40" Height="30"
Background="Transparent" Foreground="White"
                    BorderThickness="0" Click="Minimize_Click" Cursor="Hand"/>
                    <Button Grid.Column="2" Content="❐" Width="40" Height="30"
Background="Transparent" Foreground="White"
                    BorderThickness="0" Click="MaximizeRestore_Click"
Cursor="Hand"/>
                    <Button Grid.Column="3" Content="X" Width="40" Height="30"
Background="Transparent" Foreground="White"
                    BorderThickness="0" Click="Close_Click" Cursor="Hand"/>
                </Grid>
            </Border>

            <Viewbox Stretch="Uniform" Margin="0,40,0,0">
                <Grid Width="650" Height="560">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="0.5*"/>
                        <RowDefinition Height="1.5*"/>
                        <RowDefinition Height="0.5*"/>
                    </Grid.RowDefinitions>
                    <Button Content="ENG/ROM"
                        FontSize="20"
                        Foreground="White"
                        Background="Transparent"
                        BorderThickness="0"
                        Cursor="Hand"
                        HorizontalAlignment="Left"
                        VerticalAlignment="Top"
                        Margin="-38,-3,0,0"
```

```
                    Click="ChangeLanguage_Click"
                    Height="42" Width="130"
                    Style="{StaticResource HighlightButtonStyle}"/>
                <TextBlock Text="RecurSolve"
                   Grid.Row="0"
                   FontSize="50" FontWeight="Bold"
                   Foreground="White"
                   HorizontalAlignment="Center" VerticalAlignment="Center"/>

                <Border Grid.Row="1" Background="#FF4A0A0A" CornerRadius="10"
Padding="30"
                MinWidth="300" MinHeight="200"
                HorizontalAlignment="Stretch" VerticalAlignment="Center">
                    <StackPanel>
                        <TextBlock Text="Email" Foreground="White"
FontSize="18"/>
                        <TextBox x:Name="UsernameBox" Height="30" FontSize="16"
                         Background="#FF383838" Foreground="White"
                         BorderThickness="0" Margin="0,5,0,15"
                         Padding="5,5,0,0">
                            <TextBox.Template>
                                <ControlTemplate TargetType="TextBox">
                                    <Border Background="{TemplateBinding
Background}"
                                    BorderBrush="{TemplateBinding BorderBrush}"
                                    BorderThickness="{TemplateBinding
BorderThickness}"
                                    CornerRadius="8">
                                        <ScrollViewer x:Name="PART_ContentHost"
```

```
/>
                                    </Border>
                                </ControlTemplate>
                            </TextBox.Template>
                        </TextBox>

                        <TextBlock Text="{DynamicResource parolaRegister}"
Foreground="White" FontSize="18"/>
                        <PasswordBox x:Name="PasswordBox" Height="30"
FontSize="16"
                             Background="#FF383838" Foreground="White"
                             BorderThickness="0" Margin="0,5,0,15"
                             Padding="5,4,0,0"
                             >
                            <PasswordBox.Template>
                                <ControlTemplate TargetType="PasswordBox">
                                    <Border Background="{TemplateBinding
Background}"
                                        BorderBrush="{TemplateBinding BorderBrush}"
                                        BorderThickness="{TemplateBinding
BorderThickness}"
                                        CornerRadius="8">
                                        <ScrollViewer x:Name="PART_ContentHost"
/>
                                    </Border>
                                </ControlTemplate>
                            </PasswordBox.Template>
                        </PasswordBox>
```

```
                        <Button Content="Login"
                            HorizontalAlignment="Stretch"
                            Height="40"
                            Click="pornesteAcasa"
                            Style="{StaticResource LoginButtonStyle}"/>

                        <Button Content="{DynamicResource nuAiCont}"
                            FontSize="16"
                            Foreground="White"
                            Background="Transparent"
                            BorderThickness="0"
                            Cursor="Hand"
                            HorizontalAlignment="Center"
                            Click="RegisterLink_Click"
                            Style="{StaticResource HighlightButtonStyle}"/>
                    </StackPanel>
                </Border>

                <TextBlock Grid.Row="2"
                   Text="© 2025 RecurSolve by Bobea Alin"
                   Foreground="Gray" FontSize="14"
                   HorizontalAlignment="Center" VerticalAlignment="Bottom"
                   Padding="0,15"/>
            </Grid>
        </Viewbox>
    </Grid>
</Border>
```

```
</Window>

--
using proiectoicd2025.main.misc;
using System;
using System.Data.SqlClient;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using Path = System.IO.Path;

namespace proiectoicd2025
{
    public partial class login : Window
    {
        string conString;
        SqlConnection con;
        SqlCommand com;
        SqlDataReader dr;
        Window acasa;
        Window register;
        bool register_deschis = false;
        public static string utilizator_logat;

        public login()
        {
            InitializeComponent();
            string dbPath =
```

```
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Database",
"bazadate.mdf");

            string conString = $@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={dbPath};Integrated Security=True;";

            con = new SqlConnection(conString);




        }


        private void CheckBox_Checked(object sender, RoutedEventArgs e)

        {


        }


        private void pornesteAcasa(object sender, RoutedEventArgs e)

        {

            try

            {

                string encryptedEmail = Criptografie.Cripteaza(UsernameBox.Text);


                con.Open();

                com = new SqlCommand("SELECT parola, [user] FROM utilizatori WHERE
email = @e", con);

                com.Parameters.AddWithValue("@e", encryptedEmail);

                dr = com.ExecuteReader();


                if (dr.HasRows)

                {

                    while (dr.Read())
```

```
                    {
                        string encryptedPassword = dr["parola"].ToString();
                        string decryptedPassword =
Criptografie.Decripteaza(encryptedPassword);

                        if (decryptedPassword == PasswordBox.Password)
                        {
                            utilizator_logat =
Criptografie.Decripteaza(dr["user"].ToString());
                            manage.utilizatorCurent = utilizator_logat;
                            MessageBox.Show("Bine ai venit, " + utilizator_logat);

                            this.Hide();
                            acasa = new acasa();
                            acasa.Show();
                            con.Close();
                            return;
                        }
                        else
                        {
                            MessageBox.Show("Parola incorectă.", "Login Failed",
MessageBoxButton.OK, MessageBoxImage.Error);
                        }
                    }
                }
                else
                {
                    MessageBox.Show("Email-ul nu există.", "Login Failed",
MessageBoxButton.OK, MessageBoxImage.Error);
```

```
            }

            UsernameBox.Clear();
            PasswordBox.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Eroare: " + ex.Message);
        }
        finally
        {
            con.Close();
        }
    }



    private void RegisterLink_Click(object sender, RoutedEventArgs e)
    {
        if (!register_deschis)
        {
            register register = new register();
            register.Show();
            this.Close();
            //register_deschis = true;
        }
        else
```

```
    {
        //register_deschis = false;
        register.Close();
    }
}
private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        this.DragMove();
}

private void Minimize_Click(object sender, RoutedEventArgs e)
{
    WindowState = WindowState.Minimized;
}

private void MaximizeRestore_Click(object sender, RoutedEventArgs e)
{
    if (WindowState == WindowState.Maximized)
        WindowState = WindowState.Normal;
    else
        WindowState = WindowState.Maximized;
}

private void Close_Click(object sender, RoutedEventArgs e)
{
    Close();
```

```
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
```

```
            }

            Application.Current.Resources.MergedDictionaries.Clear();
Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

3. Pagina de register

Utilizatorul poate alege să-și creeze un cont din pagina de login dacă dorește. Acesta va trebui să introducă date corecte (email valid, parolă puternică) și să își aleagă o poză de profil.

```
<Window x:Class="proiectoicd2025.register"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Register" Height="700" Width="1000"
        MinHeight="700" MinWidth="900"
        ResizeMode="CanResize"
        WindowStartupLocation="CenterScreen"
        Background="Transparent"
        WindowStyle="None"
        AllowsTransparency="True">

    <Window.Resources>
        <Style TargetType="Button" x:Key="mouseOverStyle">
```

```
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Foreground" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Border Height="40" Background="#FF1E1E1E" VerticalAlignment="Top">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="" Foreground="White" VerticalAlignment="Center"
Margin="10,0,0,0" FontWeight="Bold"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Minimize_Click"/>
                <Button Grid.Column="3" Content="X" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Close_Click"/>
            </Grid>
        </Border>
```

```
        <Rectangle Fill="#FF290303" Width="{Binding ActualWidth,
RelativeSource={RelativeSource AncestorType=Window}}"

                   Height="{Binding ActualHeight, RelativeSource={RelativeSource
AncestorType=Window}}" Margin="0,40,0,0"/>


        <ScrollViewer Margin="0,40,0,0" VerticalScrollBarVisibility="Auto">

            <Grid Width="900" Height="630" HorizontalAlignment="Center"
VerticalAlignment="Center">

                <Grid.RowDefinitions>

                    <RowDefinition Height="0.2*"/>

                    <RowDefinition Height="2*"/>

                    <RowDefinition Height="0.2*"/>

                </Grid.RowDefinitions>


                <Grid.ColumnDefinitions>

                    <ColumnDefinition Width="3*"/>

                    <ColumnDefinition Width="2*"/>

                </Grid.ColumnDefinitions>


                <Button Content="ENG/ROM"

                        FontSize="20"

                        Foreground="White"

                        Background="Transparent"

                        BorderThickness="0"

                        Cursor="Hand"

                        Style="{StaticResource mouseOverStyle}"

                        HorizontalAlignment="Left"

                        VerticalAlignment="Top"
```

```
                    Margin="-38,-3,0,0"
                    Click="ChangeLanguage_Click"
                    Height="42" Width="130"/>

                <TextBlock Text="RecurSolve"
                           FontSize="50" FontWeight="Bold"
                           Foreground="White"
                           HorizontalAlignment="Left" VerticalAlignment="Top"
                           Height="69" Margin="320,10,0,0"
                           Grid.RowSpan="2" Grid.ColumnSpan="2"/>

                <Border Grid.Row="1" Grid.Column="0" Margin="-15,0,0,0"
                        Background="#FF4A0A0A" CornerRadius="10"
                        Padding="30"
                        MinWidth="300" MinHeight="350" VerticalAlignment="Center"
Height="452">
                    <StackPanel>
                        <TextBlock Text="{DynamicResource userRegister}"
Foreground="White" FontSize="18"/>
                        <TextBox x:Name="UsernameBox" Height="30" FontSize="16"
                                 Background="#FF383838" Foreground="White"
                                 BorderThickness="0" Margin="0,5,0,15"
                                 Padding="5,5,0,0"/>

                        <TextBlock Text="{DynamicResource emailRegister}"
Foreground="White" FontSize="18"/>
                        <TextBox x:Name="EmailBox" Height="30" FontSize="16"
                                 Background="#FF383838" Foreground="White"
                                 BorderThickness="0" Margin="0,5,0,15"
```

```
                                Padding="5,5,0,0"/>


                        <TextBlock Text="{DynamicResource parolaRegister}"
Foreground="White" FontSize="18"/>
                        <PasswordBox x:Name="PasswordBox" Height="30" FontSize="16"
                                     Background="#FF383838" Foreground="White"
                                     BorderThickness="0" Margin="0,5,0,15"
                                     Padding="5,4,0,0"/>


                        <TextBlock Text="{DynamicResource confirmaRegister}"
Foreground="White" FontSize="18"/>
                        <PasswordBox x:Name="ConfirmPasswordBox" Height="30"
FontSize="16"
                                     Background="#FF383838" Foreground="White"
                                     BorderThickness="0" Margin="0,5,0,15"
                                     Padding="5,4,0,0"/>


                        <Button Content="{DynamicResource inregistreazaRegister}"
                                HorizontalAlignment="Stretch"
                                Height="40"
                                Margin="0,10,0,5"
                                Click="Register_Click">
                            <Button.Style>
                                <Style TargetType="Button">
                                    <Setter Property="Background" Value="#39b359"/>
                                    <Setter Property="Foreground" Value="White"/>
                                    <Setter Property="FontSize" Value="18"/>
                                    <Setter Property="BorderBrush"
Value="Transparent"/>
```

```
                                <Setter Property="Padding" Value="5"/>
                                <Setter Property="HorizontalContentAlignment"
Value="Center"/>
                                <Setter Property="VerticalContentAlignment"
Value="Center"/>
                                <Setter Property="BorderThickness" Value="0"/>
                                <Setter Property="Cursor" Value="Hand"/>
                                <Setter Property="Margin" Value="0,10,0,5"/>
                            </Style>
                        </Button.Style>
                    </Button>

                    <Button Content="{DynamicResource existRegister}"
                            FontSize="16"
                            Foreground="White"
                            Background="Transparent"
                            BorderThickness="0"
                            Cursor="Hand"
                            Style="{StaticResource mouseOverStyle}"
                            Click="GoToLoginPage_Click"/>
                </StackPanel>
            </Border>

            <Border Grid.Row="1" Grid.Column="1" Margin="3,0,0,0"
                    Background="#FF4A0A0A" CornerRadius="10"
                    Padding="30" MinWidth="300" VerticalAlignment="Center"
Height="452">
                <StackPanel HorizontalAlignment="Center"
VerticalAlignment="Center" Width="340" Height="432">
```

```
                        <TextBlock Text="{DynamicResource profilepicRegister}"
Foreground="White" FontSize="18" Margin="20"/>
                        <Image x:Name="ProfileImage" Width="200" Height="200"
Margin="0,45"/>
                        <Button Content="{DynamicResource uploadRegister}"
                                Background="#39b359" Foreground="White"
FontSize="16" Height="40"
                                Cursor="Hand" Click="UploadProfilePicture_Click"/>
                    </StackPanel>
                </Border>

                <TextBlock Text="© 2025 RecurSolve by Bobea Alin. All rights
reserved."
                           Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2"
                           Foreground="Gray" FontSize="14"
                           HorizontalAlignment="Center" VerticalAlignment="Bottom"
                           Padding="0,15"/>
            </Grid>
        </ScrollViewer>
    </Grid>
</Window>


--
using Microsoft.Win32;
using proiectoicd2025.main.misc;
using System;
using System.Data.SqlClient;
using System.IO;
using System.Text.RegularExpressions;
```

```
using System.Windows;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using Path = System.IO.Path;

namespace proiectoicd2025
{
    public partial class register : Window
    {
        SqlConnection con;
        SqlCommand com;
        SqlDataReader dr;
        string selectedImagePath = null;

        public register()
        {
            InitializeComponent();
            string dbPath =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Database",
"bazadate.mdf");
            string conString = $@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={dbPath};Integrated Security=True;";
            con = new SqlConnection(conString);
        }

        private void UploadProfilePicture_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp;*.gif";
```

```
            if (openFileDialog.ShowDialog() == true)
            {
                selectedImagePath = openFileDialog.FileName;
                ProfileImage.Source = new BitmapImage(new Uri(selectedImagePath));
            }
        }

        private void Register_Click(object sender, RoutedEventArgs e)
        {
            string username = UsernameBox.Text;
            string email = EmailBox.Text;
            string password = PasswordBox.Password;
            string confirmPassword = ConfirmPasswordBox.Password;

            if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(email) ||
string.IsNullOrEmpty(password) || string.IsNullOrEmpty(confirmPassword))
            {
                MessageBox.Show("Please fill in all fields.", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            if (!IsValidEmail(email))
            {
                citafis("EmailError");
                return;
            }
```

```
            if (!IsStrongPassword(password))
            {
                MessageBox.Show("Password must be at least 8 characters long, " +
                                "contain at least one uppercase letter, one
lowercase letter, " +
                                "one number, and one special character.",
                                "Error", MessageBoxButton.OK,
MessageBoxImage.Error);
                return;
            }

            if (password != confirmPassword)
            {
                citafis("notmatch");
                return;
            }

            if (!string.IsNullOrEmpty(selectedImagePath))
            {
                string targetDir =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource",
"profilepics");
                Directory.CreateDirectory(targetDir);

                string destPath = System.IO.Path.Combine(targetDir, username +
".png");

                try
```

```
            {
                File.Copy(selectedImagePath, destPath, overwrite: true);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error " + ex.Message);
                return;
            }
        }

        try
        {
            con.Open();

            string encryptedEmail = Criptografie.Cripteaza(email);

            com = new SqlCommand("select * from utilizatori where email = @e",
con);
            com.Parameters.AddWithValue("@e", encryptedEmail);
            dr = com.ExecuteReader();

            if (dr.HasRows)
            {
                citafis("EmailTakenError");
                con.Close();
                return;
            }
            con.Close();
```

```
                string encryptedUsername = Criptografie.Cripteaza(username);
                string encryptedPassword = Criptografie.Cripteaza(password);

                con.Open();
                com = new SqlCommand("insert into utilizatori (email, parola,
[user]) values (@e, @p, @u)", con);
                com.Parameters.AddWithValue("@u", encryptedUsername);
                com.Parameters.AddWithValue("@e", encryptedEmail);
                com.Parameters.AddWithValue("@p", encryptedPassword);
                com.ExecuteNonQuery();
                con.Close();

                citafis("register_success");
                this.Hide();
                login loginWindow = new login();
                loginWindow.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error {ex.Message}", "Error", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
        }

        private void GoToLoginPage_Click(object sender, RoutedEventArgs e)
        {
            this.Hide();
```

```
            login loginWindow = new login();
            loginWindow.Show();
        }

        private bool IsValidEmail(string email)
        {
            var emailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
            return Regex.IsMatch(email, emailPattern);
        }

        private bool IsStrongPassword(string password)
        {
            if (password.Length < 8)
                return false;

            bool hasUpper = false, hasLower = false, hasDigit = false,
hasSpecialChar = false;

            foreach (var c in password)
            {
                if (char.IsUpper(c)) hasUpper = true;
                if (char.IsLower(c)) hasLower = true;
                if (char.IsDigit(c)) hasDigit = true;
                if (char.IsPunctuation(c) || char.IsSymbol(c)) hasSpecialChar =
true;
            }

            return hasUpper && hasLower && hasDigit && hasSpecialChar;
```

```
}

public string language = "English";
public bool infoOK = false;
main.misc.Info info;
string filePath;

public int curlang = 0;
private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
{
    if (curlang == 0)
    {
        ChangeLanguage("Romanian");
        language = "Romanian";
        curlang = 1;
    }
    else
    {
        ChangeLanguage("English");
        language = "English";
        curlang = 0;
    }
}

private void ChangeLanguage(string language)
{
    var resourceDictionary = new ResourceDictionary();
```

```
            if (language == "English")
            {
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
            {
                this.DragMove();
            }
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }
```

```
        private void MaximizeRestore_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = (this.WindowState == WindowState.Maximized)
                ? WindowState.Normal
                : WindowState.Maximized;
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
        private void citafis(string key)
        {
            string message = Application.Current.Resources[key] as string;
            MessageBox.Show(message);
        }
    }
}
```

4. Pagina de acasă

Locul unde utilizatorul este redresat după logare. De aici poate interacționa cu asistenul AI, cu diversele meniuri, etc.

```
<Window x:Class="proiectoicd2025.acasa"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:shell="clr-namespace:System.Windows.Shell;assembly=PresentationFramework"
    Title="RecurSolve"
```

```
    Height="700"
    Width="1400"
    MinHeight="895"
    MinWidth="1400"
    WindowStyle="None"
    Background="Transparent"
    AllowsTransparency="True"
    WindowStartupLocation="CenterScreen"
    Loaded="Window_Loaded">

<Window.Resources>
    <Style TargetType="Window">
        <Setter Property="shell:WindowChrome.WindowChrome">
            <Setter.Value>
                <shell:WindowChrome
                    CaptionHeight="0"
                    ResizeBorderThickness="6"
                    GlassFrameThickness="0"
                    UseAeroCaptionButtons="False"/>
            </Setter.Value>
        </Setter>
    </Style>

    <Style x:Key="SidebarButtonStyle" TargetType="Button">
        <Setter Property="Foreground" Value="White"/>
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="FontSize" Value="24"/>
```

```
    <Setter Property="Height" Value="70"/>
    <Setter Property="Margin" Value="10,10,10,0"/>
    <Setter Property="HorizontalContentAlignment" Value="Left"/>
    <Setter Property="BorderBrush" Value="White"/>
    <Setter Property="BorderThickness" Value="0,0,0,2"/>
    <Setter Property="Cursor" Value="Hand"/>
</Style>

<Style x:Key="SidebarButtonStyle2" TargetType="Button">
    <Setter Property="Foreground" Value="White"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="FontSize" Value="24"/>
    <Setter Property="Height" Value="70"/>
    <Setter Property="Margin" Value="10,10,10,0"/>
    <Setter Property="HorizontalContentAlignment" Value="Left"/>
    <Setter Property="BorderBrush" Value="White"/>
    <Setter Property="BorderThickness" Value="0,0,0,0"/>
    <Setter Property="Cursor" Value="Hand"/>
</Style>

<Style x:Key="SidebarExpanderStyle" TargetType="Expander">
    <Setter Property="Foreground" Value="White"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="FontSize" Value="26"/>
    <Setter Property="HeaderTemplate">
        <Setter.Value>
            <DataTemplate>
```

```
                <TextBlock Text="{Binding}" FontWeight="Bold" Padding="5"/>
            </DataTemplate>
          </Setter.Value>
        </Setter>
      </Style>
    </Window.Resources>

    <Border CornerRadius="20" Background="#333333" SnapsToDevicePixels="True" Margin="10">
      <Border.Effect>
        <DropShadowEffect BlurRadius="15" ShadowDepth="0" Color="Black" Opacity="0.5"/>
      </Border.Effect>

      <Grid>
        <Grid.RowDefinitions>
          <RowDefinition Height="40"/>
          <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C">
          <Grid MouseDown="TitleBar_MouseDown">
            <Grid.ColumnDefinitions>
              <ColumnDefinition/>
              <ColumnDefinition Width="Auto"/>
              <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <TextBlock Text="" Foreground="White" FontWeight="Bold" FontSize="16" VerticalAlignment="Center"
Margin="10,0,10,0" Height="40"/>
            <StackPanel Orientation="Horizontal" Grid.Column="2" VerticalAlignment="Center">
```

```
                <Button Content="—" Width="40" Height="40" Click="Minimize_Click" Background="Transparent"
Foreground="White"/>

                <Button Content="X" Width="40" Height="40" Click="Close_Click" Background="Transparent"
Foreground="White"/>

            </StackPanel>

            <Button Content="ENG/ROM"

    FontSize="20"

    Foreground="White"

    Background="Transparent"

    BorderThickness="0"

    Cursor="Hand"

    HorizontalAlignment="Left"

    VerticalAlignment="Center"

    Margin="14,0,0,0"

    Click="ChangeLanguage_Click"

    Height="42" Width="130"/>

        </Grid>

      </Border>


      <Grid Grid.Row="1">

        <Grid.ColumnDefinitions>

          <ColumnDefinition Width="400"/>

          <ColumnDefinition Width="5"/>

          <ColumnDefinition Width="*"/>

        </Grid.ColumnDefinitions>


        <Grid Grid.Column="0" Background="#FF4A0A0A">

          <Grid.RowDefinitions>
```

```
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="404.324"/>
            <RowDefinition/>
            <RowDefinition Height="Auto" MinHeight="98.369"/>
        </Grid.RowDefinitions>

        <StackPanel Margin="20,20,20,20">
            <StackPanel Orientation="Horizontal" VerticalAlignment="Center">
                <TextBlock Text="RecurSolve" FontSize="34" FontWeight="Bold" Foreground="White"/>
                <Button x:Name="accountPic" Click="open_account" Width="80" Height="80" Background="Transparent"
BorderThickness="0" Margin="100,0,0,0">
                    <Ellipse Width="70" Height="70" Stroke="White" StrokeThickness="1">
                        <Ellipse.Fill>
                            <ImageBrush x:Name="picSource" ImageSource="resource/profilepics/user_icon.png"/>
                        </Ellipse.Fill>
                    </Ellipse>
                </Button>
            </StackPanel>
            <TextBlock Text="{DynamicResource menu}" FontSize="24" Foreground="#DDDDDD" Margin="0,10,0,5"
Width="359" Height="31"/>
        </StackPanel>

        <StackPanel Grid.Row="1" Margin="10,0,10,80" Grid.RowSpan="3">
            <Expander Header="{DynamicResource drop1}" Style="{StaticResource SidebarExpanderStyle}">
                <StackPanel>
                    <Button Content="{DynamicResource math}" Click="showmateMenu" Style="{StaticResource
SidebarButtonStyle}"/>
                    <Button Content="{DynamicResource algo}" Click="showalgoMenu" Style="{StaticResource
SidebarButtonStyle}"/>
```

```
            </StackPanel>

          </Expander>


          <Expander Header="{DynamicResource drop2}" Style="{StaticResource SidebarExpanderStyle}"
Margin="0,20,0,0">

            <StackPanel>

              <Button Content="{DynamicResource gameMenu}" Style="{StaticResource SidebarButtonStyle}"
Click="open_games"/>

            </StackPanel>

          </Expander>


          <Expander Header="{DynamicResource drop3}" Style="{StaticResource SidebarExpanderStyle}"
Margin="0,20,0,0">

            <StackPanel>

              <Button Content="{DynamicResource quizMenu}" Click="open_quiz" Style="{StaticResource
SidebarButtonStyle}"/>

            </StackPanel>

          </Expander>

        </StackPanel>


        <StackPanel Grid.Row="3" Margin="0,0,0,0" VerticalAlignment="Center" HorizontalAlignment="Left" Height="80">

          <Button Content="{DynamicResource logout}" Click="log_Out" Style="{StaticResource SidebarButtonStyle}"
Width="149" />

        </StackPanel>

        <Button Content="{DynamicResource feedback}" Click="sendFeedback"  Style="{StaticResource
SidebarButtonStyle2}" Height="NaN" Margin="200,23,10,9" Grid.Row="3"/>

      </Grid>


      <GridSplitter Grid.Column="1" Width="5" Background="#555" HorizontalAlignment="Stretch"/>
```

```
        <Grid Grid.Column="2" Background="#333333">
            <StackPanel Margin="30" VerticalAlignment="Top">
                <TextBlock Text="Welcome Back, Username!" x:Name="welcome" Foreground="White" FontSize="37" FontWeight="Bold"/>
                <TextBlock Text="{DynamicResource itsgood}" Foreground="#CCCCCC" FontSize="35" Width="417" Margin="350,0,0,0"/>

                <Grid x:Name="FunFactsCarousel" HorizontalAlignment="Center" VerticalAlignment="Top" Height="250" Width="900" Margin="0,20,0,0">
                    <Grid.Resources>
                        <Style TargetType="Border" x:Key="FactStyle">
                            <Setter Property="CornerRadius" Value="15"/>
                            <Setter Property="Background" Value="#555"/>
                            <Setter Property="Padding" Value="10"/>
                            <Setter Property="Margin" Value="10"/>
                            <Setter Property="HorizontalAlignment" Value="Center"/>
                            <Setter Property="VerticalAlignment" Value="Center"/>
                            <Setter Property="RenderTransformOrigin" Value="0.5,0.5"/>
                        </Style>
                        <Style x:Key="FactTextStyle" TargetType="TextBlock">
                            <Setter Property="FontFamily" Value="Consolas"/>
                            <Setter Property="TextAlignment" Value="Center"/>
                            <Setter Property="VerticalAlignment" Value="Center"/>
                            <Setter Property="TextWrapping" Value="Wrap"/>
                            <Setter Property="Foreground" Value="White"/>
                        </Style>
                    </Grid.Resources>

                    <Grid HorizontalAlignment="Center">
```

```
                <Border x:Name="Fact1" Style="{StaticResource FactStyle}" Width="200" Height="130" Opacity="0.6">
                  <Border.RenderTransform>
                    <TranslateTransform X="-250" />
                  </Border.RenderTransform>
                  <TextBlock x:Name="Fact1Text" Text="Fact 1" FontSize="14" Style="{StaticResource FactTextStyle}"/>
                </Border>

                <Border x:Name="Fact2" Style="{StaticResource FactStyle}" Width="300" Height="200" Opacity="1">
                  <Border.RenderTransform>
                    <TranslateTransform X="0" />
                  </Border.RenderTransform>
                  <Viewbox Stretch="Uniform">
                    <TextBlock x:Name="Fact2Text" Text="Fact 2" Style="{StaticResource FactTextStyle}"
TextWrapping="Wrap" TextAlignment="Center" VerticalAlignment="Center"/>
                  </Viewbox>
                </Border>

                <Border x:Name="Fact3" Style="{StaticResource FactStyle}" Width="200" Height="130" Opacity="0.6">
                  <Border.RenderTransform>
                    <TranslateTransform X="250" />
                  </Border.RenderTransform>
                  <TextBlock x:Name="Fact3Text" Text="Fact 3" FontSize="14" Style="{StaticResource FactTextStyle}"/>
                </Border>

                <Border x:Name="Fact4" Style="{StaticResource FactStyle}" Width="200" Height="130" Opacity="0.6">
                  <Border.RenderTransform>
                    <TranslateTransform X="250" />
                  </Border.RenderTransform>
```

```
                    <TextBlock x:Name="Fact4Text" Text="Fact 4" FontSize="14" Style="{StaticResource FactTextStyle}"/>
                </Border>
            </Grid>
        </Grid>

        <Border Margin="0,30,0,30" Background="#444" CornerRadius="15" Padding="10">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>

                <ScrollViewer Grid.Row="0" VerticalScrollBarVisibility="Auto" Height="300">
                    <StackPanel x:Name="ChatMessagesPanel" Margin="5">
                        <Border Background="#555" CornerRadius="10" Padding="10" Margin="5" HorizontalAlignment="Left"
MaxWidth="500">
                            <TextBlock Text="Hello! How can I assist you in backtracking today? Cum te pot ajuta?" FontSize="18"
Foreground="White" TextWrapping="Wrap"/>
                        </Border>
                    </StackPanel>
                </ScrollViewer>

                <DockPanel Grid.Row="1" Margin="0,10,0,0">
                    <TextBox x:Name="ChatInputBox" Width="797" FontSize="16" Padding="0,8,0,0"
                             Height="40" Background="#666" Foreground="White" BorderBrush="#888" BorderThickness="1"
                             Margin="0,0,10,0" DockPanel.Dock="Left" TextAlignment="Left"  Text=""/>
                    <Button Content="{DynamicResource send}" Width="80" Height="40" FontWeight="Bold"
Background="#FF4A0A0A" Foreground="White"
                            BorderThickness="0" Cursor="Hand" Click="Send_Click"/>
```

```
                    </DockPanel>
                </Grid>
            </Border>

        </StackPanel>
    </Grid>
  </Grid>
 </Grid>
</Border>
</Window>
```

--

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media.Animation;
using System.Windows.Media;
using System.Windows.Threading;
using System.Windows.Input;
using Newtonsoft.Json;
using System.Net.Http;
using System.Text;
using System.IO;
using System.Windows.Media.Imaging;
using proiectoicd2025.main.misc;
using System.Runtime.InteropServices;
```

```
namespace proiectoicd2025
{
    public class Message
    {
        public string role { get; set; }
        public string content { get; set; }
    }


    public partial class acasa : Window
    {
        private List<Message> _conversationHistory = new List<Message>();

        private DispatcherTimer _carouselTimer;
        private Dictionary<string, List<string>> _funFacts;
        private Random _random;

        private Border Fact1Border, Fact2Border, Fact3Border, Fact4Border;
        private TextBlock Fact1TextBlock, Fact2TextBlock, Fact3TextBlock, Fact4TextBlock;

        string username;
        public acasa()
        {
            InitializeComponent();
            _conversationHistory.Add(new Message
            {
```

```
            role = "system",

            content = "You are an AI assistant embedded in an educational WPF app. Your role is to help students understand the concept of backtracking in computer science. Answer questions clearly, use examples where helpful, stay concise and friendly, and guide them to learn through exploration and explanation."

        });



        _funFacts = new Dictionary<string, List<string>>()

  {

    { "English", new List<string>

      {

        "Backtracking is used to solve puzzles like Sudoku!",

        "The N-Queens problem is a classic backtracking example.",

        "Backtracking explores all possibilities systematically.",

        "Crossword puzzle generation uses backtracking!",

        "It's a brute-force technique made smarter with pruning.",

        "Backtracking can undo choices step-by-step.",

        "Used in AI for constraint satisfaction problems.",

        "Mazes are solved with backtracking algorithms!",

        "Backtracking is like trying all paths until you win!",

        "It's simple but powerful for exponential problems."

      }

    },

    { "Romanian", new List<string>

      {

        "Backtracking-ul este folosit pentru a rezolva puzzle-uri precum Sudoku!",

        "Problema celor N regine este un exemplu clasic de backtracking.",

        "Backtracking explorează toate posibilitățile sistematic.",

        "Generarea puzzle-urilor de cuvinte încrucișate folosește backtracking!",
```

```
            "Este o tehnică de forță brută făcută mai inteligentă cu pruning.",
            "Backtracking poate anula alegerile pas cu pas.",
            "Este folosit în AI pentru probleme de satisfacție a constrângerilor.",
            "Labirinturile sunt rezolvate cu algoritmi de backtracking!",
            "Backtracking este ca și cum ai încerca toate drumurile până câștigi!",
            "Este simplu, dar puternic pentru probleme exponențiale."
        }
    }
};
        _random = new Random();

        SetupCarouselElements();
        SetupInitialFacts();
        StartCarousel();
        username = manage.utilizatorCurent;
        LoadProfileImage(false);
        DisplayWelcomeMessage();
    }

    private void DisplayWelcomeMessage()
    {
        string welcomeMessage = string.Empty;

        if (language == "English")
        {
            welcomeMessage = $"Welcome back, {username}!";
        }
```

```
        else if (language == "Romanian")
        {
            welcomeMessage = $"Bine ai venit, {username}!";
        }

        welcome.Text = welcomeMessage;
    }

    public void LoadProfileImage(bool call)
    {
        string tempImagePath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource", "profilepics",
"temp_profile_image.png");
        string imagePath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource", "profilepics",
$"{username}.png");

        string finalImagePath;

        if (File.Exists(tempImagePath) && call)
            finalImagePath = tempImagePath;
        else if (File.Exists(imagePath))
            finalImagePath = imagePath;
        else
        {
            picSource.ImageSource = new BitmapImage(new Uri("resource/profilepics/user_icon.png", UriKind.Relative));
            return;
        }

        try
```

```
    {
        using (FileStream fs = new FileStream(finalImagePath, FileMode.Open, FileAccess.Read, FileShare.ReadWrite))
        {
            BitmapImage bmp = new BitmapImage();
            bmp.BeginInit();
            bmp.CacheOption = BitmapCacheOption.OnLoad;
            bmp.StreamSource = fs;
            bmp.EndInit();
            bmp.Freeze();

            picSource.ImageSource = null;
            picSource.ImageSource = bmp;
        }
    }
    catch (Exception ex)
    {
        picSource.ImageSource = new BitmapImage(new Uri("resource/profilepics/user_icon.png", UriKind.Relative));
        MessageBox.Show($"Failed to load profile image: {ex.Message}");
    }
}

public void sendFeedback(object sender, RoutedEventArgs e)
{
    contact contcustom = new contact();
    contcustom.Show();
}
```

```
    public void log_Out(object sender, RoutedEventArgs e)
    {
        MessageBoxResult result = MessageBox.Show("Are you sure you want to log out?", "Log Out",
MessageBoxButton.YesNo, MessageBoxImage.Warning);
        if (result == MessageBoxResult.Yes)
        {
            manage.utilizatorCurent = "";
            login Login = new login();
            this.Close();
            Login.Show();
        }
    }

    public void open_account(object sender, RoutedEventArgs e)
    {
        account contcustom = new account(this);
        contcustom.Show();
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        var radius = 20;
        var rect = new RectangleGeometry(new Rect(0, 0, this.ActualWidth, this.ActualHeight), radius, radius);
        this.Clip = rect;

        this.SizeChanged += (s, ev) =>
        {
            rect.Rect = new Rect(0, 0, this.ActualWidth, this.ActualHeight);
```

```
    };
}

private void Expander_Expanded(object sender, RoutedEventArgs e)
{
    Expander expander = sender as Expander;
    if (expander != null)
    {
        Storyboard expandStoryboard = (Storyboard)FindResource("ExpandAnimation");
        expandStoryboard.Begin(expander);
    }
}

private void Expander_Collapsed(object sender, RoutedEventArgs e)
{
    Expander expander = sender as Expander;
    if (expander != null)
    {
        Storyboard collapseStoryboard = (Storyboard)FindResource("CollapseAnimation");
        collapseStoryboard.Begin(expander);
    }
}

private Border _msj;

private void ShowTypingBubble()
{
```

```
    if (_msj != null) return;

    TextBlock pct = new TextBlock
    {
        Text = "...",
        FontSize = 18,
        Foreground = Brushes.White,
        TextWrapping = TextWrapping.Wrap
    };

    _msj = new Border
    {
        Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#555")),
        CornerRadius = new CornerRadius(10),
        Padding = new Thickness(10),
        Margin = new Thickness(5),
        HorizontalAlignment = HorizontalAlignment.Left,
        MaxWidth = 100,
        Child = pct
    };

    ChatMessagesPanel.Children.Add(_msj);
}

private void RemoveTypingBubble()
{
    if (_msj != null)
```

```
    {
        ChatMessagesPanel.Children.Remove(_msj);
        _msj = null;
    }
}

private async Task charGPT(string message, bool isUser)
{
    var textBlock = new TextBlock
    {
        Text = "",
        FontSize = 18,
        Foreground = Brushes.White,
        TextWrapping = TextWrapping.Wrap
    };

    Border messageBubble = new Border
    {
        Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString(isUser ? "#2E8B57" : "#555")),
        CornerRadius = new CornerRadius(10),
        Padding = new Thickness(10),
        Margin = new Thickness(5),
        HorizontalAlignment = isUser ? HorizontalAlignment.Right : HorizontalAlignment.Left,
        MaxWidth = 500,
        Child = textBlock
    };
```

```
    ChatMessagesPanel.Children.Add(messageBubble);


    foreach (char c in message)
    {
        textBlock.Text += c;
        await Task.Delay(20);
    }
}



private void Send_Click(object sender, RoutedEventArgs e)
{
    string userMessage = ChatInputBox.Text.Trim();
    if (!string.IsNullOrEmpty(userMessage))
    {
        AddChatMessage(userMessage, isUser: true);
        ChatInputBox.Clear();
        GetAIResponse(userMessage);
    }
}


private void AddChatMessage(string message, bool isUser)
{
    Border messageBubble = new Border
    {
        Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString(isUser ? "#2E8B57" : "#555")),
        CornerRadius = new CornerRadius(10),
```

```
            Padding = new Thickness(10),
            Margin = new Thickness(5),
            HorizontalAlignment = isUser ? HorizontalAlignment.Right : HorizontalAlignment.Left,
            MaxWidth = 500,
            Child = new TextBlock
            {
                Text = message,
                FontSize = 18,
                Foreground = Brushes.White,
                TextWrapping = TextWrapping.Wrap
            }
        };

        ChatMessagesPanel.Children.Add(messageBubble);
    }

    private async void GetAIResponse(string userMessage)
    {
        string apiKey = "sk-proj-
J_wvSlTxelLHat4AdAGqD0ViZbxSubfIxwiBniXrO769ITZ80CJL9VtJvenp6g2fxk9dL9tj_vT3BlbkFJ5J4gc6kdh2wGHoJe2oO27zFXGqRT
ai08CP8-msaAWFhdERk6TAlZvuLbMv6xK090sflblTb0MA"; // replace with your API key
        string endpoint = "https://api.openai.com/v1/chat/completions";

        using HttpClient client = new HttpClient();
        client.DefaultRequestHeaders.Add("Authorization", $"Bearer {apiKey}");

        // Add user message to history
        _conversationHistory.Add(new Message { role = "user", content = userMessage });
```

```
var requestBody = new
{
    model = "gpt-4",
    messages = _conversationHistory
};

var jsonContent = JsonConvert.SerializeObject(requestBody);
var httpContent = new StringContent(jsonContent, Encoding.UTF8, "application/json");

try
{
    ShowTypingBubble();

    var response = await client.PostAsync(endpoint, httpContent);
    var result = await response.Content.ReadAsStringAsync();

    RemoveTypingBubble();

    if (!response.IsSuccessStatusCode)
    {
        AddChatMessage($"API Error: {response.StatusCode}\n{result}", isUser: false);
        return;
    }

    dynamic json = JsonConvert.DeserializeObject(result);
    string reply = json?.choices?[0]?.message?.content;
```

```
        if (!string.IsNullOrWhiteSpace(reply))
        {
            _conversationHistory.Add(new Message { role = "assistant", content = reply });
            await charGPT(reply.Trim(), isUser: false);
        }
        else
        {
            AddChatMessage("ERROR", isUser: false);
        }
    }
    catch (Exception ex)
    {
        RemoveTypingBubble();
        AddChatMessage("Error: " + ex.Message, isUser: false);
    }
}



private void SetupCarouselElements()
{
    Fact1Border = Fact1;
    Fact2Border = Fact2;
    Fact3Border = Fact3;
    Fact4Border = Fact4;
```

```
    Fact1TextBlock = Fact1Text;
    Fact2TextBlock = Fact2Text;
    Fact3TextBlock = Fact3Text;
    Fact4TextBlock = Fact4Text;
}

private void SetupInitialFacts()
{
    Fact1TextBlock.Text = GetRandomFunFact();
    Fact2TextBlock.Text = GetRandomFunFact();
    Fact3TextBlock.Text = GetRandomFunFact();
    Fact4TextBlock.Text = GetRandomFunFact();

    SetSlide(Fact1Border, -250, 200, 130, 0.6);
    SetSlide(Fact2Border, 0, 300, 200, 1.0);
    SetSlide(Fact3Border, 250, 200, 130, 0.6);
    SetSlide(Fact4Border, 500, 200, 130, 0.0);
}

private void StartCarousel()
{
    _carouselTimer = new DispatcherTimer();
    _carouselTimer.Interval = TimeSpan.FromSeconds(8);
    _carouselTimer.Tick += (s, e) => AnimateCarousel();
    _carouselTimer.Start();
}
```

```
private void AnimateCarousel()
{
    AnimateSlide(Fact1Border, -500);
    AnimateSlide(Fact2Border, -250);
    AnimateSlide(Fact3Border, 0);
    AnimateSlide(Fact4Border, 250);

    AnimateSize(Fact1Border, 200, 200, 130, 130);
    AnimateSize(Fact2Border, 300, 200, 200, 130);
    AnimateSize(Fact3Border, 200, 300, 130, 200);
    AnimateSize(Fact4Border, 200, 200, 130, 130);

    AnimateOpacity(Fact1Border, 0.6, 0.0);
    AnimateOpacity(Fact2Border, 1.0, 0.6);
    AnimateOpacity(Fact3Border, 0.6, 1.0);
    AnimateOpacity(Fact4Border, 0.0, 0.6);

    var resetTimer = new DispatcherTimer();
    resetTimer.Interval = TimeSpan.FromSeconds(0.8);
    resetTimer.Tick += (s, e) =>
    {
        resetTimer.Stop();
        ResetSlides();
    };
    resetTimer.Start();
}
```

```
private void open_games(object sender, RoutedEventArgs e)
{
    meniujocuri meniujocuri = new meniujocuri();
    meniujocuri.Show();
}

private void ResetSlides()
{
    SetSlide(Fact1Border, 500, 200, 130, 0.0);
    Fact1TextBlock.Text = GetRandomFunFact();

    var oldFact1 = Fact1Border;
    var oldFact1Text = Fact1TextBlock;

    Fact1Border = Fact2Border;
    Fact1TextBlock = Fact2TextBlock;

    Fact2Border = Fact3Border;
    Fact2TextBlock = Fact3TextBlock;

    Fact3Border = Fact4Border;
    Fact3TextBlock = Fact4TextBlock;

    Fact4Border = oldFact1;
    Fact4TextBlock = oldFact1Text;
}
```

```
private void AnimateSlide(Border fact, double toX)
{
   if (fact.RenderTransform is TranslateTransform tt)
   {
      var anim = new DoubleAnimation(tt.X, toX, TimeSpan.FromSeconds(0.8))
      {
         EasingFunction = new CubicEase { EasingMode = EasingMode.EaseInOut }
      };
      tt.BeginAnimation(TranslateTransform.XProperty, anim);
   }
}

private void AnimateSize(Border fact, double fromW, double toW, double fromH, double toH)
{
   var widthAnim = new DoubleAnimation(fromW, toW, TimeSpan.FromSeconds(0.8))
   {
      EasingFunction = new CubicEase { EasingMode = EasingMode.EaseInOut }
   };
   var heightAnim = new DoubleAnimation(fromH, toH, TimeSpan.FromSeconds(0.8))
   {
      EasingFunction = new CubicEase { EasingMode = EasingMode.EaseInOut }
   };
   fact.BeginAnimation(WidthProperty, widthAnim);
   fact.BeginAnimation(HeightProperty, heightAnim);
}

private void AnimateOpacity(Border fact, double from, double to)
```

```
{
    var anim = new DoubleAnimation(from, to, TimeSpan.FromSeconds(0.8));
    fact.BeginAnimation(OpacityProperty, anim);
}

private void SetSlide(Border fact, double x, double width, double height, double opacity)
{
    if (fact.RenderTransform is TranslateTransform tt)
        tt.X = x;
    fact.Width = width;
    fact.Height = height;
    fact.Opacity = opacity;
}

private string GetRandomFunFact()
{
    if (_funFacts.ContainsKey(language))
    {
        List<string> factsInCurrentLanguage = _funFacts[language];
        return factsInCurrentLanguage[_random.Next(factsInCurrentLanguage.Count)];
    }

    return _funFacts["English"][_random.Next(_funFacts["English"].Count)];
}

public string language = "English";
public bool infoOK = false;
```

```
main.misc.Info info;
string filePath;


public int curlang = 0;
private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
{
   if (curlang == 0)
   {
      ChangeLanguage("Romanian");
      language = "Romanian";
      DisplayWelcomeMessage();
      curlang = 1;
   }
   else
   {
      ChangeLanguage("English");
      language = "English";
      DisplayWelcomeMessage();
      curlang = 0;
   }
}
private void ChangeLanguage(string language)
{
   var resourceDictionary = new ResourceDictionary();
   if (language == "English")
      resourceDictionary.Source = new Uri("limbi/eng.xaml", UriKind.Relative);
   else if (language == "Romanian")
```

```
        resourceDictionary.Source = new Uri("limbi/ro.xaml", UriKind.Relative);


    Application.Current.Resources.MergedDictionaries.Clear();
    Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
    this.Dispatcher.Invoke(() => this.InvalidateVisual());
}
private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
    {
        this.DragMove();
    }
}


private void Minimize_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = WindowState.Minimized;
}


private void Close_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
private void showalgoMenu(object sender, RoutedEventArgs e)
{
    meniualgoritmi meniualgoritmi = new meniualgoritmi();
    meniualgoritmi.Show();
```

```
    }
    private void open_quiz(object sender, RoutedEventArgs e)
    {
        quiz quiz = new quiz();
        quiz.Show();
    }
    private void showmateMenu(object sender, RoutedEventArgs e)
    {
        meniumatematica meniumate = new meniumatematica();
        meniumate.Show();
    }
  }
}
```

5. Customizare cont

Din pagina acasă, utilizatorul prin apăsarea pozei de profil poate deschide meniul de customizare al contului, de unde își poate modifica toate aspectele legate de cont.

```
<Window x:Class="proiectoicd2025.account"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Customize Account" Height="600" Width="700"
        MinHeight="600" MinWidth="700"
        WindowStartupLocation="CenterScreen"
        Background="Transparent"
        AllowsTransparency="True"
        Closed="Window_Closed"
```

```
        WindowStyle="None">

    <Window.Resources>
        <Style TargetType="Button" x:Key="mouseOverStyle">
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Foreground" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Border Height="40" Background="#FF1E1E1E" VerticalAlignment="Top">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="" MouseDown="TitleBar_MouseDown" Foreground="White"
VerticalAlignment="Center" Margin="10,0,0,0" FontWeight="Bold" Height="40"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Minimize_Click"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
```

```
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Close_Click"/>

            </Grid>

        </Border>


        <Rectangle Fill="#FF290303" Width="{Binding ActualWidth,
RelativeSource={RelativeSource AncestorType=Window}}"

                   Height="{Binding ActualHeight, RelativeSource={RelativeSource
AncestorType=Window}}" Margin="0,40,0,0"/>


        <Viewbox Stretch="Uniform" Margin="0,40,0,10">

            <Grid Width="650" Height="570">

                <Grid.RowDefinitions>

                    <RowDefinition Height="0.5*"/>

                    <RowDefinition Height="2*"/>

                </Grid.RowDefinitions>


                <Button Content="ENG/ROM" FontSize="20" Foreground="White"
Background="Transparent" BorderThickness="0" Cursor="Hand"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="-16,4,0,0"
Click="ChangeLanguage_Click" Height="42" Width="130"/>

                <TextBlock Text="Customize Your Account" Grid.Row="0" FontSize="36"
FontWeight="Bold" Foreground="White" HorizontalAlignment="Center"
VerticalAlignment="Center"/>


                <Border Grid.Row="1" Background="#FF4A0A0A" CornerRadius="10"
Padding="30" MinWidth="300" MinHeight="200" HorizontalAlignment="Stretch"
VerticalAlignment="Center">

                    <StackPanel>

                        <TextBlock Text="Change Username" Foreground="White"
FontSize="18"/>

                        <TextBox x:Name="UsernameBox" Height="30" FontSize="16"
Background="#FF383838" Foreground="White" BorderThickness="0" Margin="0,5,0,15"
```

```
Padding="5,5,0,0"/>

                        <TextBlock Text="Change Email" Foreground="White"
FontSize="18"/>

                        <TextBox x:Name="EmailBox" Height="30" FontSize="16"
Background="#FF383838" Foreground="White" BorderThickness="0" Margin="0,5,0,15"
Padding="5,5,0,0"/>

                        <TextBlock Text="Change Password" Foreground="White"
FontSize="18"/>

                        <PasswordBox x:Name="PasswordBox" Height="30" FontSize="16"
Background="#FF383838" Foreground="White" BorderThickness="0" Margin="0,5,0,15"
Padding="5,4,0,0"/>



                        <TextBlock Text="Profile Picture" Foreground="White"
FontSize="18" Margin="0,0,0,5"/>

                        <StackPanel Orientation="Horizontal"
HorizontalAlignment="Left" Margin="0,0,0,15" Width="590">

                            <Image x:Name="ProfileImage"
Source="/resource/profilepics/user_icon.png" Width="96" Height="96"
Stretch="UniformToFill" ClipToBounds="True" Margin="0,0,10,0"/>

                            <Button Content="Upload New Picture" Width="150"
Height="30" Background="#39b359" Foreground="White" FontSize="14"
BorderThickness="0" Cursor="Hand" VerticalAlignment="Center"
Click="UploadPicture_Click"/>

                            <Button Content="Save Changes" Margin="80,0,0,0"
Width="250" Height="50" Background="#39b359" Foreground="White" FontSize="14"
BorderThickness="0" Cursor="Hand" VerticalAlignment="Center"
Click="SaveChanges_Click"/>

                        </StackPanel>

                    </StackPanel>

                </Border>

            </Grid>

        </Viewbox>

    </Grid>

</Window>
```

```
--
using System;
using System.IO;
using System.Windows.Media.Imaging;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Windows;
using Microsoft.Win32;
using proiectoicd2025.main.misc;
using System.Data.SqlClient;
using System.Windows.Input;

namespace proiectoicd2025
{
    public partial class account : Window
    {
        string username;
        private acasa acasaWin;
        public account(acasa acasaWindow)
        {
            InitializeComponent();
            username = manage.utilizatorCurent;
            acasaWin = acasaWindow;
            LoadProfileImage();
        }
```

```
        private void LoadProfileImage()
        {
            string tempImagePath = 
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource", 
"profilepics", "temp_profile_image.png");
            string imagePath = 
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource", 
"profilepics", $"{username}.png");

            string finalPath = File.Exists(tempImagePath) ? tempImagePath : 
imagePath;

            try
            {
                using (var stream = new FileStream(finalPath, FileMode.Open, 
FileAccess.Read, FileShare.ReadWrite))
                {
                    BitmapImage bmp = new BitmapImage();
                    bmp.BeginInit();
                    bmp.CacheOption = BitmapCacheOption.OnLoad;
                    bmp.StreamSource = stream;
                    bmp.EndInit();
                    bmp.Freeze();

                    ProfileImage.Source = null;
                    ProfileImage.Source = bmp;
                }
            }
            catch
            {
```

```
                ProfileImage.Source = new BitmapImage(new
Uri("resource/profilepics/user_icon.png", UriKind.Relative));
            }
        }


        private void Window_Closed(object sender, EventArgs e)
        {
            acasaWin.LoadProfileImage(true);
        }

        private static BitmapImage LoadBitmapImage(string fileName)
        {
            using (var stream = new FileStream(fileName, FileMode.Open,
FileAccess.Read))
            {
                var bitmapImage = new BitmapImage();
                bitmapImage.BeginInit();
                bitmapImage.CacheOption = BitmapCacheOption.OnLoad;
                bitmapImage.StreamSource = stream;
                bitmapImage.EndInit();
                bitmapImage.Freeze();
                return bitmapImage;
            }
        }

        private void UploadPicture_Click(object sender, RoutedEventArgs e)
        {
```

```
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";

            if (dialog.ShowDialog() == true)
            {
                string selectedFile = dialog.FileName;
                ResizeAndSaveImage(selectedFile);
                LoadProfileImage();
            }
        }

        private void ResizeAndSaveImage(string sourcePath)
        {
            string username = manage.utilizatorCurent;

            string tempSavePath =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource",
"profilepics", "temp_profile_image.png");

            try
            {
                using (var srcImage = new Bitmap(sourcePath))
                {
                    using (var newImage = new Bitmap(512, 512))
                    {
                        using (var graphics = Graphics.FromImage(newImage))
                        {
                            graphics.CompositingQuality =
CompositingQuality.HighQuality;
```

```
                            graphics.InterpolationMode =
InterpolationMode.HighQualityBicubic;
                            graphics.SmoothingMode = SmoothingMode.HighQuality;
                            graphics.DrawImage(srcImage, 0, 0, 512, 512);
                        }

                        using (FileStream fs = new FileStream(tempSavePath,
FileMode.Create, FileAccess.Write))
                        {
                            newImage.Save(fs, ImageFormat.Png);
                        }
                    }
                }



                LoadProfileImage();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }



        private void SaveChanges_Click(object sender, RoutedEventArgs e)
        {
```

```
            string newUsername = UsernameBox.Text.Trim();
            string newEmail = EmailBox.Text.Trim();
            string newPassword = PasswordBox.Password.Trim();

            if (string.IsNullOrWhiteSpace(newUsername) ||
                string.IsNullOrWhiteSpace(newEmail) ||
                string.IsNullOrWhiteSpace(newPassword))
            {
                citafis("ValidationError");
                return;
            }

            try
            {
                string encryptedCurrentUsername =
Criptografie.Cripteaza(manage.utilizatorCurent);
                string encryptedNewUsername = Criptografie.Cripteaza(newUsername);
                string encryptedNewEmail = Criptografie.Cripteaza(newEmail);
                string encryptedNewPassword = Criptografie.Cripteaza(newPassword);

                string dbPath =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Database",
"bazadate.mdf");
                string conString = $@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={dbPath};Integrated Security=True;";

                using SqlConnection con = new(conString);
                con.Open();
```

```
                using SqlCommand checkEmailCmd = new("SELECT COUNT(*) FROM
utilizatori WHERE email = @e AND [user] != @currentUser", con);

                checkEmailCmd.Parameters.AddWithValue("@e", encryptedNewEmail);

                checkEmailCmd.Parameters.AddWithValue("@currentUser",
encryptedCurrentUsername);


                int emailCount = (int)checkEmailCmd.ExecuteScalar();

                if (emailCount > 0)

                {

                    citafis("EmailTakenError");

                    return;

                }


                using SqlCommand updateCmd = new(@"UPDATE utilizatori

                                        SET [user] = @newUser, email = @newEmail,
parola = @newPass

                                        WHERE [user] = @currentUser", con);

                updateCmd.Parameters.AddWithValue("@newUser", encryptedNewUsername);

                updateCmd.Parameters.AddWithValue("@newEmail", encryptedNewEmail);

                updateCmd.Parameters.AddWithValue("@newPass", encryptedNewPassword);

                updateCmd.Parameters.AddWithValue("@currentUser",
encryptedCurrentUsername);


                int rowsAffected = updateCmd.ExecuteNonQuery();


                if (rowsAffected > 0)

                {

                    manage.utilizatorCurent = newUsername;
```

```
                    string baseDir =
System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "resource",
"profilepics");

                    string oldPic = Path.Combine(baseDir,
$"{Criptografie.Decripteaza(encryptedCurrentUsername)}.png");

                    string newPic = Path.Combine(baseDir, $"{newUsername}.png");


                    if (File.Exists(oldPic))

                    {

                        File.Move(oldPic, newPic, overwrite: true);

                    }


                    citafis("SaveChangesButton");


                    acasaWin.LoadProfileImage(true);


                    this.Close();

                }

                else

                {

                    citafis("NoChanges");

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Error: {ex.Message}", "Error",
MessageBoxButton.OK, MessageBoxImage.Error);

            }

        }
```

```
        public string language = "English";



        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }
        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            else if (language == "Romanian")
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
```

```
UriKind.Relative);

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
        private void citafis(string key)
        {
            string message = Application.Current.Resources[key] as string;
            MessageBox.Show(message);
        }
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
            {
                this.DragMove();
            }
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
```

```
        }
    }
}
```

--

6. Pagina de feedback

Utilizatorul poate trimite un mesaj anonim către un email prestabilit pentru a oferi feedback despre orice legat de aplicație.

```
<Window x:Class="proiectoicd2025.main.misc.contact"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Send Feedback"
        Height="400"
        Width="600"
        Background="Transparent"
        WindowStartupLocation="CenterScreen"
        ResizeMode="NoResize"
        Closed="Window_Closed"
        AllowsTransparency="True"
        WindowStyle="None">

    <Grid>
        <Border Height="40" Background="#FF1E1E1E" VerticalAlignment="Top">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
```

```
                <ColumnDefinition Width="Auto"/>

            </Grid.ColumnDefinitions>


            <TextBlock Text="" Foreground="White" VerticalAlignment="Center"
FontWeight="Bold" Height="40"/>


            <Button Grid.Column="1" Content="—" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Minimize_Click"/>

            <Button Grid.Column="3" Content="X" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Close_Click"/>

        </Grid>

    </Border>


    <Rectangle Fill="#FF290303"

               Width="{Binding ActualWidth, RelativeSource={RelativeSource
AncestorType=Window}}"

               Height="{Binding ActualHeight, RelativeSource={RelativeSource
AncestorType=Window}}"

               Margin="0,40,0,0"/>


    <Viewbox Stretch="Uniform" Margin="10,40,10,10">

        <Grid Width="550" Height="350">

            <Grid.RowDefinitions>

                <RowDefinition Height="Auto"/>

                <RowDefinition Height="*"/>

                <RowDefinition Height="Auto"/>

            </Grid.RowDefinitions>
```

```
                <TextBlock Text="{DynamicResource feedSend}"
                           FontSize="28"
                           FontWeight="Bold"
                           Foreground="White"
                           HorizontalAlignment="Center"
                           Margin="0,10,0,20"/>

                <Border Grid.Row="1"
                        Background="#FF4A0A0A"
                        CornerRadius="10"
                        Padding="20"
                        Margin="0,0,0,20">
                    <StackPanel>
                        <TextBlock Text="{DynamicResource feedCont}"
Foreground="White" FontSize="16" Margin="0,0,0,10"/>
                        <TextBox x:Name="FeedbackBox"
                                 AcceptsReturn="True"
                                 TextWrapping="Wrap"
                                 VerticalScrollBarVisibility="Auto"
                                 Height="150"
                                 FontSize="14"
                                 Background="#FF383838"
                                 Foreground="White"
                                 BorderThickness="0"
                                 Padding="10"
                                 HorizontalAlignment="Stretch">
                            <TextBox.Template>
                                <ControlTemplate TargetType="TextBox">
```

```
                                    <Border Background="{TemplateBinding
Background}" CornerRadius="8">
                                        <ScrollViewer x:Name="PART_ContentHost"/>
                                    </Border>
                                </ControlTemplate>
                            </TextBox.Template>
                        </TextBox>
                    </StackPanel>
                </Border>

                <Button Content="{DynamicResource send}"
                        Grid.Row="2"
                        Width="200"
                        Height="40"
                        HorizontalAlignment="Center"
                        Background="#39b359"
                        Foreground="White"
                        FontSize="16"
                        BorderThickness="0"
                        Click="SendFeedback_Click"
                        Cursor="Hand">
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Background="{TemplateBinding Background}"
CornerRadius="8">
                                <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
```

```
                    </Button.Template>
                    <Button.Style>
                        <Style TargetType="Button">
                            <Style.Triggers>
                                <Trigger Property="IsMouseOver" Value="True">
                                    <Setter Property="Background"
Value="DarkGoldenrod"/>
                                </Trigger>
                                <Trigger Property="IsPressed" Value="True">
                                    <Setter Property="Background"
Value="Goldenrod"/>
                                </Trigger>
                            </Style.Triggers>
                        </Style>
                    </Button.Style>
                </Button>
            </Grid>
        </Viewbox>
    </Grid>
</Window>

--
using System;
using System.Net;
using System.Net.Mail;
using System.Windows;
using System.Windows.Input;

namespace proiectoicd2025.main.misc
```

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

        private void Window_Closed(object sender, EventArgs e)
        {
            this.Close();
        }

        private void SendFeedback_Click(object sender, RoutedEventArgs e)
        {
            string message = FeedbackBox.Text.Trim();

            if (string.IsNullOrWhiteSpace(message))
            {
                MessageBox.Show("Please enter a message before sending.", "Empty
Message", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            try
            {
                string smtpServer = "smtp.gmail.com";
                int smtpPort = 587;
```

```
                string fromEmail = "yeeter9e7@gmail.com";
                string fromPassword = "dide edvc pfyn oopo";
                string toEmail = "alinbobea70@gmail.com";

                MailMessage mail = new MailMessage();
                mail.From = new MailAddress(fromEmail);
                mail.To.Add(toEmail);
                mail.Subject = "FEEDBACK RECURSOLVE";
                mail.Body = message;

                SmtpClient smtpClient = new SmtpClient(smtpServer, smtpPort)
                {
                    Credentials = new NetworkCredential(fromEmail, fromPassword),
                    EnableSsl = true
                };

                smtpClient.Send(mail);
                MessageBox.Show("Your feedback has been sent anonymously!",
"Success", MessageBoxButton.OK, MessageBoxImage.Information);
                FeedbackBox.Text = "";
            }
            catch (Exception ex)
            {
                MessageBox.Show("Failed to send message.\n\n" + ex.Message, "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
```

```
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.LeftButton == MouseButtonState.Pressed)
            {
                this.DragMove();
            }
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

SECȚIUNEA 3.2 Algoritmi

1. Copac binar

```
<Window x:Class="proiectoicd2025.main.algoritmi.copacbinar"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Binary Tree Traversals" Height="600" Width="800"
        Background="#1E1E1E" WindowStartupLocation="CenterScreen"
        WindowStyle="None" AllowsTransparency="True"
```

```
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
```

```
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="{DynamicResource titluCOPAC}" Foreground="White"
FontWeight="Bold" FontSize="16" Margin="10"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}"/>
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,0,5,0"/>
            </Grid>
        </Border>
```

```
        <Canvas x:Name="TreeCanvas" Grid.Row="1" Background="Black"/>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">

            <Button Content="{DynamicResource order1}" Click="Preorder_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource order2}" Click="Inorder_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource order3}" Click="Postorder_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using Path = System.IO.Path;
```

```
namespace proiectoicd2025.main.algoritmi
{
    public partial class copacbinar : Window
    {
        private class TreeNode
        {
            public int Value { get; set; }
            public TreeNode Left { get; set; }
            public TreeNode Right { get; set; }
            public Ellipse Ellipse { get; set; }
        }

        private TreeNode root;
        private const double NodeRadius = 20;
        private const double VerticalSpacing = 100;
        private const double HorizontalSpacing = 100;

        public string language = "English";
        public bool infoOK = false;
        public int curlang = 0;

        private main.misc.Info info;
        private string filePath;

        public copacbinar()
        {
            InitializeComponent();
```

```
    root = CreateTree();
    DrawTree();
}

private TreeNode CreateTree()
{
    return new TreeNode
    {
        Value = 1,
        Left = new TreeNode
        {
            Value = 2,
            Left = new TreeNode { Value = 4 },
            Right = new TreeNode { Value = 5 }
        },
        Right = new TreeNode
        {
            Value = 3,
            Left = new TreeNode { Value = 6 },
            Right = new TreeNode { Value = 7 }
        }
    };
}

private void DrawTree()
{
    TreeCanvas.Children.Clear();
```

```
    if (root != null)
    {
        DrawNode(root, 400, 50, 200);
    }
}

private void DrawNode(TreeNode node, double x, double y, double xSpacing)
{
    if (node == null)
        return;

    var ellipse = new Ellipse
    {
        Width = NodeRadius * 2,
        Height = NodeRadius * 2,
        Fill = Brushes.White,
        Stroke = Brushes.Red,
        StrokeThickness = 2
    };
    Canvas.SetLeft(ellipse, x - NodeRadius);
    Canvas.SetTop(ellipse, y - NodeRadius);
    TreeCanvas.Children.Add(ellipse);
    node.Ellipse = ellipse;

    var text = new TextBlock
    {
        Text = node.Value.ToString(),
```

```
            Foreground = Brushes.Red,
            FontWeight = FontWeights.Bold,
            FontSize = 16
        };
        Canvas.SetLeft(text, x - NodeRadius / 2);
        Canvas.SetTop(text, y - NodeRadius / 2);
        TreeCanvas.Children.Add(text);

        if (node.Left != null)
        {
            var line = new Line
            {
                X1 = x,
                Y1 = y + NodeRadius,
                X2 = x - xSpacing,
                Y2 = y + VerticalSpacing - NodeRadius,
                Stroke = Brushes.White,
                StrokeThickness = 2
            };
            TreeCanvas.Children.Add(line);

            DrawNode(node.Left, x - xSpacing, y + VerticalSpacing, xSpacing /
2);
        }

        if (node.Right != null)
        {
            var line = new Line
```

```
                {
                    X1 = x,
                    Y1 = y + NodeRadius,
                    X2 = x + xSpacing,
                    Y2 = y + VerticalSpacing - NodeRadius,
                    Stroke = Brushes.White,
                    StrokeThickness = 2
                };
                TreeCanvas.Children.Add(line);

                DrawNode(node.Right, x + xSpacing, y + VerticalSpacing, xSpacing /
2);
            }
        }

        private async Task PreorderTraversal(TreeNode node)
        {
            if (node == null) return;
            await HighlightNode(node);
            await PreorderTraversal(node.Left);
            await PreorderTraversal(node.Right);
        }

        private async Task InorderTraversal(TreeNode node)
        {
            if (node == null) return;
            await InorderTraversal(node.Left);
            await HighlightNode(node);
```

```
    await InorderTraversal(node.Right);
}

private async Task PostorderTraversal(TreeNode node)
{
    if (node == null) return;
    await PostorderTraversal(node.Left);
    await PostorderTraversal(node.Right);
    await HighlightNode(node);
}

private async Task HighlightNode(TreeNode node)
{
    node.Ellipse.Fill = Brushes.OrangeRed;
    await Task.Delay(500);
    node.Ellipse.Fill = Brushes.White;
}

private async void Preorder_Click(object sender, RoutedEventArgs e)
{
    await PreorderTraversal(root);
}

private async void Inorder_Click(object sender, RoutedEventArgs e)
{
    await InorderTraversal(root);
}
```

```
private async void Postorder_Click(object sender, RoutedEventArgs e)
{
    await PostorderTraversal(root);
}

private void Close_Click(object sender, RoutedEventArgs e)
{
    Close();
}

private void Minimize_Click(object sender, RoutedEventArgs e)
{
    WindowState = WindowState.Minimized;
}

private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        DragMove();
}

private void info_show(object sender, RoutedEventArgs e)
{
    if (!infoOK)
    {
        info = new main.misc.Info();
```

```
                filePath = string.Empty;


                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\copacbinarEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\copacbinarRO.txt");
                }


                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }


        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
```

```
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();

            if (language == "English")
            {
                this.language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                this.language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }
```

```
            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

2. Algoritmul lui Dijkstra

```
<Window x:Class="proiectoicd2025.main.algoritmi.djik"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Dijkstra's Algorithm" Height="600" Width="800"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
```

```
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
```

```
        </Grid.RowDefinitions>


        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,481" Grid.RowSpan="2">

            <Grid MouseDown="TitleBar_MouseDown">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition/>

                    <ColumnDefinition Width="Auto"/>

                    <ColumnDefinition Width="Auto"/>

                </Grid.ColumnDefinitions>


                <TextBlock Text="{DynamicResource titluDjik}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>

            </Grid>

        </Border>


        <Canvas Name="GraphCanvas" Grid.Row="1" Background="Black"
ClipToBounds="True" Margin="10" />


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10,0,10">

            <Button Content="{DynamicResource addDjik}" Click="AddNode_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource connectDjik}"
Click="ConnectMode_Click" Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
```

```
ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

using System.IO;

using Path = System.IO.Path;


namespace proiectoicd2025.main.algoritmi

{

    public partial class djik : Window

    {

        private Dictionary<int, Point> nodePositions = new();

        private Dictionary<int, Ellipse> nodeVisuals = new();

        private Dictionary<int, List<(int neighbor, int weight)>> graph = new();
```

```
        private Dictionary<(int, int), Line> edgeVisuals = new();
        private Dictionary<(int, int), TextBlock> edgeLabels = new();

        private int nodeCounter = 0;
        private bool connectMode = false;
        private int? selectedNode = null;

        private int? startNode = null;
        private int? endNode = null;

        public djik()
        {
            InitializeComponent();
            GraphCanvas.MouseLeftButtonDown += GraphCanvas_MouseLeftButtonDown;
        }

        private void GraphCanvas_MouseLeftButtonDown(object sender,
MouseButtonEventArgs e)
        {
            if (!connectMode)
            {
                var position = e.GetPosition(GraphCanvas);
                AddNode(position);
            }
        }

        private void AddNode(Point position)
        {
```

```
    int id = nodeCounter++;
    nodePositions[id] = position;
    graph[id] = new();

    var ellipse = new Ellipse
    {
        Width = 30,
        Height = 30,
        Fill = Brushes.White,
        Stroke = Brushes.Red,
        StrokeThickness = 2,
        Tag = id
    };
    Canvas.SetLeft(ellipse, position.X - 15);
    Canvas.SetTop(ellipse, position.Y - 15);
    ellipse.MouseLeftButtonDown += Node_Click;

    nodeVisuals[id] = ellipse;
    GraphCanvas.Children.Add(ellipse);
}

private void Node_Click(object sender, MouseButtonEventArgs e)
{
    e.Handled = true;

    var node = (int)((Ellipse)sender).Tag;
```

```
if (connectMode)
{
    if (selectedNode == null)
    {
        selectedNode = node;
        nodeVisuals[node].Fill = Brushes.Orange;
    }
    else
    {
        if (selectedNode != node)
        {
            int weight = new Random().Next(1, 10);
            graph[selectedNode.Value].Add((node, weight));
            graph[node].Add((selectedNode.Value, weight));

            much(selectedNode.Value, node, weight);
        }

        nodeVisuals[selectedNode.Value].Fill = Brushes.White;
        selectedNode = null;
    }
}
else
{
    if (startNode == null)
    {
        startNode = node;
```

```
            nodeVisuals[node].Fill = Brushes.Lime;
        }
        else if (endNode == null)
        {
            endNode = node;
            nodeVisuals[node].Fill = Brushes.Red;
        }
    }
}

private void much(int from, int to, int weight)
{
    var line = new Line
    {
        X1 = nodePositions[from].X,
        Y1 = nodePositions[from].Y,
        X2 = nodePositions[to].X,
        Y2 = nodePositions[to].Y,
        Stroke = Brushes.White,
        StrokeThickness = 2
    };

    GraphCanvas.Children.Add(line);
    edgeVisuals[(from, to)] = line;
    edgeVisuals[(to, from)] = line;

    var midX = (line.X1 + line.X2) / 2;
```

```
    var midY = (line.Y1 + line.Y2) / 2;

    var label = new TextBlock
    {
        Text = weight.ToString(),
        Foreground = Brushes.White,
        FontWeight = FontWeights.Bold,
        Background = Brushes.Black,
        Padding = new Thickness(2)
    };

    Canvas.SetLeft(label, midX);
    Canvas.SetTop(label, midY);

    GraphCanvas.Children.Add(label);
    edgeLabels[(from, to)] = label;
    edgeLabels[(to, from)] = label;
}


private void ConnectMode_Click(object sender, RoutedEventArgs e)
{
    connectMode = !connectMode;
    selectedNode = null;
}

private async void Solve_Click(object sender, RoutedEventArgs e)
```

```
{
    if (startNode == null || endNode == null) return;

    var distances = new Dictionary<int, int>();
    var previous = new Dictionary<int, int?>();
    var queue = new List<int>();

    foreach (var node in graph.Keys)
    {
        distances[node] = int.MaxValue;
        previous[node] = null;
        queue.Add(node);
    }

    distances[startNode.Value] = 0;

    while (queue.Count > 0)
    {
        var current = queue.OrderBy(n => distances[n]).First();
        queue.Remove(current);

        if (current == endNode.Value) break;

        nodeVisuals[current].Fill = Brushes.Yellow;
        await Task.Delay(300);

        foreach (var (neighbor, weight) in graph[current])
```

```
        {
            int alt = distances[current] + weight;
            if (alt < distances[neighbor])
            {
                distances[neighbor] = alt;
                previous[neighbor] = current;
            }
        }

        nodeVisuals[current].Fill = Brushes.Gray;
    }

    int? pathNode = endNode;
    while (previous[pathNode.Value] != null)
    {
        int from = previous[pathNode.Value].Value;
        int to = pathNode.Value;

        edgeVisuals[(from, to)].Stroke = Brushes.Lime;
        await Task.Delay(300);

        pathNode = from;
    }
}

private void Reset_Click(object sender, RoutedEventArgs e)
{
```

```
    GraphCanvas.Children.Clear();
    graph.Clear();
    nodePositions.Clear();
    nodeVisuals.Clear();
    edgeVisuals.Clear();
    nodeCounter = 0;
    selectedNode = null;
    startNode = null;
    endNode = null;
    edgeLabels.Clear();
}

private void AddNode_Click(object sender, RoutedEventArgs e)
{
    connectMode = false;
}

private void Minimize_Click(object sender, RoutedEventArgs e)
{
    WindowState = WindowState.Minimized;
}

private void Close_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

```
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                DragMove();
        }
        public string language = "English";
        public bool infoOK = false;
        main.misc.Info info;
        string filePath;
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\djikEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\djikRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
```

```
    }
    else
    {
        info.Close();
        infoOK = false;
    }
}

public int curlang = 0;
private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
{
    if (curlang == 0)
    {
        ChangeLanguage("Romanian");
        language = "Romanian";
        curlang = 1;
    }
    else
    {
        ChangeLanguage("English");
        language = "English";
        curlang = 0;
    }
}

private void ChangeLanguage(string language)
{
```

```
            var resourceDictionary = new ResourceDictionary();

            if (language == "English")

            {

                language = "English";

                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);

            }

            else if (language == "Romanian")

            {

                language = "Romanian";

                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);

            }


            Application.Current.Resources.MergedDictionaries.Clear();


Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

            this.Dispatcher.Invoke(() => this.InvalidateVisual());

        }

    }

}
```

3. Umplere tip paint

```
<Window x:Class="proiectoicd2025.main.algoritmi.flood"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        Title="Flood Fill" Height="600" Width="800"

        Background="#1E1E1E" WindowStartupLocation="CenterScreen"

        WindowStyle="None" AllowsTransparency="True"

        ResizeMode="NoResize">
```

```
    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
```

```
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="{DynamicResource titluFlood}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}"/>
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,0,5,0"/>
            </Grid>
        </Border>
```

```
        <Canvas x:Name="GridCanvas" Grid.Row="1" Background="Black"
MouseLeftButtonDown="Canvas_MouseLeftButtonDown"/>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">

            <Button Content="{DynamicResource generate}" Click="GenerateGrid_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.IO;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using Path = System.IO.Path;


namespace proiectoicd2025.main.algoritmi

{

    public partial class flood : Window

    {
```

```
private const int Rows = 20;
private const int Columns = 20;
private const int CellSize = 20;
private int dynamicRows;
private int dynamicCols;

private Rectangle[,] cells = new Rectangle[Rows, Columns];
private Brush fillColor = Brushes.OrangeRed;

public flood()
{
    InitializeComponent();
    GenerateGrid();
}

private void GenerateGrid()
{
    GridCanvas.Children.Clear();

    int availableWidth = (int)GridCanvas.ActualWidth;
    int availableHeight = (int)GridCanvas.ActualHeight;

    dynamicRows = availableHeight / CellSize;
    dynamicCols = availableWidth / CellSize;

    cells = new Rectangle[dynamicRows, dynamicCols];
    var rand = new Random();
```

```
            for (int row = 0; row < dynamicRows; row++)
            {
                for (int col = 0; col < dynamicCols; col++)
                {
                    var rect = new Rectangle
                    {
                        Width = CellSize - 1,
                        Height = CellSize - 1,
                        Fill = rand.NextDouble() < 0.7 ? Brushes.White :
Brushes.Gray
                    };

                    Canvas.SetLeft(rect, col * CellSize);
                    Canvas.SetTop(rect, row * CellSize);
                    GridCanvas.Children.Add(rect);
                    cells[row, col] = rect;
                }
            }
        }



        private async void Canvas_MouseLeftButtonDown(object sender,
MouseButtonEventArgs e)
        {
            Point click = e.GetPosition(GridCanvas);
            int row = (int)(click.Y / CellSize);
```

```
    int col = (int)(click.X / CellSize);

    if (IsInBounds(row, col))
    {
        var targetColor = cells[row, col].Fill;
        if (targetColor != fillColor)
        {
            await FloodFill(row, col, targetColor);
        }
    }
}

private async Task FloodFill(int row, int col, Brush targetColor)
{
    if (!IsInBounds(row, col)) return;
    if (cells[row, col].Fill != targetColor) return;

    cells[row, col].Fill = fillColor;
    await Task.Delay(10);

    await FloodFill(row + 1, col, targetColor);
    await FloodFill(row - 1, col, targetColor);
    await FloodFill(row, col + 1, targetColor);
    await FloodFill(row, col - 1, targetColor);
}

private bool IsInBounds(int row, int col)
```

```
        {
            return row >= 0 && row < dynamicRows && col >= 0 && col < dynamicCols;
        }



        private void GenerateGrid_Click(object sender, RoutedEventArgs e)
        {
            GenerateGrid();
        }


        private void Close_Click(object sender, RoutedEventArgs e) => Close();


        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;


        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                DragMove();
        }
        public string language = "English";
        public bool infoOK = false;
        main.misc.Info info;
        string filePath;
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
```

```
                info = new main.misc.Info();

                filePath = string.Empty;

                if (language == "English")

                {

                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\floodEN.txt");

                }

                else if (language == "Romanian")

                {

                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\floodRO.txt");

                }

                info.SetContentFromFile(filePath);

                info.Show();

                infoOK = true;

            }

            else

            {

                info.Close();

                infoOK = false;

            }

        }


        public int curlang = 0;

        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)

        {

            if (curlang == 0)

            {
```

```
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }
```

```
            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

4. Generare procedurală

```
<Window x:Class="proiectoicd2025.main.algoritmi.generare"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Fractal Terrain Generator" Height="600" Width="700"
        Background="#1E1E1E" WindowStartupLocation="CenterScreen"
        WindowStyle="None" AllowsTransparency="True"
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
```

```
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
```

```
BorderThickness="0,0,0,2">

            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition/>

                    <ColumnDefinition Width="Auto"/>

                    <ColumnDefinition Width="Auto"/>

                </Grid.ColumnDefinitions>

                <TextBlock Text="Fractal Terrain Generator" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,0,5,0"/>

            </Grid>

        </Border>


        <Canvas x:Name="GridCanvas" Grid.Row="1" Background="Black"
Margin="0,0,0,10"/>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">

            <Button Content="{DynamicResource generate}" Click="Start_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
```

```
using System;
using System.IO;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;
using Path = System.IO.Path;

namespace proiectoicd2025.main.algoritmi
{
    public partial class generare : Window
    {
        private const int Rows = 40;
        private const int Columns = 50;
        private const int CellSize = 12;

        private char[,] map;
        private Random random = new Random();

        public generare()
        {
            InitializeComponent();
            GenerateMap();
        }
```

```
        private bool[,] visited;

        private async void GenerateMap()
        {
            map = new char[Rows, Columns];
            visited = new bool[Rows, Columns];

            for (int r = 0; r < Rows; r++)
                for (int c = 0; c < Columns; c++)
                    map[r, c] = '#';

            for (int r = 1; r < Rows; r += 2)
                for (int c = 1; c < Columns; c += 2)
                    map[r, c] = ' ';

            GridCanvas.Children.Clear();

            int startRow = (Rows / 2) | 1;
            int startCol = (Columns / 2) | 1;

            await Carve(startRow, startCol);

            for (int r = 0; r < Rows; r++)
                for (int c = 0; c < Columns; c++)
                    DrawCell(r, c, map[r, c] == '#' ? Brushes.Black :
Brushes.White);
        }
```

```
private async Task Carve(int row, int col)
{
    visited[row, col] = true;

    var directions = new (int dr, int dc)[]
    {
(-2, 0), (2, 0), (0, -2), (0, 2)
    };
    Shuffle(directions);

    foreach (var (dr, dc) in directions)
    {
        int newRow = row + dr;
        int newCol = col + dc;

        if (InBounds(newRow, newCol) && !visited[newRow, newCol])
        {
            map[row + dr / 2, col + dc / 2] = '.';
            map[newRow, newCol] = '.';

            DrawCell(row + dr / 2, col + dc / 2, Brushes.White);
            DrawCell(newRow, newCol, Brushes.White);

            await Task.Delay(2);
            await Carve(newRow, newCol);
        }
    }
```

```
}
private bool InBounds(int r, int c) =>
    r > 0 && r < Rows - 1 && c > 0 && c < Columns - 1;

private void DrawCell(int row, int col, Brush color)
{
    var rect = new Rectangle
    {
        Width = CellSize,
        Height = CellSize,
        Fill = color
    };
    Canvas.SetLeft(rect, col * CellSize);
    Canvas.SetTop(rect, row * CellSize);
    GridCanvas.Children.Add(rect);
}



private void Shuffle((int, int)[] array)
{
    for (int i = array.Length - 1; i > 0; i--)
    {
        int j = random.Next(i + 1);
        (array[i], array[j]) = (array[j], array[i]);
    }
}
```

```
        private void Start_Click(object sender, RoutedEventArgs e)
        {
            GridCanvas.Children.Clear();
            GenerateMap();
        }

        private void Close_Click(object sender, RoutedEventArgs e) => Close();
        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                DragMove();
        }

        public string language = "English";
        public bool infoOK = false;
        main.misc.Info info;
        string filePath;

        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
```

```
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\generareEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\generareRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }

        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
```

```
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            else
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

5. Drum Hamiltonian

```
<Window x:Class="proiectoicd2025.main.algoritmi.hamiltonian"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Hamiltonian Path" Height="600" Width="800"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
```

```
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
```

```
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>

            <TextBlock Text="{DynamicResource titluHamiltonian}"
Foreground="White" FontWeight="Bold" FontSize="16" VerticalAlignment="Center"
Margin="10,0"/>
            <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>
            <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>
        </Grid>
    </Border>

    <Canvas Name="GraphCanvas" Grid.Row="1" Background="Black"
ClipToBounds="True" Margin="10" />

    <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10,0,10">
        <Button Content="{DynamicResource addDjik}" Click="AddNode_Click"
Style="{StaticResource ThemedButtonStyle}"/>
        <Button Content="{DynamicResource connectDjik}"
Click="ConnectMode_Click" Style="{StaticResource ThemedButtonStyle}"/>
        <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}"/>
        <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
ThemedButtonStyle}"/>
        <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>
        <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>
    </StackPanel>
</Grid>
</Window>
```

```
--
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;
using Path = System.IO.Path;

namespace proiectoicd2025.main.algoritmi
{
    public partial class hamiltonian : Window
    {
        private int nodeCounter = 0;
        private bool isConnectMode = false;
        private Ellipse selectedNode = null;

        private Dictionary<Ellipse, int> nodeIds = new();
        private Dictionary<int, Point> nodePositions = new();
        private Dictionary<int, List<int>> graph = new();

        public hamiltonian()
        {
            InitializeComponent();
```

```
}

private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        DragMove();
}

private void Minimize_Click(object sender, RoutedEventArgs e)
{
    WindowState = WindowState.Minimized;
}

private void Close_Click(object sender, RoutedEventArgs e)
{
    Close();
}

private void AddNode_Click(object sender, RoutedEventArgs e)
{
    Random rand = new Random();
    double x = rand.Next(50, (int)GraphCanvas.ActualWidth - 50);
    double y = rand.Next(50, (int)GraphCanvas.ActualHeight - 50);

    Ellipse node = new()
    {
        Width = 30,
```

```
            Height = 30,
            Fill = Brushes.Red,
            Stroke = Brushes.White,
            StrokeThickness = 2,
            Tag = nodeCounter
        };

        node.MouseLeftButtonDown += Node_Click;
        GraphCanvas.Children.Add(node);
        Canvas.SetLeft(node, x);
        Canvas.SetTop(node, y);

        nodeIds[node] = nodeCounter;
        nodePositions[nodeCounter] = new Point(x + 15, y + 15);
        graph[nodeCounter] = new List<int>();
        nodeCounter++;
    }

    private void ConnectMode_Click(object sender, RoutedEventArgs e)
    {
        isConnectMode = !isConnectMode;
        selectedNode = null;
    }

    private void Node_Click(object sender, MouseButtonEventArgs e)
    {
        if (!isConnectMode) return;
```

```
var clickedNode = sender as Ellipse;
if (selectedNode == null)
{
    selectedNode = clickedNode;
    clickedNode.Stroke = Brushes.Yellow;
}
else
{
    if (clickedNode == selectedNode)
    {
        selectedNode.Stroke = Brushes.White;
        selectedNode = null;
        return;
    }

    int id1 = nodeIds[selectedNode];
    int id2 = nodeIds[clickedNode];

    if (!graph[id1].Contains(id2))
    {
        graph[id1].Add(id2);
        graph[id2].Add(id1);

        var line = new Line
        {
            X1 = nodePositions[id1].X,
```

```
                Y1 = nodePositions[id1].Y,
                X2 = nodePositions[id2].X,
                Y2 = nodePositions[id2].Y,
                Stroke = Brushes.White,
                StrokeThickness = 2
            };
            GraphCanvas.Children.Insert(0, line);
        }

        selectedNode.Stroke = Brushes.White;
        selectedNode = null;
    }
}

private async void Solve_Click(object sender, RoutedEventArgs e)
{
    var path = new List<int>();
    foreach (var node in graph.Keys)
    {
        path.Clear();
        var visited = new HashSet<int>();
        if (await FindHamiltonianPathVisual(node, visited, path))
        {
            return;
        }
    }
```

```
            //MessageBox.Show("nu gasit");
        }


        private async Task<bool> FindHamiltonianPathVisual(int current, HashSet<int>
visited, List<int> path)
        {
            visited.Add(current);
            path.Add(current);

            if (visited.Count == graph.Count)
            {
                await HighlightPathVisual(path);
                return true;
            }

            foreach (var neighbor in graph[current])
            {
                if (!visited.Contains(neighbor))
                {
                    if (await FindHamiltonianPathVisual(neighbor, visited, path))
                        return true;
                }
            }

            visited.Remove(current);
            path.Remove(current);
            return false;
        }
```

```
        private bool FindHamiltonianPath(int current, HashSet<int> visited,
List<int> path)
        {
            visited.Add(current);
            path.Add(current);

            if (visited.Count == graph.Count)
                return true;

            foreach (var neighbor in graph[current])
            {
                if (!visited.Contains(neighbor))
                {
                    if (FindHamiltonianPath(neighbor, visited, path))
                        return true;
                }
            }

            visited.Remove(current);
            path.Remove(current);
            return false;
        }

        private async Task HighlightPathVisual(List<int> path)
        {
            foreach (UIElement element in GraphCanvas.Children)
```

```
{
    if (element is Line line)
        line.Stroke = Brushes.White;
}

for (int i = 0; i < path.Count - 1; i++)
{
    int start = path[i];
    int end = path[i + 1];

    foreach (UIElement element in GraphCanvas.Children)
    {
        if (element is Line line)
        {
            if ((Math.Abs(line.X1 - nodePositions[start].X) < 1 &&
                 Math.Abs(line.Y1 - nodePositions[start].Y) < 1 &&
                 Math.Abs(line.X2 - nodePositions[end].X) < 1 &&
                 Math.Abs(line.Y2 - nodePositions[end].Y) < 1) ||
                (Math.Abs(line.X1 - nodePositions[end].X) < 1 &&
                 Math.Abs(line.Y1 - nodePositions[end].Y) < 1 &&
                 Math.Abs(line.X2 - nodePositions[start].X) < 1 &&
                 Math.Abs(line.Y2 - nodePositions[start].Y) < 1))
            {
                line.Stroke = Brushes.Lime;
                await Task.Delay(500);
                break;
            }
```

```
            }
        }
    }
}



private void Reset_Click(object sender, RoutedEventArgs e)
{
    GraphCanvas.Children.Clear();
    nodeCounter = 0;
    nodeIds.Clear();
    nodePositions.Clear();
    graph.Clear();
    selectedNode = null;
}
public string language = "English";
public bool infoOK = false;
main.misc.Info info;
string filePath;
private void info_show(object sender, RoutedEventArgs e)
{
    if (!infoOK)
    {
        info = new main.misc.Info();
        filePath = string.Empty;
        if (language == "English")
        {
            filePath = Path.Combine(Directory.GetCurrentDirectory(),
```

```
@"resource\info\algoritmi\hamiltonianEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\hamiltonianRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }

        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
```

```
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
```

```
}
```

6. Turnurile din Hanoi

```
<Window x:Class="proiectoicd2025.main.algoritmi.hanoi"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Tower of Hanoi" Height="600" Width="800"
        Background="#1E1E1E" WindowStartupLocation="CenterScreen"
        WindowStyle="None" AllowsTransparency="True"
        ResizeMode="NoResize">


    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
```

```
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
```

```
                    <ColumnDefinition Width="Auto"/>

                </Grid.ColumnDefinitions>

                <TextBlock Text="Hanoi" Foreground="White" FontWeight="Bold"
FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,0,5,0"/>

            </Grid>

        </Border>



        <Canvas Name="HanoiCanvas" Grid.Row="1" Background="Black"/>



        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">

            <Button Content="Start" Click="Start_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

using System.Windows;
```

```
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;
using Path = System.IO.Path;

namespace proiectoicd2025.main.algoritmi
{
    public partial class hanoi : Window
    {
        private const int DiskCount = 6;
        private readonly Stack<Rectangle>[] towers = new Stack<Rectangle>[3];
        private readonly double towerWidth = 10;
        private readonly double towerHeight = 200;
        private readonly double canvasWidth = 800;
        private readonly double canvasHeight = 500;

        public hanoi()
        {
            InitializeComponent();
            InitializeTowers();
        }

        private void InitializeTowers()
        {
            HanoiCanvas.Children.Clear();
            for (int i = 0; i < 3; i++)
```

```
            {
                towers[i] = new Stack<Rectangle>();

                var rod = new Rectangle
                {
                    Width = towerWidth,
                    Height = towerHeight,
                    Fill = Brushes.White
                };

                double x = (i + 1) * canvasWidth / 4 - towerWidth / 2;
                Canvas.SetLeft(rod, x);
                Canvas.SetTop(rod, canvasHeight - towerHeight);
                HanoiCanvas.Children.Add(rod);
            }

            for (int i = DiskCount; i > 0; i--)
            {
                var disk = new Rectangle
                {
                    Height = 20,
                    Width = 30 + i * 20,
                    Fill = new SolidColorBrush(Color.FromRgb((byte)(255 - i * 20),
(byte)(50 + i * 10), 0)),
                    RadiusX = 6,
                    RadiusY = 6
                };
                towers[0].Push(disk);
```

```
            HanoiCanvas.Children.Add(disk);
        }

        DrawDisks();
    }

    private void DrawDisks()
    {
        for (int i = 0; i < 3; i++)
        {
            var stack = towers[i];
            double baseX = (i + 1) * canvasWidth / 4;
            int level = 0;

            foreach (var disk in stack.Reverse())
            {
                Canvas.SetLeft(disk, baseX - disk.Width / 2);
                Canvas.SetTop(disk, canvasHeight - 20 - level * 22);
                Panel.SetZIndex(disk, 1);
                level++;
            }
        }
    }

    private async void Start_Click(object sender, RoutedEventArgs e)
    {
```

```
            await SolveHanoi(DiskCount, 0, 2, 1);
        }

        private async Task SolveHanoi(int n, int from, int to, int aux)
        {
            if (n == 0) return;

            await SolveHanoi(n - 1, from, aux, to);

            var disk = towers[from].Pop();
            towers[to].Push(disk);
            DrawDisks();
            await Task.Delay(300);

            await SolveHanoi(n - 1, aux, to, from);
        }

        private void Close_Click(object sender, RoutedEventArgs e) => Close();

        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                DragMove();
        }
        public string language = "English";
```

```
        public bool infoOK = false;

        main.misc.Info info;

        string filePath;

        private void info_show(object sender, RoutedEventArgs e)

        {

            if (!infoOK)

            {

                info = new main.misc.Info();

                filePath = string.Empty;

                if (language == "English")

                {

                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\hanoiEN.txt");

                }

                else if (language == "Romanian")

                {

                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\hanoiRO.txt");

                }

                info.SetContentFromFile(filePath);

                info.Show();

                infoOK = true;

            }

            else

            {

                info.Close();

                infoOK = false;

            }
```

```
        }

        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
```

```
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

7. Maze solver

```
<Window x:Class="proiectoicd2025.main.algoritmi.mazesolver"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Maze Solver" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="CanResizeWithGrip">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
```

```
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
        <Style x:Key="CellButtonStyle" TargetType="Border">
```

```
            <Setter Property="Width" Value="40"/>
            <Setter Property="Height" Value="40"/>
            <Setter Property="Margin" Value="1"/>
            <Setter Property="Background" Value="DimGray"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="1"/>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,501" Grid.RowSpan="2">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="{DynamicResource titluMaze}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>
```

```
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>

            </Grid>

        </Border>


        <UniformGrid Name="MazeGrid" Grid.Row="1" Rows="10" Columns="10"
Margin="0,10" />


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Bottom">

            <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System.IO;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;


namespace proiectoicd2025.main.algoritmi
```

```
{
    public partial class mazesolver : Window
    {
        private Border[,] cells = new Border[10, 10];
        private int[,] maze = new int[10, 10];
        private bool[,] visited = new bool[10, 10];

        public mazesolver()
        {
            InitializeComponent();
            GenerateMazeGrid();
            GenerateMaze();
        }

        private void GenerateMazeGrid()
        {
            MazeGrid.Children.Clear();
            for (int row = 0; row < 10; row++)
            {
                for (int col = 0; col < 10; col++)
                {
                    Border cell = new Border
                    {
                        Style = (Style)FindResource("CellButtonStyle"),
                        Background = Brushes.DimGray
                    };
                    MazeGrid.Children.Add(cell);
```

```
                cells[row, col] = cell;
            }
        }
    }

    private void GenerateMaze()
    {
        var rand = new Random();
        for (int row = 0; row < 10; row++)
        {
            for (int col = 0; col < 10; col++)
            {
                maze[row, col] = rand.Next(0, 100) < 25 ? 1 : 0;
                visited[row, col] = false;
                cells[row, col].Background = maze[row, col] == 1 ?
Brushes.Black : Brushes.DimGray;
            }
        }

        maze[0, 0] = maze[9, 9] = 0;
        cells[0, 0].Background = Brushes.Green;
        cells[9, 9].Background = Brushes.Red;
    }

    private async void Solve_Click(object sender, RoutedEventArgs e)
    {
        await SolveMaze(0, 0);
    }
```

```
private async Task<bool> SolveMaze(int row, int col)
{
    if (row < 0 || col < 0 || row >= 10 || col >= 10)
        return false;
    if (maze[row, col] == 1 || visited[row, col])
        return false;

    visited[row, col] = true;

    if (!(row == 0 && col == 0) && !(row == 9 && col == 9))
        cells[row, col].Background = Brushes.Orange;

    await Task.Delay(50);

    if (row == 9 && col == 9)
        return true;

    int[] dx = { 0, 1, 0, -1 };
    int[] dy = { 1, 0, -1, 0 };

    for (int i = 0; i < 4; i++)
    {
        if (await SolveMaze(row + dx[i], col + dy[i]))
            return true;
    }
```

```
            if (!(row == 0 && col == 0) && !(row == 9 && col == 9))
                cells[row, col].Background = Brushes.DarkSlateGray;

            return false;
        }

        private void Reset_Click(object sender, RoutedEventArgs e)
        {
            GenerateMaze();
        }

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;

        private void Close_Click(object sender, RoutedEventArgs e) => Close();
        public string language = "English";
        public bool infoOK = false;
        main.misc.Info info;
        string filePath;
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
```

```
            {
                info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\mazesolverEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\mazesolverRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }

        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
```

```
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }
```

```
            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

8. Sortare prin interclasare

```
<Window x:Class="proiectoicd2025.main.algoritmi.sort"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Merge Sort" Height="591" Width="787"
        Background="#1E1E1E" WindowStartupLocation="CenterScreen"
        WindowStyle="None" AllowsTransparency="True" ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
```

```
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
```

```
BorderThickness="0,0,0,2">

            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition/>

                    <ColumnDefinition Width="Auto"/>

                    <ColumnDefinition Width="Auto"/>

                </Grid.ColumnDefinitions>

                <TextBlock Text="{DynamicResource titluSort}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,0,5,0"/>

            </Grid>

        </Border>


        <Canvas x:Name="BarCanvas" Grid.Row="1" Background="Black" />


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10,0,10">

            <Button Content="{DynamicResource generate}" Click="GenerateBars_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Sort" Click="SortBars_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>
```

```
    </Grid>
</Window>

--
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Shapes;
using Path = System.IO.Path;

namespace proiectoicd2025.main.algoritmi
{
    public partial class sort : Window
    {
        private int BarCount = 50;
        private List<Rectangle> bars = new();
        private List<int> values = new();

        public sort()
        {
            InitializeComponent();
            Loaded += (_, _) => GenerateBars();
```

```
            SizeChanged += (_, _) => GenerateBars();
        }

        private void GenerateBars_Click(object sender, RoutedEventArgs e) =>
GenerateBars();

        private void GenerateBars()
        {
            if (BarCanvas.ActualWidth == 0 || BarCanvas.ActualHeight == 0) return;

            BarCanvas.Children.Clear();
            bars.Clear();
            values.Clear();

            double spacing = 2;
            double barWidth = (BarCanvas.ActualWidth - spacing * (BarCount - 1)) /
BarCount;
            double canvasHeight = BarCanvas.ActualHeight;
            var rand = new Random();

            for (int i = 0; i < BarCount; i++)
            {
                int rawValue = rand.Next(10, (int)canvasHeight);
                values.Add(rawValue);

                double scaledHeight = rawValue;
                var rect = new Rectangle
                {
```

```
                    Width = barWidth,
                    Height = scaledHeight,
                    Fill = Brushes.White
                };

                double x = i * (barWidth + spacing);
                double y = canvasHeight - scaledHeight;

                Canvas.SetLeft(rect, x);
                Canvas.SetTop(rect, y);
                bars.Add(rect);
                BarCanvas.Children.Add(rect);
            }
        }

        private void Reset_Click(object sender, RoutedEventArgs e) =>
GenerateBars();

        private async void SortBars_Click(object sender, RoutedEventArgs e) => await
MergeSort(0, values.Count - 1);

        private async Task MergeSort(int left, int right)
        {
            if (left >= right) return;

            int mid = (left + right) / 2;
            await MergeSort(left, mid);
            await MergeSort(mid + 1, right);
```

```
    await Merge(left, mid, right);
}

private async Task Merge(int left, int mid, int right)
{
    var temp = new List<int>();
    int i = left, j = mid + 1;

    while (i <= mid && j <= right)
    {
        Highlight(i, j);
        await Task.Delay(30);

        if (values[i] <= values[j])
        {
            temp.Add(values[i++]);
        }
        else
        {
            temp.Add(values[j++]);
        }
    }

    while (i <= mid) temp.Add(values[i++]);
    while (j <= right) temp.Add(values[j++]);

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

```
    {
        values[left + k] = temp[k];
        UpdateBar(left + k, temp[k]);
        await Task.Delay(20);
    }

    UnhighlightAll();
}

private void Highlight(int i, int j)
{
    bars[i].Fill = Brushes.DarkOrange;
    bars[j].Fill = Brushes.DarkOrange;
}

private void UnhighlightAll()
{
    foreach (var bar in bars)
        bar.Fill = Brushes.White;
}

private void UpdateBar(int index, int height)
{
    double canvasHeight = BarCanvas.ActualHeight;
    bars[index].Height = height;
    Canvas.SetTop(bars[index], canvasHeight - height);
}
```

```
        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;


        private void Close_Click(object sender, RoutedEventArgs e) => Close();


        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                DragMove();
        }


        public string language = "English";
        public bool infoOK = false;
        main.misc.Info info;
        string filePath;


        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\sortEN.txt");
                else if (language == "Romanian")
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\sortRO.txt");
```

```
            info.SetContentFromFile(filePath);
            info.Show();
            infoOK = true;
        }
        else
        {
            info.Close();
            infoOK = false;
        }
    }

    public int curlang = 0;

    private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
    {
        if (curlang == 0)
        {
            ChangeLanguage("Romanian");
            language = "Romanian";
            curlang = 1;
        }
        else
        {
            ChangeLanguage("English");
            language = "English";
            curlang = 0;
        }
```

```
        }


        private void ChangeLanguage(string language)

        {

            var resourceDictionary = new ResourceDictionary();

            if (language == "English")

                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);

            else if (language == "Romanian")

                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);


            Application.Current.Resources.MergedDictionaries.Clear();


Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

            this.Dispatcher.Invoke(() => this.InvalidateVisual());

        }

    }

}
```

9. Sortare topologică

```
<Window x:Class="proiectoicd2025.main.algoritmi.topological"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        Title="Topological Sort" Height="600" Width="800"

        Background="#1E1E1E"

        WindowStartupLocation="CenterScreen"

        WindowStyle="None"

        AllowsTransparency="True"

        ResizeMode="NoResize">
```

```
    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
```

```
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,481" Grid.RowSpan="2">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="{DynamicResource titluTopo}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>
            </Grid>
        </Border>
```

```
        <Canvas Name="GraphCanvas" Grid.Row="1" Background="Black"
ClipToBounds="True" Margin="10"/>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10,0,10">

            <Button Content="{DynamicResource addDjik}" Click="AddNode_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource connectDjik}"
Click="ConnectMode_Click" Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="Sort" Click="Sort_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Controls;

using System.IO;
```

```
using Path = System.IO.Path;


namespace proiectoicd2025.main.algoritmi
{
    public partial class topological : Window
    {
        private Dictionary<int, Point> nodePositions = new();
        private Dictionary<int, Ellipse> nodeVisuals = new();
        private Dictionary<int, List<int>> graph = new();
        private Dictionary<(int, int), Line> edgeVisuals = new();


        private int nodeCounter = 0;
        private int? selectedNode = null;
        private bool connectMode = false;


        public topological()
        {
            InitializeComponent();
            GraphCanvas.MouseLeftButtonDown += GraphCanvas_MouseLeftButtonDown;
        }


        private void GraphCanvas_MouseLeftButtonDown(object sender,
MouseButtonEventArgs e)
        {
            if (!connectMode)
            {
                var pos = e.GetPosition(GraphCanvas);
                AddNode(pos);
```

```
    }
}

private void AddNode(Point position)
{
    int id = nodeCounter++;
    nodePositions[id] = position;
    graph[id] = new();

    var ellipse = new Ellipse
    {
        Width = 30,
        Height = 30,
        Fill = Brushes.White,
        Stroke = Brushes.Red,
        StrokeThickness = 2,
        Tag = id
    };

    var label = new TextBlock
    {
        Text = id.ToString(),
        Foreground = Brushes.Red,
        FontWeight = FontWeights.Bold,
        FontSize = 14,
        Tag = id
    };
```

```
        Canvas.SetLeft(ellipse, position.X - 15);
        Canvas.SetTop(ellipse, position.Y - 15);
        Canvas.SetLeft(label, position.X - 6);
        Canvas.SetTop(label, position.Y - 10);

        ellipse.MouseLeftButtonDown += Node_Click;

        nodeVisuals[id] = ellipse;
        GraphCanvas.Children.Add(ellipse);
        GraphCanvas.Children.Add(label);
    }

    private void Node_Click(object sender, MouseButtonEventArgs e)
    {
        e.Handled = true;
        int node = (int)((Ellipse)sender).Tag;

        if (connectMode)
        {
            if (selectedNode == null)
            {
                selectedNode = node;
                nodeVisuals[node].Fill = Brushes.Orange;
            }
            else
            {
```

```
            if (selectedNode != node)
            {
                graph[selectedNode.Value].Add(node);
                DrawArrow(selectedNode.Value, node);
            }

            nodeVisuals[selectedNode.Value].Fill = Brushes.White;
            selectedNode = null;
        }
    }
}

private void DrawArrow(int from, int to)
{
    var start = nodePositions[from];
    var end = nodePositions[to];

    Vector direction = end - start;
    direction.Normalize();
    Point midpoint = new Point(
        (start.X + end.X) / 2,
        (start.Y + end.Y) / 2
    );

    var line = new Line
    {
        X1 = start.X,
```

```
    Y1 = start.Y,
    X2 = end.X,
    Y2 = end.Y,
    Stroke = Brushes.White,
    StrokeThickness = 2
};
GraphCanvas.Children.Add(line);

double angle = Math.Atan2(end.Y - start.Y, end.X - start.X);
double arrowLength = 14;
double arrowAngle = Math.PI / 6;

Point p1 = new Point(
    midpoint.X - arrowLength * Math.Cos(angle - arrowAngle),
    midpoint.Y - arrowLength * Math.Sin(angle - arrowAngle));
Point p2 = new Point(
    midpoint.X - arrowLength * Math.Cos(angle + arrowAngle),
    midpoint.Y - arrowLength * Math.Sin(angle + arrowAngle));

var arrowHead = new Polygon
{
    Points = new PointCollection { midpoint, p1, p2 },
    Fill = Brushes.SaddleBrown
};
GraphCanvas.Children.Add(arrowHead);

edgeVisuals[(from, to)] = line;
```

```
        }



        private async void Sort_Click(object sender, RoutedEventArgs e)
        {
            var inDegree = graph.Keys.ToDictionary(k => k, k => 0);

            foreach (var kv in graph)
                foreach (var neighbor in kv.Value)
                    inDegree[neighbor]++;

            var queue = new Queue<int>(inDegree.Where(kv => kv.Value == 0).Select(kv
=> kv.Key));
            var result = new List<int>();

            while (queue.Count > 0)
            {
                int node = queue.Dequeue();
                result.Add(node);

                nodeVisuals[node].Fill = Brushes.Lime;
                await Task.Delay(300);

                foreach (var neighbor in graph[node])
                {
                    inDegree[neighbor]--;
                    if (inDegree[neighbor] == 0)
                        queue.Enqueue(neighbor);
```

```
                }
            }

            if (result.Count != graph.Count)
            {
                MessageBox.Show("Cycle detected! Topological sort not possible.",
"Error", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }
        }

        private void ConnectMode_Click(object sender, RoutedEventArgs e)
        {
            connectMode = !connectMode;
            selectedNode = null;
        }

        private void Reset_Click(object sender, RoutedEventArgs e)
        {
            GraphCanvas.Children.Clear();
            nodeCounter = 0;
            selectedNode = null;
            nodePositions.Clear();
            nodeVisuals.Clear();
            edgeVisuals.Clear();
            graph.Clear();
        }
```

```
private void AddNode_Click(object sender, RoutedEventArgs e)
{
    connectMode = false;
}

private void Minimize_Click(object sender, RoutedEventArgs e)
{
    WindowState = WindowState.Minimized;
}

private void Close_Click(object sender, RoutedEventArgs e)
{
    Close();
}

private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        DragMove();
}
public string language = "English";
public bool infoOK = false;
main.misc.Info info;
string filePath;
private void info_show(object sender, RoutedEventArgs e)
{
    if (!infoOK)
```

```
            {
                info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\topologicalEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\algoritmi\topologicalRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }

        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
```

```
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }
```

```
            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

            this.Dispatcher.Invoke(() => this.InvalidateVisual());

        }

    }

}
```

SECȚIUNEA 3.3 Jocuri

1. Labirint 3D

```
<Window x:Class="proiectoicd2025.main.jocuri.raytest"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        Title="Raycasting with Mirrors" Width="1000" Height="500"

        ResizeMode="CanResizeWithGrip"

        WindowStartupLocation="CenterScreen"

        Background="Transparent"

        WindowStyle="None"

        AllowsTransparency="True">


    <Window.Resources>

        <Style TargetType="Button" x:Key="mouseOverStyle">

            <Style.Triggers>

                <Trigger Property="IsMouseOver" Value="True">

                    <Setter Property="Foreground" Value="Black"/>

                </Trigger>

            </Style.Triggers>

        </Style>
```

```
    </Window.Resources>


    <Grid>
        <Border Height="40" Background="#FF1E1E1E" VerticalAlignment="Top">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="" Foreground="White" VerticalAlignment="Center"
Margin="10,0,0,0" FontWeight="Bold" Height="40"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Minimize_Click"/>
                <Button Grid.Column="3" Content="X" Width="40" Height="30"
Background="Transparent" Foreground="White" BorderThickness="0"
Click="Close_Click"/>
            </Grid>
        </Border>

        <Rectangle Fill="Black" Margin="0,40,0,0"/>

        <Grid Margin="0,40,0,0">
            <Canvas Name="DrawingCanvas" Background="Black"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
```

```
            <Canvas Name="MiniMapCanvas" Background="White" Width="200" Height="200"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10">

                <Canvas.RenderTransform>

                    <ScaleTransform x:Name="MiniMapScaleTransform" ScaleX="1"
ScaleY="1"/>

                </Canvas.RenderTransform>

            </Canvas>



            <StackPanel HorizontalAlignment="Right" VerticalAlignment="Top"
Margin="10" Width="250">

                <TextBlock Text="{DynamicResource comenziRay}"
TextAlignment="Center" FontSize="18" FontWeight="Bold" Foreground="Red"
Margin="0,0,0,10" HorizontalAlignment="Center"/>

                <TextBlock Text="{DynamicResource startRay}" Foreground="Blue"
TextAlignment="Center" FontSize="14" Margin="0,5"/>

                <TextBlock Text="{DynamicResource miscareRay}" Foreground="Blue"
TextAlignment="Center" FontSize="14" Margin="0,5"/>

                <TextBlock Text="{DynamicResource regenRay}" Foreground="Blue"
TextAlignment="Center" FontSize="14" Margin="0,5"/>

                <TextBlock Text="I - Info" Foreground="Blue" TextAlignment="Center"
FontSize="14" Margin="0,5"/>

                <TextBlock Text="L - ENG/ROM" Foreground="Blue"
TextAlignment="Center" FontSize="14" Margin="0,5"/>

            </StackPanel>

        </Grid>

    </Grid>

</Window>

--
using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;
```

```
using System.Windows;
using proiectoicd2025.main.misc;
using System.IO;
using Path = System.IO.Path;

namespace proiectoicd2025.main.jocuri
{
    public partial class raytest : Window
    {
        const int L = 8;
        const int H = 8;
        const int S = 25;
        const double F = Math.PI / 3;

        string limba = "English";

        int[,] m = new int[L, H]
        {
            {1,1,1,1,1,1,1,1},
            {1,0,0,0,0,0,0,1},
            {1,0,0,0,1,0,0,1},
            {1,0,0,0,1,0,0,1},
            {1,0,1,0,1,1,1,1},
            {1,0,1,0,1,0,0,1},
            {1,0,0,0,0,0,0,1},
            {1,1,1,1,1,1,1,1}
        };
```

```
double px = 3.5, py = 3.5, pa = 0;
const double MS = 0.02;
const double TS = Math.PI / 60;

Random r = new Random();
Point tp;
bool[,] v = new bool[L, H];
List<Point> cPath = new List<Point>();
int idx = 0;
bool autoExplore = false;

public raytest()
{
    InitializeComponent();
    CompositionTarget.Rendering += OnRender;
    this.KeyDown += OnKeyDown;
    SetTreasure();
}

private void SetTreasure()
{
    int x, y;
    do
    {
        x = r.Next(1, L - 1);
        y = r.Next(1, H - 1);
```

```
    } while (m[x, y] != 0);
    tp = new Point(x, y);
}

private void OnRender(object sender, EventArgs e)
{
    if (autoExplore)
        UpdateAutoExplore();

    RenderScene();
}

private void RenderScene()
{
    DrawingCanvas.Children.Clear();
    MiniMapCanvas.Children.Clear();

    int w = (int)DrawingCanvas.ActualWidth;
    int h = (int)DrawingCanvas.ActualHeight;
    if (w == 0 || h == 0) return;

    for (int col = 0; col < w; col++)
    {
        double rayA = pa - F / 2 + F * col / w;
        RayHitInfo hit = CastRay(px, py, rayA);
        int wallHeight = (int)(h / (hit.D + 0.0001));
        wallHeight = Math.Min(wallHeight, h);
```

```
        Rectangle slice = new Rectangle
        {
            Width = 1,
            Height = wallHeight,
            Fill = Brushes.Gray
        };
        Canvas.SetLeft(slice, col);
        Canvas.SetTop(slice, (h - wallHeight) / 2);
        DrawingCanvas.Children.Add(slice);
    }

    DrawMiniMap();
}

private void DrawMiniMap()
{
    for (int y = 0; y < H; y++)
    {
        for (int x = 0; x < L; x++)
        {
            Rectangle tile = new Rectangle
            {
                Width = S - 1,
                Height = S - 1,
                Fill = m[x, y] == 1 ? Brushes.DarkGray :
                       v[x, y] ? Brushes.LightYellow : Brushes.White
```

```
        };
        Canvas.SetLeft(tile, x * S);
        Canvas.SetTop(tile, y * S);
        MiniMapCanvas.Children.Add(tile);
    }
}

Rectangle treasure = new Rectangle
{
    Width = S - 5,
    Height = S - 5,
    Fill = Brushes.Gold
};
Canvas.SetLeft(treasure, tp.X * S + 2);
Canvas.SetTop(treasure, tp.Y * S + 2);
MiniMapCanvas.Children.Add(treasure);

if (cPath != null)
{
    foreach (var pt in cPath)
    {
        Rectangle step = new Rectangle
        {
            Width = S - 10,
            Height = S - 10,
            Fill = Brushes.LightGreen
        };
```

```
            Canvas.SetLeft(step, pt.X * S + 5);
            Canvas.SetTop(step, pt.Y * S + 5);
            MiniMapCanvas.Children.Add(step);
        }
    }

    Ellipse player = new Ellipse
    {
        Width = 6,
        Height = 6,
        Fill = Brushes.Blue
    };
    Canvas.SetLeft(player, px * S - 3);
    Canvas.SetTop(player, py * S - 3);
    MiniMapCanvas.Children.Add(player);

    Line dirLine = new Line
    {
        X1 = px * S,
        Y1 = py * S,
        X2 = px * S + Math.Cos(pa) * 20,
        Y2 = py * S + Math.Sin(pa) * 20,
        Stroke = Brushes.Red,
        StrokeThickness = 2
    };
    MiniMapCanvas.Children.Add(dirLine);
}
```

```
private void UpdateAutoExplore()
{
    int pxInt = (int)px;
    int pyInt = (int)py;

    if (!v[pxInt, pyInt])
        v[pxInt, pyInt] = true;

    if ((int)tp.X == pxInt && (int)tp.Y == pyInt)
    {
        autoExplore = false;
        return;
    }

    if (cPath == null || idx >= cPath.Count)
    {
        Point? next = FindNearestUnvisited(new Point(pxInt, pyInt));
        if (next != null)
        {
            cPath = FindPath(new Point(pxInt, pyInt), next.Value);
            idx = 0;
        }
        else
        {
            autoExplore = false;
            return;
```

```
    }
}

if (cPath == null || idx >= cPath.Count)
    return;

Point target = cPath[idx];
double dx = target.X + 0.5 - px;
double dy = target.Y + 0.5 - py;
double distance = Math.Sqrt(dx * dx + dy * dy);
double targetAngle = Math.Atan2(dy, dx);
double angleDiff = NormalizeAngle(targetAngle - pa);

if (Math.Abs(angleDiff) > 0.05)
{
    pa += Math.Sign(angleDiff) * TS;
    return;
}

if (distance > 0.05)
{
    px += Math.Cos(pa) * MS;
    py += Math.Sin(pa) * MS;
}
else
{
    idx++;
```

```
            }
        }

        private Point? FindNearestUnvisited(Point start)
        {
            Queue<Point> q = new Queue<Point>();
            HashSet<(int, int)> visitedBFS = new HashSet<(int, int)>();
            q.Enqueue(start);
            visitedBFS.Add(((int)start.X, (int)start.Y));

            while (q.Count > 0)
            {
                Point cur = q.Dequeue();
                int x = (int)cur.X, y = (int)cur.Y;

                if (m[x, y] == 0 && !v[x, y])
                    return cur;

                foreach (var (dx, dy) in new[] { (0, 1), (1, 0), (0, -1), (-1, 0) })
                {
                    int nx = x + dx, ny = y + dy;
                    if (nx >= 0 && ny >= 0 && nx < L && ny < H && !
visitedBFS.Contains((nx, ny)) && m[nx, ny] == 0)
                    {
                        visitedBFS.Add((nx, ny));
                        q.Enqueue(new Point(nx, ny));
                    }
                }
```

```
    }


    return null;
}
private bool openInfo = false;
main.misc.Info info;
string filePath;
private void OnKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.T)
    {
        autoExplore = !autoExplore;
        cPath = null;
        idx = 0;
    }


    if (e.Key == Key.L)
    {
        ChangeLanguage(curLang == 0 ? "Romanian" : "English");
    }


    if (e.Key == Key.R)
    {
        SetTreasure();
        v = new bool[L, H];
        cPath = null;
        idx = 0;
```

```
            }


            if (e.Key == Key.I)
            {
                if (!openInfo)
                {
                    info = new main.misc.Info();
                    filePath = string.Empty;
                    if (limba == "English")
                    {
                        filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\rayEN.txt");
                    }
                    else if (limba == "Romanian")
                    {
                        filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\rayRO.txt");
                    }
                    //MessageBox.Show((string)filePath);
                    info.SetContentFromFile(filePath);
                    info.Show();
                    openInfo = true;
                }
                else
                {
                    openInfo = false;
                    info.Close();
                }
```

```
        if (!autoExplore)
        {
            if (e.Key == Key.Up)
            {
                double nx = px + Math.Cos(pa) * MS;
                double ny = py + Math.Sin(pa) * MS;
                if (m[(int)nx, (int)ny] == 0)
                {
                    px = nx;
                    py = ny;
                }
            }
            else if (e.Key == Key.Left)
            {
                pa -= TS;
            }
            else if (e.Key == Key.Right)
            {
                pa += TS;
            }
        }
    }
}

private double NormalizeAngle(double a)
{
```

```
    while (a < -Math.PI) a += 2 * Math.PI;
    while (a > Math.PI) a -= 2 * Math.PI;
    return a;
}

public struct RayHitInfo { public double D; }

public RayHitInfo CastRay(double x, double y, double angle)
{
    double sin = Math.Sin(angle), cos = Math.Cos(angle);
    double rayX = x, rayY = y;

    while (true)
    {
        rayX += cos * 0.01;
        rayY += sin * 0.01;
        int mapX = (int)rayX, mapY = (int)rayY;

        if (mapX < 0 || mapX >= L || mapY < 0 || mapY >= H)
            return new RayHitInfo { D = 1000 };

        if (m[mapX, mapY] == 1)
            return new RayHitInfo { D = Distance(x, y, rayX, rayY) };
    }
}

private double Distance(double x1, double y1, double x2, double y2)
```

```
        {
            return Math.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
        }

        class Node
        {
            public int X, Y;
            public Node Parent;
            public double G, H;
            public double F => G + H;
        }

        private List<Point> FindPath(Point start, Point goal)
        {
            var open = new List<Node>();
            var closed = new HashSet<(int, int)>();
            var startNode = new Node { X = (int)start.X, Y = (int)start.Y, G = 0, H
= Heuristic(start, goal) };
            open.Add(startNode);

            while (open.Count > 0)
            {
                var current = open.OrderBy(n => n.F).First();
                if (current.X == (int)goal.X && current.Y == (int)goal.Y)
                    return ReconstructPath(current);

                open.Remove(current);
                closed.Add((current.X, current.Y));
```

```
                foreach (var neighbor in GetNeighbors(current))
                {
                    if (closed.Contains((neighbor.X, neighbor.Y)) || m[neighbor.X,
neighbor.Y] == 1)
                        continue;

                    double tentativeG = current.G + 1;
                    var existing = open.FirstOrDefault(n => n.X == neighbor.X && n.Y
== neighbor.Y);

                    if (existing == null)
                    {
                        neighbor.G = tentativeG;
                        neighbor.H = Heuristic(new Point(neighbor.X, neighbor.Y),
goal);
                        neighbor.Parent = current;
                        open.Add(neighbor);
                    }
                    else if (tentativeG < existing.G)
                    {
                        existing.G = tentativeG;
                        existing.Parent = current;
                    }
                }
            }

            return null;
        }
```

```
private double Heuristic(Point a, Point b)
{
    return Math.Abs(a.X - b.X) + Math.Abs(a.Y - b.Y);
}

private List<Node> GetNeighbors(Node node)
{
    var dirs = new[] { (0, 1), (1, 0), (0, -1), (-1, 0) };
    var neighbors = new List<Node>();

    foreach (var (dx, dy) in dirs)
    {
        int nx = node.X + dx, ny = node.Y + dy;
        if (nx >= 0 && ny >= 0 && nx < L && ny < H)
            neighbors.Add(new Node { X = nx, Y = ny });
    }

    return neighbors;
}

private List<Point> ReconstructPath(Node node)
{
    var path = new List<Point>();
    while (node != null)
    {
        path.Add(new Point(node.X, node.Y));
```

```
        node = node.Parent;
    }
    path.Reverse();
    return path;
}

int curLang = 0;
private void ChangeLanguage(string language)
{
    var rd = new ResourceDictionary();
    if (language == "English")
    {
        language = "English";
        curLang = 0;
        limba = "English";
        rd.Source = new Uri("limbi/eng.xaml", UriKind.Relative);
    }
    else if (language == "Romanian")
    {
        language = "Romanian";
        curLang = 1;
        limba = "Romanian";
        rd.Source = new Uri("limbi/ro.xaml", UriKind.Relative);
    }

    Application.Current.Resources.MergedDictionaries.Clear();
    Application.Current.Resources.MergedDictionaries.Add(rd);
```

```
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            WindowState = WindowState.Minimized;
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

    }
}
```

2. Regine

```
<Window x:Class="proiectoicd2025.main.jocuri.regine"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="N-Queens" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
```

```
        AllowsTransparency="True"
        ResizeMode="CanResizeWithGrip">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
```

```
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="N-Queens" Foreground="White" FontWeight="Bold"
FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" />
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" />
```

```
            </Grid>
        </Border>

        <UniformGrid Grid.Row="1" Name="ChessBoard" Rows="8" Columns="8"
Margin="0,10"/>

        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center">
            <Button Content="Reset" Click="Reset_Click" Style="{StaticResource
ThemedButtonStyle}"/>
            <Button Content="{DynamicResource CheckSUDOKU}" Click="Check_Click"
Style="{StaticResource ThemedButtonStyle}"/>
            <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}"/>
            <Button Content="Info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}"/>
            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}"/>
        </StackPanel>
    </Grid>
</Window>
--
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace proiectoicd2025.main.jocuri
{
    public partial class regine : Window
```

```
    {
        private int N = 8;
        private Button[,] board;

        public regine()
        {
            InitializeComponent();
            SetupBoard();
        }

        private void SetupBoard()
        {
            ChessBoard.Children.Clear();
            board = new Button[N, N];

            for (int row = 0; row < N; row++)
            {
                for (int col = 0; col < N; col++)
                {
                    var cell = new Button
                    {
                        Background = (row + col) % 2 == 0 ? Brushes.White :
Brushes.Gray,
                        Tag = new Point(row, col),
                        FontSize = 24,
                    };
                    cell.Click += Cell_Click;
                    board[row, col] = cell;
```

```
            ChessBoard.Children.Add(cell);
        }
    }
}

private void Cell_Click(object sender, RoutedEventArgs e)
{
    var btn = sender as Button;
    var pos = (Point)btn.Tag;
    int row = (int)pos.X, col = (int)pos.Y;

    if ((string)btn.Content == "♛")
        btn.Content = null;
    else
        btn.Content = "♛";
}

private void Reset_Click(object sender, RoutedEventArgs e)
{
    SetupBoard();
}

private void Check_Click(object sender, RoutedEventArgs e)
{
    int[] queens = new int[N];
    for (int row = 0; row < N; row++)
    {
```

```
    bool found = false;
    for (int col = 0; col < N; col++)
    {
        if ((string)board[row, col].Content == "♛")
        {
            queens[row] = col;
            found = true;
            break;
        }
    }
    if (!found)
    {
        MessageBox.Show("Not all queens are placed.");
        return;
    }
}

for (int i = 0; i < N; i++)
{
    for (int j = i + 1; j < N; j++)
    {
        if (queens[i] == queens[j] ||
            queens[i] - i == queens[j] - j ||
            queens[i] + i == queens[j] + j)
        {
            MessageBox.Show("Queens are attacking each other!");
            return;
```

```
                }
            }
        }

        MessageBox.Show("Well done! All queens are safe.");
    }

    private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
    {
        if (e.ChangedButton == MouseButton.Left)
            this.DragMove();
    }

    private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;

    private void Close_Click(object sender, RoutedEventArgs e) => Close();

    private void Solve_Click(object sender, RoutedEventArgs e)
    {
        int[] solution = new int[N];
        if (PlaceQueens(solution, 0))
        {
            for (int row = 0; row < N; row++)
            {
                for (int col = 0; col < N; col++)
                {
                    board[row, col].Content = null;
```

```
            }
        }

        for (int row = 0; row < N; row++)
        {
            board[row, solution[row]].Content = "♕";
        }
    }
    else
    {
        MessageBox.Show("No solution found.");
    }
}

private bool PlaceQueens(int[] queens, int row)
{
    if (row == N)
        return true;

    for (int col = 0; col < N; col++)
    {
        bool safe = true;
        for (int prev = 0; prev < row; prev++)
        {
            if (queens[prev] == col ||
                queens[prev] - prev == col - row ||
                queens[prev] + prev == col + row)
```

```
            {
                safe = false;
                break;
            }
        }

        if (safe)
        {
            queens[row] = col;
            if (PlaceQueens(queens, row + 1))
                return true;
        }
    }

    return false;
}
public bool infoOK = false;
main.misc.Info info;
string filePath;
string language = "English";
private void info_show(object sender, RoutedEventArgs e)
{
    if (!infoOK)
    {
        info = new main.misc.Info();
        filePath = string.Empty;
        if (language == "English")
```

```
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\regineEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\regineRO.txt");
                }
                info.SetContentFromFile(filePath);
                info.Show();
                infoOK = true;
            }
            else
            {
                info.Close();
                infoOK = false;
            }
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
```

```
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
```

```
    }
}
```

3. Sudoku

```
<Window x:Class="proiectoicd2025.main.jocuri.sudoku"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Sudoku" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="CanResizeWithGrip">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
```

```
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>

        <Style x:Key="ThemedComboBoxStyle" TargetType="ComboBox">
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="5,2"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="FocusVisualStyle" Value="{x:Null}"/>
            <Setter Property="Template">
                <Setter.Value>
```

```
                    <ControlTemplate TargetType="ComboBox">
                        <Border Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}"
                        CornerRadius="2">
                            <Grid>
                                <ToggleButton Name="ToggleButton"
                                      Background="Transparent"
                                      BorderThickness="0"
                                      Focusable="False"
                                      IsChecked="{Binding Path=IsDropDownOpen,
RelativeSource={RelativeSource TemplatedParent}, Mode=TwoWay}"
                                      ClickMode="Press" />

                                <ContentPresenter Name="ContentSite"
                                          IsHitTestVisible="False"
                                          Content="{TemplateBinding
SelectionBoxItem}"
                                          ContentTemplate="{TemplateBinding
SelectionBoxItemTemplate}"
                                          ContentTemplateSelector="{TemplateBinding
ItemTemplateSelector}"
                                          Margin="10,0"
                                          VerticalAlignment="Center"
                                          HorizontalAlignment="Left"/>

                                 <Popup Name="Popup"
                                Placement="Bottom"
                                IsOpen="{TemplateBinding IsDropDownOpen}"
```

```
                                AllowsTransparency="True"
                                Focusable="False"
                                PopupAnimation="Slide">
                                    <Border Background="#1E1E1E"
                                    BorderBrush="Red"
                                    BorderThickness="2"
                                    CornerRadius="8"
                                    SnapsToDevicePixels="True"
                                    Margin="0,2,0,0">
                                        <ScrollViewer Margin="4"
SnapsToDevicePixels="True">
                                            <StackPanel IsItemsHost="True"
KeyboardNavigation.DirectionalNavigation="Contained"/>
                                        </ScrollViewer>
                                    </Border>
                                </Popup>
                            </Grid>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>


        <ControlTemplate x:Key="ComboBoxToggleButton" TargetType="ToggleButton">
            <Grid>
                <Path x:Name="Arrow"
```

```
                    Data="M 0 0 L 4 4 L 8 0 Z"
                    Fill="Red"
                    HorizontalAlignment="Right"
                    Margin="0,0,10,0"
                    VerticalAlignment="Center"
                    Width="10"
                    Height="6"/>
            </Grid>
        </ControlTemplate>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="Sudoku" Foreground="White" FontWeight="Bold"
```

```
FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="–" Width="40" Height="30"
Click="Minimize_Click"

                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,0,5"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click"

                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,5,5"/>

            </Grid>

        </Border>


        <Border Grid.Row="1" BorderBrush="Red" BorderThickness="3" CornerRadius="10"
Padding="0,10,0,0" Margin="0,10,0,15">

            <UniformGrid Name="SudokuGrid" Rows="9" Columns="9"/>

        </Border>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Bottom">

            <ComboBox x:Name="DifficultySelector"

                      Width="100"

                      Style="{StaticResource ThemedComboBoxStyle}"

                      SelectedIndex="0"

                      SelectionChanged="Difficulty_Changed">

                <ComboBoxItem Content="{DynamicResource EasySUDOKU}"/>

                <ComboBoxItem Content="{DynamicResource MediumSUDOKU}"/>

                <ComboBoxItem Content="Hard"/>

            </ComboBox>


            <Button Content="Restart" Click="Restart_Click" Style="{StaticResource
```

```
ThemedButtonStyle}"/>

            <Button Content="{DynamicResource CheckSUDOKU}" Click="Check_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="{DynamicResource SolveSUDOKU}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}"/>

            <Button Content="ENG/ROM" Click="Translate_Click" Style="{StaticResource
ThemedButtonStyle}"/>


        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;


namespace proiectoicd2025.main.jocuri

{

    public partial class sudoku : Window

    {

        private readonly string[] easyPuzzles = new string[] {

"530070000600195000098000060800060003400803001700020006060000280000419005000080079"

        };


        private readonly string[] mediumPuzzles = new string[] {
```

```
"100489006000003100040070200007001800300000004001700500005010020009800000400256001"
        };

        private readonly string[] hardPuzzles = new string[] {
"000000907000420180000705026100904000050000400005070099201080000340590005070000000"
        };

        private TextBox[,] cells = new TextBox[9, 9];
        private int[,] board = new int[9, 9];
        private bool[,] isInitial = new bool[9, 9];
        private string selectedDifficultyIndex = "easy";


        public sudoku()
        {
            InitializeComponent();
            GenerateSudokuGrid();
            LoadRandomPuzzle();
        }

        private void GenerateSudokuGrid()
        {
            SudokuGrid.Children.Clear();
            for (int row = 0; row < 9; row++)
            {
                for (int col = 0; col < 9; col++)
```

```
                {
                    var box = new TextBox
                    {
                        FontSize = 20,
                        FontWeight = FontWeights.Bold,
                        HorizontalContentAlignment = HorizontalAlignment.Center,
                        VerticalContentAlignment = VerticalAlignment.Center,
                        BorderBrush = Brushes.White,
                        BorderThickness = new Thickness(0.5),
                        MaxLength = 1,
                        Background = ((row / 3 + col / 3) % 2 == 0) ?
Brushes.DimGray : Brushes.Black,
                        Foreground = Brushes.White,
                        Tag = (row, col),
                        CaretBrush = Brushes.Red
                    };

                    cells[row, col] = box;
                    SudokuGrid.Children.Add(box);
                }
            }
        }
        private void LoadRandomPuzzle()
        {
            Random random = new Random();
            string puzzle;

            switch (selectedDifficultyIndex)
```

```
    {
        case "0":
            puzzle = easyPuzzles[random.Next(easyPuzzles.Length)];
            break;
        case "1":
            puzzle = mediumPuzzles[random.Next(mediumPuzzles.Length)];
            break;
        case "2":
            puzzle = hardPuzzles[random.Next(hardPuzzles.Length)];
            break;
        default:
            puzzle = easyPuzzles[0];
            break;
    }

    LoadPuzzle(puzzle);
}



private void LoadPuzzle(string puzzle)
{
    for (int row = 0; row < 9; row++)
    {
        for (int col = 0; col < 9; col++)
        {
            int value = int.Parse(puzzle[row * 9 + col].ToString());
```

```
                    board[row, col] = value;
                    isInitial[row, col] = value != 0;
                    cells[row, col].Text = value == 0 ? "" : value.ToString();
                    cells[row, col].IsReadOnly = value != 0;
                    cells[row, col].Background = ((row / 3 + col / 3) % 2 == 0) ?
Brushes.DimGray : Brushes.Black;
                }
            }
        }

        private void UpdateBoardFromUI()
        {
            for (int row = 0; row < 9; row++)
                for (int col = 0; col < 9; col++)
                    board[row, col] = int.TryParse(cells[row, col].Text, out int
val) ? val : 0;
        }

        private void UpdateUIFromBoard()
        {
            for (int row = 0; row < 9; row++)
            {
                for (int col = 0; col < 9; col++)
                {
                    cells[row, col].Text = board[row, col] == 0 ? "" : board[row,
col].ToString();
                }
            }
        }
```

```
        private void Check_Click(object sender, RoutedEventArgs e)
        {
            UpdateBoardFromUI();
            if (IsValidSudoku(board))
            {
                MessageBox.Show((string)
Application.Current.Resources["LooksGoodSoFarSUDOKU"],
                                (string)
Application.Current.Resources["CheckSUDOKU"],
                                MessageBoxButton.OK, MessageBoxImage.Information);
            }
            else
            {
                MessageBox.Show((string)
Application.Current.Resources["TheresAMistakeSUDOKU"],
                                (string)
Application.Current.Resources["CheckSUDOKU"],
                                MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }

        private async void Solve_Click(object sender, RoutedEventArgs e)
        {
            UpdateBoardFromUI();
            bool solved = await VisualSolve(board);
            if (!solved)
            {
                MessageBox.Show((string)
```

```
Application.Current.Resources["ThisPuzzleCannotBeSolvedSUDOKU"],
                                (string)
Application.Current.Resources["SolveSUDOKU"],
                                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }


        private async Task<bool> VisualSolve(int[,] grid)
        {
            for (int row = 0; row < 9; row++)
            {
                for (int col = 0; col < 9; col++)
                {
                    if (grid[row, col] == 0)
                    {
                        for (int num = 1; num <= 9; num++)
                        {
                            if (IsSafe(grid, row, col, num))
                            {
                                grid[row, col] = num;
                                cells[row, col].Text = num.ToString();
                                cells[row, col].Foreground = Brushes.DarkRed;
                                await Task.Delay(10);
                                if (await VisualSolve(grid))
                                {
                                    cells[row, col].Foreground = Brushes.White;
                                    return true;
                                }
```

```
                            grid[row, col] = 0;
                            cells[row, col].Text = "";
                            await Task.Delay(5);
                        }
                    }
                    return false;
                }
            }
        }
        return true;
    }

    private bool IsValidSudoku(int[,] grid)
    {
        for (int row = 0; row < 9; row++)
            for (int col = 0; col < 9; col++)
                if (grid[row, col] != 0 && !IsSafe(grid, row, col, grid[row,
col], true))
                    return false;
        return true;
    }

    private bool IsSafe(int[,] grid, int row, int col, int num, bool selfCheck =
false)
    {
        for (int i = 0; i < 9; i++)
        {
            if (grid[row, i] == num && (!selfCheck || i != col)) return false;
```

```
                if (grid[i, col] == num && (!selfCheck || i != row)) return false;
            }

            int boxRow = row / 3 * 3;
            int boxCol = col / 3 * 3;
            for (int i = boxRow; i < boxRow + 3; i++)
                for (int j = boxCol; j < boxCol + 3; j++)
                    if (grid[i, j] == num && (!selfCheck || i != row || j != col))
return false;

            return true;
        }

        private void Restart_Click(object sender, RoutedEventArgs e)
        {
            LoadRandomPuzzle();
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }
```

```
        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Translate_Click(object sender, RoutedEventArgs e)
        {
            if (curl == 0)
                ChangeLanguage("Romanian");
            else if (curl == 1)
                ChangeLanguage("English");
        }
        private int curl = 0;
        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                curl = 1;
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                curl = 0;
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
```

```
            }

            Application.Current.Resources.MergedDictionaries.Clear();
Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }

        private void Difficulty_Changed(object sender, SelectionChangedEventArgs e)
        {
            selectedDifficultyIndex = DifficultySelector.SelectedIndex.ToString();
        }

    }
}
```

SECȚIUNEA 3.4 Matematică

1. Graf colorat

```
<Window x:Class="proiectoicd2025.main.matematica.grafcolorat"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Subset Sum Problem" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize"
        Loaded="Window_Loaded">
```

```
    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
```

```
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,476" Grid.RowSpan="2">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="{DynamicResource mate1}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click"
                    Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click"
                    Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>
            </Grid>
```

```
        </Border>


        <Border Grid.Row="1" BorderBrush="Red" BorderThickness="3" CornerRadius="2"
Padding="0,10,0,0" Margin="0,10,0,0">

            <Canvas x:Name="GraphCanvas" Background="Transparent"
MouseLeftButtonDown="GraphCanvas_MouseLeftButtonDown" Margin="0,-10,0,0"/>

        </Border>


        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="10">

            <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}" Width="200" Height="29"/>

            <Button Content="ENG/ROM" Click="ChangeLanguage_Click"
Style="{StaticResource ThemedButtonStyle}" Width="200" Height="29"/>

            <Button Content="INFO" Click="info_show" Foreground="White"
Background="#2C2C2C" x:Name="Info" Margin="30,0,0,0" FontSize="15" Width="100"
Height="25"/>

        </StackPanel>

    </Grid>


</Window>

--
using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Shapes;
using Path = System.IO.Path;

namespace proiectoicd2025.main.matematica
{
    public partial class grafcolorat : Window
    {
        private class Nod
        {
            public int Id { get; set; }
            public Point Poz { get; set; }
        }

        private List<Nod> noduri = new List<Nod>();
        private List<Tuple<int, int>> muchii = new List<Tuple<int, int>>();
        private Dictionary<int, int> culori = new Dictionary<int, int>();
        private Nod nodSelectat = null;
        private int contorNoduri = 0;

        public grafcolorat()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
```

```
            GraphCanvas.Background = Brushes.Black;
            GraphCanvas.MouseLeftButtonDown += GraphCanvas_MouseLeftButtonDown;
            GraphCanvas.MouseRightButtonDown += GraphCanvas_MouseRightButtonDown;
        }

        private void GraphCanvas_MouseLeftButtonDown(object sender,
MouseButtonEventArgs e)
        {
            Point clickPos = e.GetPosition(GraphCanvas);

            Nod nodClickat = noduri.FirstOrDefault(nod =>
            {
                double dx = clickPos.X - nod.Poz.X;
                double dy = clickPos.Y - nod.Poz.Y;
                return Math.Sqrt(dx * dx + dy * dy) <= 20;
            });

            if (nodClickat != null)
            {
                VerNod(nodClickat);
            }
            else
            {
                var nodNou = new Nod
                {
                    Id = contorNoduri++,
                    Poz = clickPos
                };
```

```
            noduri.Add(nodNou);

            nodSelectat = null;

            desGraph();
        }
    }

    private void GraphCanvas_MouseRightButtonDown(object sender,
MouseButtonEventArgs e)
    {
        Point clickPos = e.GetPosition(GraphCanvas);

        foreach (var nod in noduri.ToList())
        {
            double dx = clickPos.X - nod.Poz.X;
            double dy = clickPos.Y - nod.Poz.Y;
            if (Math.Sqrt(dx * dx + dy * dy) <= 20)
            {
                StergeNod(nod);
                return;
            }
        }
    }

    private void VerNod(Nod nod)
    {
        if (nodSelectat == null)
```

```
            {
                nodSelectat = nod;
            }
            else if (nodSelectat != nod)
            {
                muchii.Add(Tuple.Create(nodSelectat.Id, nod.Id));
                nodSelectat = null;
                desGraph();
            }
        }

        private void StergeNod(Nod nod)
        {
            contorNoduri--;
            noduri.Remove(nod);
            muchii = muchii.Where(m => m.Item1 != nod.Id && m.Item2 !=
nod.Id).ToList();
            culori.Remove(nod.Id);
            if (nodSelectat == nod)
                nodSelectat = null;
            desGraph();
        }

        private void desGraph()
        {
            GraphCanvas.Children.Clear();

            foreach (var muchie in muchii)
```

```
{
    var nodA = noduri.First(n => n.Id == muchie.Item1);
    var nodB = noduri.First(n => n.Id == muchie.Item2);

    var linie = new Line
    {
        X1 = nodA.Poz.X,
        Y1 = nodA.Poz.Y,
        X2 = nodB.Poz.X,
        Y2 = nodB.Poz.Y,
        Stroke = Brushes.Gray,
        StrokeThickness = 2
    };
    GraphCanvas.Children.Add(linie);
}

foreach (var nod in noduri)
{
    var elipsa = new Ellipse
    {
        Width = 40,
        Height = 40,
        Fill = new SolidColorBrush(getculoare(nod.Id)),
        Stroke = nod == nodSelectat ? Brushes.Red : Brushes.White,
        StrokeThickness = 2
    };
    Canvas.SetLeft(elipsa, nod.Poz.X - 20);
```

```
            Canvas.SetTop(elipsa, nod.Poz.Y - 20);
            GraphCanvas.Children.Add(elipsa);

            var eticheta = new TextBlock
            {
                Text = nod.Id.ToString(),
                Foreground = Brushes.Black,
                FontSize = 14,
                FontWeight = FontWeights.Bold,
                Width = 40,
                TextAlignment = TextAlignment.Center
            };
            Canvas.SetLeft(eticheta, nod.Poz.X - 20);
            Canvas.SetTop(eticheta, nod.Poz.Y - 10);
            GraphCanvas.Children.Add(eticheta);
        }
    }

    private Color getculoare(int id)
    {
        if (culori.ContainsKey(id))
        {
            int culoareIdx = culori[id];
            return ColorFromIndex(culoareIdx);
        }
        return Colors.LightGray;
    }
```

```
        private Color ColorFromIndex(int index)
        {
            Color[] paleta = {
                    Colors.Red, Colors.Blue, Colors.Yellow
                };
            return paleta[index % paleta.Length];
        }

        private async void Solve_Click(object sender, RoutedEventArgs e)
        {
            culori.Clear();
            bool succes = await VisualColorGraph(0);
            if (!succes)
                MessageBox.Show("Nu am gasit o solutie de colorare.", "Graph
Coloring", MessageBoxButton.OK, MessageBoxImage.Warning);
            desGraph();
        }

        private async Task<bool> VisualColorGraph(int nodIndex)
        {
            if (nodIndex == noduri.Count) return true;

            int curentId = noduri[nodIndex].Id;
            for (int culoare = 0; culoare < 8; culoare++)
            {
                if (safe(curentId, culoare))
                {
```

```
                culori[curentId] = culoare;
                desGraph();
                await Task.Delay(200);

                if (await VisualColorGraph(nodIndex + 1)) return true;

                culori.Remove(curentId);
                desGraph();
                await Task.Delay(100);
            }
        }
        return false;
    }

    private bool safe(int nodId, int culoare)
    {
        foreach (var muchie in muchii)
        {
            if (muchie.Item1 == nodId && culori.TryGetValue(muchie.Item2, out
int culoareVecin) && culoareVecin == culoare)
                return false;
            if (muchie.Item2 == nodId && culori.TryGetValue(muchie.Item1, out
culoareVecin) && culoareVecin == culoare)
                return false;
        }
        return true;
    }
```

```
        private void Close_Click(object sender, RoutedEventArgs e) => Close();


        private void Minimize_Click(object sender, RoutedEventArgs e) => WindowState
= WindowState.Minimized;


        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.LeftButton == MouseButtonState.Pressed)
                DragMove();
        }


        public bool infoOK = false;
        private main.misc.Info _Info;
        string filePath;
        string language = "English";
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                _Info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\matematica\grafcoloratEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
```

```
@"resource\info\jocuri\grafcoloratRO.txt");
                }
                _Info.SetContentFromFile(filePath);
                _Info.Show();
                infoOK = true;
            }
            else
            {
                _Info.Close();
                infoOK = false;
            }
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
```

```
        }


        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }


            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

2. Magic Square

```
<Window x:Class="proiectoicd2025.main.matematica.magicsquare"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        Title="Magic Square Generator" Height="700" Width="700"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
```

```
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,49" Grid.RowSpan="2">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="{DynamicResource mate2}" Foreground="White"
```

```
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="–" Width="40" Height="30"
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>

            </Grid>

        </Border>


        <StackPanel Grid.Row="1" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10">

            <TextBlock Text="Rows:" Foreground="White" VerticalAlignment="Center"
Margin="5"/>

            <TextBox x:Name="RowInput" Width="50" Margin="5"/>

            <TextBlock Text="Columns:" Foreground="White" VerticalAlignment="Center"
Margin="5"/>

            <TextBox x:Name="ColumnInput" Width="50" Margin="5"/>

            <TextBlock Text="Target Sum:" Foreground="White"
VerticalAlignment="Center" Margin="5"/>

            <TextBox x:Name="SumInput" Width="70" Margin="5"/>

            <Button Content="Generate Grid" Click="GenerateGrid_Click"
Style="{StaticResource ThemedButtonStyle}"/>

        </StackPanel>


        <ScrollViewer Grid.Row="2" Margin="0,10"
HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">

            <UniformGrid x:Name="MagicGrid" Rows="3" Columns="3" Margin="10"/>

        </ScrollViewer>


        <StackPanel Grid.Row="3" Orientation="Horizontal"
HorizontalAlignment="Center" Margin="0,10">

            <Button Content="{DynamicResource solveDjik}" Width="150"
Click="Solve_Click" Style="{StaticResource ThemedButtonStyle}"/>
```

```
            <Button Content="INFO" Width="100" Click="open_info"
Style="{StaticResource ThemedButtonStyle}" Margin="20,0,0,0"/>
        </StackPanel>
    </Grid>
</Window>
--
using System;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace proiectoicd2025.main.matematica
{
    public partial class magicsquare : Window
    {
        private TextBox[,] gridBoxes;
        private int rows, cols, targetSum;

        public magicsquare()
        {
            InitializeComponent();
        }

        private void GenerateGrid_Click(object sender, RoutedEventArgs e)
        {
            if (!int.TryParse(RowInput.Text, out rows) ||
```

```
            !int.TryParse(ColumnInput.Text, out cols) ||
            !int.TryParse(SumInput.Text, out targetSum))
        {
            MessageBox.Show("Please enter valid numbers for rows, columns, and
target sum.", "Invalid Input", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }

        MagicGrid.Rows = rows;
        MagicGrid.Columns = cols;
        MagicGrid.Children.Clear();

        gridBoxes = new TextBox[rows, cols];

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                var box = new TextBox
                {
                    Margin = new Thickness(3),
                    Background = System.Windows.Media.Brushes.LightGray,
                    FontSize = 20,
                    Width = 50,
                    Height = 50,
                    TextAlignment = TextAlignment.Center,
                    VerticalContentAlignment = VerticalAlignment.Center
                };
```

```
            gridBoxes[i, j] = box;
            MagicGrid.Children.Add(box);
        }
    }
}

private async void Solve_Click(object sender, RoutedEventArgs e)
{
    if (gridBoxes == null) return;

    int[,] board = new int[rows, cols];
    bool[,] fixedCells = new bool[rows, cols];

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (int.TryParse(gridBoxes[i, j].Text, out int val) && val > 0)
            {
                board[i, j] = val;
                fixedCells[i, j] = true;
            }
            else
            {
                board[i, j] = 0;
                fixedCells[i, j] = false;
            }
```

```
            }
        }

        bool solved = await VisualSolve(board, fixedCells, 0, 0);
        if (!solved)
        {
            citafis("NOTFOUND");
        }
        else MessageBox.Show("Resolved!");
    }



    private async Task<bool> VisualSolve(int[,] board, bool[,] fixedCells, int
row, int col)
    {
        if (row == rows)
            return CheckValid(board);

        int nextRow = col == cols - 1 ? row + 1 : row;
        int nextCol = col == cols - 1 ? 0 : col + 1;

        if (fixedCells[row, col])
            return await VisualSolve(board, fixedCells, nextRow, nextCol);

        for (int num = 1; num <= rows * cols; num++)
        {
            if (!UsedInGrid(board, num))
            {
```

```
                    board[row, col] = num;


                    gridBoxes[row, col].Text = num.ToString();

                    gridBoxes[row, col].Background =
System.Windows.Media.Brushes.LightGreen;


                    await Task.Delay(100);


                    if (await VisualSolve(board, fixedCells, nextRow, nextCol))

                        return true;


                    board[row, col] = 0;

                    gridBoxes[row, col].Text = "";

                    gridBoxes[row, col].Background =
System.Windows.Media.Brushes.LightGray;


                    await Task.Delay(40);

                }

            }


            return false;

        }




        private bool UsedInGrid(int[,] board, int num)

        {

            foreach (var val in board)

                if (val == num)
```

```
            return true;
    return false;
}

private bool CheckValid(int[,] board)
{
    for (int i = 0; i < rows; i++)
    {
        if (board.Cast<int>().Any(v => v == 0)) return false;

        int rowSum = 0;
        for (int j = 0; j < cols; j++)
            rowSum += board[i, j];
        if (rowSum != targetSum) return false;
    }

    for (int j = 0; j < cols; j++)
    {
        int colSum = 0;
        for (int i = 0; i < rows; i++)
            colSum += board[i, j];
        if (colSum != targetSum) return false;
    }

    if (rows == cols)
    {
        int diag1 = 0, diag2 = 0;
```

```
            for (int i = 0; i < rows; i++)
            {
                diag1 += board[i, i];
                diag2 += board[i, rows - i - 1];
            }
            if (diag1 != targetSum || diag2 != targetSum)
                return false;
        }

        return true;
    }

    public bool infoOK = false;
    private main.misc.Info Info;
    string filePath;
    string language = "English";
    private void info_show(object sender, RoutedEventArgs e)
    {
        if (!infoOK)
        {
            Info = new main.misc.Info();
            filePath = string.Empty;
            if (language == "English")
            {
                filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\matematica\grafcoloratEN.txt");
            }
            else if (language == "Romanian")
```

```
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\grafcoloratRO.txt");
                }
                Info.SetContentFromFile(filePath);
                Info.Show();
                infoOK = true;
            }
            else
            {
                Info.Close();
                infoOK = false;
            }
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
```

```
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }
```

```
        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void open_info(object sender, RoutedEventArgs e)
        {

        }
        private void citafis(string key)
        {
            string message = Application.Current.Resources[key] as string;
            MessageBox.Show(message);
        }
    }
}
```

3. Operatori

```
<Window x:Class="proiectoicd2025.main.matematica.operatori"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Operatori" Height="500" Width="700"
```

```
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
```

```
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2" Margin="0,0,0,55" Grid.RowSpan="2">
            <Grid MouseDown="TitleBar_MouseDown">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="{DynamicResource mate3}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
```

```
Click="Minimize_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,0,5"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click" Style="{StaticResource ThemedButtonStyle}" Margin="0,5,5,5"/>

            </Grid>

        </Border>


        <StackPanel Grid.Row="1" Orientation="Horizontal" Margin="0,10"
HorizontalAlignment="Center">

            <TextBox x:Name="DigitsInput" MaxLength="5" Width="200" Height="30"
Background="#2C2C2C" Foreground="White" BorderBrush="Red" BorderThickness="2"
Margin="5" FontSize="14" HorizontalContentAlignment="Center"/>

            <TextBox x:Name="TargetInput" Width="100" Height="30" Background="#
2C2C2C" Foreground="White" BorderBrush="Red" BorderThickness="2" Margin="5"
FontSize="14" HorizontalContentAlignment="Center"/>

            <Button Content="{DynamicResource solveDjik}" Click="Solve_Click"
Style="{StaticResource ThemedButtonStyle}" Height="30" Width="100"/>

            <Button x:Name="info" Click="info_show" Style="{StaticResource
ThemedButtonStyle}" Height="30" Width="100" Margin="5" Content="INFO"/>

        </StackPanel>


        <Border Grid.Row="2" BorderBrush="Red" BorderThickness="2" CornerRadius="10"
Margin="20,10">

            <ScrollViewer>

                <ListBox x:Name="ResultsList" Background="#1E1E1E"
Foreground="White" BorderThickness="0" FontSize="14"/>

            </ScrollViewer>

        </Border>

    </Grid>

</Window>

--

using System;

using System.Collections.Generic;
```

```
using System.IO;
using System.Windows;
using System.Windows.Input;

namespace proiectoicd2025.main.matematica
{
    public partial class operatori : Window
    {
        public operatori()
        {
            InitializeComponent();
            DigitsInput.Text = "123";
            TargetInput.Text = "6";
        }

        private IEnumerable<string> GetPermutations(string str)
        {
            if (str.Length == 1) yield return str;
            else
            {
                for (int i = 0; i < str.Length; i++)
                {
                    var rest = str.Remove(i, 1);
                    foreach (var subPerm in GetPermutations(rest))
                    {
                        yield return str[i] + subPerm;
                    }
```

```
            }
        }
    }

    private async void Solve_Click(object sender, RoutedEventArgs e)
    {
        string digits = DigitsInput.Text.Trim();
        if (!int.TryParse(TargetInput.Text, out int target))
        {
            MessageBox.Show("Please enter a valid target number.", "Invalid
Input", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }

        ResultsList.Items.Clear();
        var results = new HashSet<string>();

        var initialResults = await Task.Run(() => AddOperators(digits, target));
        results.UnionWith(initialResults);

        foreach (var perm in GetPermutations(digits))
        {
            var permResults = await Task.Run(() => AddOperators(perm, target));
            results.UnionWith(permResults);
        }

        if (results.Count == 0)
        {
```

```
                ResultsList.Items.Add("No valid expressions found.");
            }
            else
            {
                foreach (var expr in results)
                {
                    ResultsList.Items.Add(expr);
                }
            }
        }

        private List<string> AddOperators(string num, int target)
        {
            var result = new List<string>();
            if (string.IsNullOrEmpty(num)) return result;
            Backtrack(result, "", num, target, 0, 0, 0);
            return result;
        }

        private void Backtrack(List<string> result, string path, string num, int
target, int pos, long eval, long multed)
        {
            if (pos == num.Length)
            {
                if (eval == target)
                    result.Add(path);
                return;
            }
```

```
            for (int i = pos; i < num.Length; i++)
            {
                if (i != pos && num[pos] == '0') break;

                string curStr = num.Substring(pos, i - pos + 1);
                long cur = long.Parse(curStr);

                if (pos == 0)
                {
                    Backtrack(result, path + curStr, num, target, i + 1, cur, cur);
                }
                else
                {
                    Backtrack(result, path + "+" + curStr, num, target, i + 1, eval
+ cur, cur);
                    Backtrack(result, path + "-" + curStr, num, target, i + 1,
eval - cur, -cur);
                    Backtrack(result, path + "*" + curStr, num, target, i + 1,
eval - multed + multed * cur, multed * cur);
                }
            }
        }

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }
```

```
        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        public bool infoOK = false;
        private main.misc.Info _Info;
        string filePath;
        string language = "English";
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                _Info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\matematica\operatoriEN.txt");
                }
                else if (language == "Romanian")
                {
```

```
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\operatoriRO.txt");
                }
                _Info.SetContentFromFile(filePath);
                _Info.Show();
                infoOK = true;
            }
            else
            {
                _Info.Close();
                infoOK = false;
            }
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
```

```
        }


        private void ChangeLanguage(string language)

        {

            var resourceDictionary = new ResourceDictionary();

            if (language == "English")

            {

                language = "English";

                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);

            }

            else if (language == "Romanian")

            {

                language = "Romanian";

                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);

            }


            Application.Current.Resources.MergedDictionaries.Clear();


Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

            this.Dispatcher.Invoke(() => this.InvalidateVisual());

        }

    }

}
```

4. Palindrom

```
<Window x:Class="proiectoicd2025.main.matematica.palindrom"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        Title="Palindrome Partitioning" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize"
        Loaded="Window_Loaded">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
```

```
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
```

```
                <TextBlock Text="{DynamicResource mate4}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>

                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click"

                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,0,5"/>

                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click"

                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,5,5"/>

            </Grid>

        </Border>



        <Border Grid.Row="1"  CornerRadius="10" Padding="0,10,0,0"
Margin="0,10,0,15">

            <UniformGrid Name="PalindromeGrid" Rows="1" Columns="1"
HorizontalAlignment="Center" VerticalAlignment="Center">

            </UniformGrid>

        </Border>



        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="10">

            <TextBox Name="InputString" Width="200" FontSize="14" Height="30"
Background="#2C2C2C" Foreground="White" BorderBrush="Red" BorderThickness="2"
Margin="0,0,10,0"/>

            <Button Content="{DynamicResource solveDjik}" Margin="0,0,0,0"
Click="Solve_Click" Style="{StaticResource ThemedButtonStyle}" Width="200"/>

            <Button Content="INFO" Click="open_info" Foreground="White"
Background="#2C2C2C" x:Name="info" Margin="30,0,0,0" FontSize="15" Width="100"
Height="25" Padding="0,0,0,0"/>

        </StackPanel>



        <ListBox Name="ResultListBox" Grid.Row="1" Margin="10,10,10,0" Background="#
2C2C2C" Foreground="White" BorderBrush="Red" BorderThickness="2" FontSize="14"
```

```
Padding="10"/>
    </Grid>
</Window>
--
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;

namespace proiectoicd2025.main.matematica
{
    public partial class palindrom : Window
    {
        public palindrom()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {

        }

        private void Solve_Click(object sender, RoutedEventArgs e)
        {
            string input = InputString.Text.Trim();
```

```
            if (string.IsNullOrEmpty(input))
            {
                MessageBox.Show("Please enter a string.", "Input Error",
MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            var result = GetPalindromePartitionsWithPermutations(input);
            DisplayResult(result);
        }

        private List<List<string>> GetPalindromePartitionsWithPermutations(string s)
        {
            var result = new HashSet<string>();
            var permutations = GetUniquePermutations(s.ToCharArray());

            foreach (var perm in permutations)
            {
                Backtrack(perm, 0, new List<string>(), result);
            }

            return result.Select(r => r.Split('|').ToList()).ToList();
        }

        private HashSet<string> GetUniquePermutations(char[] chars)
        {
            var result = new HashSet<string>();
            Permute(chars, 0, result);
```

```
    return result;
}

private void Permute(char[] chars, int start, HashSet<string> result)
{
    if (start == chars.Length)
    {
        result.Add(new string(chars));
        return;
    }

    var seen = new HashSet<char>();
    for (int i = start; i < chars.Length; i++)
    {
        if (seen.Contains(chars[i])) continue;
        seen.Add(chars[i]);

        Swap(chars, start, i);
        Permute(chars, start + 1, result);
        Swap(chars, start, i);
    }
}

private void Swap(char[] arr, int i, int j)
{
    var temp = arr[i];
    arr[i] = arr[j];
```

```
            arr[j] = temp;
        }

        private void Backtrack(string s, int start, List<string> current,
HashSet<string> result)
        {
            if (start == s.Length)
            {
                result.Add(string.Join("|", current));
                return;
            }

            for (int end = start + 1; end <= s.Length; end++)
            {
                var substring = s.Substring(start, end - start);
                if (IsPalindrome(substring))
                {
                    current.Add(substring);
                    Backtrack(s, end, current, result);
                    current.RemoveAt(current.Count - 1);
                }
            }
        }
        private bool IsPalindrome(string s)
        {
            int left = 0, right = s.Length - 1;
            while (left < right)
            {
```

```
            if (s[left] != s[right])
                return false;
            left++;
            right--;
        }
        return true;
    }

    private void DisplayResult(List<List<string>> result)
    {
        ResultListBox.Items.Clear();

        if (result.Count == 0)
        {
            ResultListBox.Items.Add("No Palindromes found.");
            return;
        }

        foreach (var partition in result)
        {
            ResultListBox.Items.Add(string.Join(" ; ", partition));
        }
    }

    private void Close_Click(object sender, RoutedEventArgs e)
    {
        Close();
```

```
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            WindowState = WindowState.Minimized;
        }

        private void TitleBar_MouseDown(object sender,
System.Windows.Input.MouseButtonEventArgs e)
        {
            if (e.LeftButton == System.Windows.Input.MouseButtonState.Pressed)
                DragMove();
        }

        private void open_info(object sender, RoutedEventArgs e)
        {

        }
        public bool infoOK = false;
        private main.misc.Info Info;
        string filePath;
        string language = "English";
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                Info = new main.misc.Info();
                filePath = string.Empty;
```

```
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\matematica\palindromEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\palindromRO.txt");
                }
                Info.SetContentFromFile(filePath);
                Info.Show();
                infoOK = true;
            }
            else
            {
                Info.Close();
                infoOK = false;
            }
        }
        public int curlang = 0;
        private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
        {
            if (curlang == 0)
            {
                ChangeLanguage("Romanian");
                language = "Romanian";
                curlang = 1;
```

```
            }
            else
            {
                ChangeLanguage("English");
                language = "English";
                curlang = 0;
            }
        }

        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
```

```
        }
    }
}
```

5. Subseturi

```
<Window x:Class="proiectoicd2025.main.matematica.subseturi"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Subset Sum Problem" Height="600" Width="600"
        Background="#1E1E1E"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        AllowsTransparency="True"
        ResizeMode="NoResize"
        Loaded="Window_Loaded">

    <Window.Resources>
        <Style x:Key="ThemedButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#2C2C2C"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="Template">
```

```
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
                                BorderBrush="{TemplateBinding BorderBrush}"
                                BorderThickness="{TemplateBinding BorderThickness}"
                                CornerRadius="8">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="Black"/>
                </Trigger>
            </Style.Triggers>
        </Style>
    </Window.Resources>

    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Background="#2C2C2C" BorderBrush="Red"
```

```
BorderThickness="0,0,0,2">
            <Grid MouseDown="TitleBar_MouseDown" Margin="0,0,0,-2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>

                <TextBlock Text="{DynamicResource mate5}" Foreground="White"
FontWeight="Bold" FontSize="16" VerticalAlignment="Center" Margin="10,0"/>
                <Button Grid.Column="1" Content="—" Width="40" Height="30"
Click="Minimize_Click"
                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,0,5"/>
                <Button Grid.Column="2" Content="X" Width="40" Height="30"
Click="Close_Click"
                        Style="{StaticResource ThemedButtonStyle}"
Margin="0,5,5,5"/>
            </Grid>
        </Border>

        <Border Grid.Row="1" BorderBrush="Red" BorderThickness="3" CornerRadius="10"
Padding="0,10,0,0" Margin="0,10,0,15">
            <UniformGrid Name="SubsetSumGrid" Rows="3" Columns="5"
HorizontalAlignment="Center" VerticalAlignment="Center">
            </UniformGrid>
        </Border>

        <StackPanel Grid.Row="2" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="10">
            <Button Content="{DynamicResource solveDjik}" Margin ="0,0,0,0"
```

```
Click="Solve_Click" Style="{StaticResource ThemedButtonStyle}" Width="200"/>

            <TextBlock x:Name="suma" FontSize="15" Text ="Sum:" Padding="10,0,0,0"
Foreground="White" Background="#2C2C2C" Width="100" Height="20" Margin="20,0,0,0"/>

            <Button Content="INFO" Click="open_info" Foreground="White"
Background="#2C2C2C" x:Name="info" Margin="30,0,0,0" FontSize="15" Width="100"
Height="25" Padding="0,0,0,0"/>

        </StackPanel>

    </Grid>

</Window>

--
using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;


namespace proiectoicd2025.main.matematica

{

    public partial class subseturi : Window

    {

        public subseturi()

        {

            InitializeComponent();

        }


        private void DisplaySolution(List<int> solution)

        {
```

```
            var cloned = new List<int>(solution);

            MessageBox.Show($"Solution found: {string.Join(", ", cloned)}",
"Solution", MessageBoxButton.OK, MessageBoxImage.Information);

        }


        private bool FindSubsetSum(List<int> nums, int target, List<int>
currentSubset, int index)

        {

            if (target == 0)

            {

                DisplaySolution(new List<int>(currentSubset));

                return true;

            }


            if (index >= nums.Count || target < 0)

                return false;


            currentSubset.Add(nums[index]);

            if (FindSubsetSum(nums, target - nums[index], currentSubset, index + 1))

                return true;

            currentSubset.RemoveAt(currentSubset.Count - 1);


            if (FindSubsetSum(nums, target, currentSubset, index + 1))

                return true;


            return false;

        }
```

```
        private void Solve_Click(object sender, RoutedEventArgs e)
        {
            string inputText = suma.Text;
            string digitsOnly = new string(inputText.Where(char.IsDigit).ToArray());

            if (!int.TryParse(digitsOnly, out int target))
            {
                MessageBox.Show("Something didn't go right!", "Invalid Input",
MessageBoxButton.OK, MessageBoxImage.Error);
            }

            List<int> nums = new List<int>();
            foreach (var child in SubsetSumGrid.Children)
            {
                if (child is TextBox textBox && !string.IsNullOrEmpty(textBox.Text))
                {
                    if (int.TryParse(textBox.Text, out int number))
                    {
                        nums.Add(number);
                    }
                }
            }

            List<int> currentSubset = new List<int>();
            bool found = FindSubsetSum(nums, target, currentSubset, 0);

            if (!found)
            {
```

```
                MessageBox.Show((string)Application.Current.Resources["NOTFOUND"],
"Not Found", MessageBoxButton.OK, MessageBoxImage.Information);
            }
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            int rows = 3;
            int cols = 5;
            Random rand = new Random();
            int total = 0;

            for (int i = 0; i < rows * cols; i++)
            {
                int value = rand.Next(1, 10);
                total += value;

                TextBox textBox = new TextBox
                {
                    Width = 50,
                    Height = 50,
                    FontSize = 20,
                    Margin = new Thickness(5),
                    HorizontalAlignment = HorizontalAlignment.Center,
                    VerticalAlignment = VerticalAlignment.Center,
                    Text = value.ToString(),
                    Background = new
System.Windows.Media.SolidColorBrush(System.Windows.Media.Color.FromRgb(44, 44,
44)),
```

```
                    Foreground = new
System.Windows.Media.SolidColorBrush(System.Windows.Media.Colors.White),
                    BorderBrush = new
System.Windows.Media.SolidColorBrush(System.Windows.Media.Colors.Red),
                    BorderThickness = new Thickness(3),
                    TextAlignment = TextAlignment.Center,
                };

                SubsetSumGrid.Children.Add(textBox);
                Grid.SetRow(textBox, i / cols);
                Grid.SetColumn(textBox, i % cols);
            }

            suma.Text = $"Sum: {total / 2}";
        }

        private void Minimize_Click(object sender, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void TitleBar_MouseDown(object sender, MouseButtonEventArgs e)
        {
            if (e.ChangedButton == MouseButton.Left)
                this.DragMove();
        }

        private void Close_Click(object sender, RoutedEventArgs e)
```

```
        {
            this.Close();
        }

        private void open_info(object sender, RoutedEventArgs e)
        {

        }

        public bool infoOK = false;
        private main.misc.Info Info;
        string filePath;
        string language = "English";
        private void info_show(object sender, RoutedEventArgs e)
        {
            if (!infoOK)
            {
                Info = new main.misc.Info();
                filePath = string.Empty;
                if (language == "English")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\matematica\subseturiEN.txt");
                }
                else if (language == "Romanian")
                {
                    filePath = Path.Combine(Directory.GetCurrentDirectory(),
@"resource\info\jocuri\subseturiRO.txt");
```

```
            }
            Info.SetContentFromFile(filePath);
            Info.Show();
            infoOK = true;
        }
        else
        {
            Info.Close();
            infoOK = false;
        }
    }
    public int curlang = 0;
    private void ChangeLanguage_Click(object sender, RoutedEventArgs e)
    {
        if (curlang == 0)
        {
            ChangeLanguage("Romanian");
            language = "Romanian";
            curlang = 1;
        }
        else
        {
            ChangeLanguage("English");
            language = "English";
            curlang = 0;
        }
    }
```

```
        private void ChangeLanguage(string language)
        {
            var resourceDictionary = new ResourceDictionary();
            if (language == "English")
            {
                language = "English";
                resourceDictionary.Source = new Uri("limbi/eng.xaml",
UriKind.Relative);
            }
            else if (language == "Romanian")
            {
                language = "Romanian";
                resourceDictionary.Source = new Uri("limbi/ro.xaml",
UriKind.Relative);
            }

            Application.Current.Resources.MergedDictionaries.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);
            this.Dispatcher.Invoke(() => this.InvalidateVisual());
        }
    }
}
```

SECȚIUNEA 3.5 Diverse

1. Clasa criptografie

```
using System;
using System.Collections.Generic;
```

```
using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;


namespace proiectoicd2025.main.misc

{

    public static class Criptografie

    {

        private static readonly byte[] Key = Encoding.UTF8.GetBytes("A7f!
g9D3s7L0m6Q1");

        private static readonly byte[] IV = Encoding.UTF8.GetBytes("B2k!
v6T9w4Z8n1E3");


        public static string Cripteaza(string text)

        {

            using Aes aes = Aes.Create();

            aes.Key = Key;

            aes.IV = IV;


            ICryptoTransform encryptor = aes.CreateEncryptor();

            using MemoryStream ms = new();

            using CryptoStream cs = new(ms, encryptor, CryptoStreamMode.Write);

            using (StreamWriter sw = new(cs))

            {

                sw.Write(text);

            }
```

```
            return Convert.ToBase64String(ms.ToArray());
        }

        public static string Decripteaza(string text)
        {
            byte[] buffer = Convert.FromBase64String(text);
            using Aes aes = Aes.Create();
            aes.Key = Key;
            aes.IV = IV;

            ICryptoTransform decryptor = aes.CreateDecryptor();
            using MemoryStream ms = new(buffer);
            using CryptoStream cs = new(ms, decryptor, CryptoStreamMode.Read);
            using StreamReader sr = new(cs);
            return sr.ReadToEnd();
        }
    }
}
```

2. Clasa manage

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace proiectoicd2025.main.misc
{
```

```
    public static class manage
    {
        public static string utilizatorCurent { get; set; }
    }
}
```

2. Quiz

```
<Window x:Class="proiectoicd2025.main.misc.quiz"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Recursion and Backtracking Quiz" Height="600" Width="900"
        WindowStyle="None" ResizeMode="CanResizeWithGrip"
        AllowsTransparency="True" Background="Transparent"
        WindowStartupLocation="CenterScreen">

    <Window.Resources>
        <Style x:Key="GameButtonStyle" TargetType="Button">
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="Background" Value="#FF4A0A0A"/>
            <Setter Property="FontSize" Value="18"/>
            <Setter Property="FontWeight" Value="SemiBold"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Padding" Value="10"/>
            <Setter Property="Height" Value="50"/>
            <Setter Property="Cursor" Value="Hand"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="FocusVisualStyle">
                <Setter.Value>
```

```
                    <Style TargetType="Button">
                        <Setter Property="BorderBrush" Value="Orange"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Style>
                </Setter.Value>
            </Setter>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Background="{TemplateBinding Background}"
CornerRadius="10">
                            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>

    <Border Background="#2D2D2D" CornerRadius="10">
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <Border Background="#3C3C3C" Grid.Row="0"
MouseDown="TitleBar_MouseDown">
```

```
                <DockPanel>

                    <TextBlock Text="Quiz" Foreground="White" FontSize="18"
VerticalAlignment="Center" Margin="10"/>

                    <StackPanel Orientation="Horizontal" HorizontalAlignment="Right"
DockPanel.Dock="Right">

                        <Button Content="−" Width="40" Height="30" Margin="0,0,5,0"
Click="Minimize_Click"

                                Background="Transparent" Foreground="White"
BorderThickness="0" FontSize="14"/>

                        <Button Content="×" Width="40" Height="30"
Click="Close_Click"

                                Background="Transparent" Foreground="White"
BorderThickness="0" FontSize="14"/>

                    </StackPanel>

                </DockPanel>

            </Border>


            <Grid Grid.Row="1" Margin="10">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition Width="2*" />

                    <ColumnDefinition Width="Auto" />

                    <ColumnDefinition Width="3*" />

                </Grid.ColumnDefinitions>


                <StackPanel Grid.Column="0" Margin="10" VerticalAlignment="Top">

                    <TextBlock x:Name="QuestionText" Foreground="White"
FontSize="22" TextWrapping="Wrap" Margin="0,0,0,20"/>

                    <StackPanel x:Name="OptionPanel">

                    </StackPanel>

                    <Button Content="Next Question" Style="{StaticResource
GameButtonStyle}" Click="NextQuestion_Click" Margin="0,20,0,0"/>
```

```
                </StackPanel>

                <Rectangle Grid.Column="1" Width="1" Fill="Gray" Margin="10"/>

                <Border Grid.Column="2" Background="#1E1E1E" CornerRadius="5"
Padding="10" Margin="10">
                    <ScrollViewer VerticalScrollBarVisibility="Auto">
                        <TextBlock x:Name="CodeSampleText" FontFamily="Consolas"
FontSize="16"
                                   Foreground="LightGreen" TextWrapping="Wrap"/>
                    </ScrollViewer>
                </Border>
            </Grid>
        </Grid>
    </Border>
</Window>

--
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace proiectoicd2025.main.misc
{
    public class Intrebare
```

```
    {
        public string Text_EN { get; set; }
        public string Text_RO { get; set; }
        public string Cod { get; set; }
        public string[] Optiuni_EN { get; set; }
        public string[] Optiuni_RO { get; set; }
        public int IndexCorect { get; set; }

        public string Text(string limba) => limba == "Romanian" ? Text_RO : Text_EN;
        public string[] Optiuni(string limba) => limba == "Romanian" ? Optiuni_RO :
Optiuni_EN;
    }

    public partial class quiz : Window
    {
        private List<Intrebare> intrebari;
        private int idx = 0;
        private string limba = "English";

        public quiz()
        {
            InitializeComponent();
            Incarca();
            Afiseaza();
        }

        private void Incarca()
        {
```

```
            string cale = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"main", "misc", "quiz.txt");

            intrebari = new List<Intrebare>();


            if (!File.Exists(cale))

            {

                MessageBox.Show("Quiz file not found!");

                return;

            }


            var linii = File.ReadAllLines(cale);

            Intrebare curenta = null;


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

            {

                var linie = linii[i].Trim();


                if (string.IsNullOrWhiteSpace(linie)) continue;


                if (linie.StartsWith("#QUESTION"))

                {

                    if (curenta != null && curenta.Optiuni_EN != null &&
curenta.Optiuni_RO != null)

                        intrebari.Add(curenta);

                    curenta = new Intrebare();

                }

                else if (curenta != null)

                {
```

```
                    i = ProceseazaLinie(curenta, linii, i);
                }
            }

            if (curenta != null && curenta.Optiuni_EN != null &&
curenta.Optiuni_RO != null)
                intrebari.Add(curenta);
        }

        private int ProceseazaLinie(Intrebare q, string[] linii, int i)
        {
            string linie = linii[i].Trim();

            if (linie.StartsWith("EN:"))
                q.Text_EN = linie.Substring(3).Trim();
            else if (linie.StartsWith("RO:"))
                q.Text_RO = linie.Substring(3).Trim();
            else if (linie.StartsWith("CODE:"))
            {
                i++;
                List<string> cod = new List<string>();

                while (i < linii.Length)
                {
                    string urm = linii[i].Trim();
                    if (urm.StartsWith("EN_OPTIONS:") ||
urm.StartsWith("RO_OPTIONS:") || urm.StartsWith("ANSWER:"))
                    {
```

```
                    i--;
                    break;
                }
                cod.Add(linii[i]);
                i++;
            }

            q.Cod = string.Join(Environment.NewLine, cod);
        }
        else if (linie.StartsWith("EN_OPTIONS:"))
            q.Optiuni_EN = linie.Substring(11).Trim().Split('|');
        else if (linie.StartsWith("RO_OPTIONS:"))
            q.Optiuni_RO = linie.Substring(11).Trim().Split('|');
        else if (linie.StartsWith("ANSWER:"))
            q.IndexCorect = int.TryParse(linie.Substring(7).Trim(), out int r) ?
r : 0;

        return i;
    }

    private void Afiseaza()
    {
        if (intrebari.Count == 0 || idx >= intrebari.Count)
        {
            QuestionText.Text = limba == "Romanian" ? "Nu mai sunt întrebări." :
"No more questions.";
            OptionPanel.Children.Clear();
            CodeSampleText.Visibility = Visibility.Collapsed;
```

```
                return;
            }

            var q = intrebari[idx];
            QuestionText.Text = q.Text(limba);
            CodeSampleText.Text = q.Cod;
            CodeSampleText.Visibility = string.IsNullOrWhiteSpace(q.Cod) ?
Visibility.Collapsed : Visibility.Visible;

            OptionPanel.Children.Clear();
            var optiuni = q.Optiuni(limba);
            foreach (var opt in optiuni)
                OptionPanel.Children.Add(Buton(opt));
        }

        private Button Buton(string text)
        {
            var btn = new Button
            {
                Content = text,
                Margin = new Thickness(5),
                Padding = new Thickness(10),
                FontSize = 16,
                FocusVisualStyle = null,
                Background = Brushes.LightGray,
                BorderBrush = Brushes.DarkGray,
                BorderThickness = new Thickness(1),
            };
```

```
    btn.Click += Selectie;
    return btn;
}


private void Selectie(object s, RoutedEventArgs e)
{
    var btn = s as Button;
    int ales = OptionPanel.Children.IndexOf(btn);
    var q = intrebari[idx];

    var original = btn.Background;

    if (ales == q.IndexCorect)
    {
        btn.Background = Brushes.LightGreen;
        MessageBox.Show(limba == "Romanian" ? "Corect!" : "Correct!", "✓");
    }
    else
    {
        btn.Background = Brushes.Red;
        MessageBox.Show(limba == "Romanian" ? "Greșit!" : "Wrong!", "✗");
    }

    var timer = new System.Windows.Threading.DispatcherTimer
    {
        Interval = TimeSpan.FromMilliseconds(1500)
```

```
    };
    timer.Tick += (sender2, args) =>
    {
        btn.Background = original;
        timer.Stop();
    };
    timer.Start();
}



private void NextQuestion_Click(object s, RoutedEventArgs e)
{
    idx++;
    Afiseaza();
}

private void SchimbaLimba(object s, RoutedEventArgs e)
{
    limba = limba == "English" ? "Romanian" : "English";
    Afiseaza();
}

private void TitleBar_MouseDown(object s, MouseButtonEventArgs e)
{
    if (e.ChangedButton == MouseButton.Left)
        this.DragMove();
}
```

```
        private void Minimize_Click(object s, RoutedEventArgs e)
        {
            this.WindowState = WindowState.Minimized;
        }

        private void Close_Click(object s, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```