Thank you for sharing the complete code and providing detailed requirements for the two new features. I’ve thoroughly reviewed your code and will implement the changes while staying as close as possible to your existing code structure, style, and MVVM pattern. Below, I’ll outline the two features, explain the necessary changes, and provide the modified code wrapped in `<xaiArtifact>` tags with precise indications of where and why changes were made. I’ll ensure the changes integrate seamlessly with your existing codebase, using your existing classes, converters, and settings.

### Feature Requirements

1. \*\*Configurable Single Deadline for Krias Shema and Tefila\*\*:

- Allow users to configure the app to show only one deadline for Krias Shema (`EOS1`) and one for Tefila (`EOT1`) instead of the default two deadlines each (`EOS1`, `EOS2`, `EOT1`, `EOT2`).

- When configured for single deadlines:

- The GUI should display `EOS1` on the top row and `EOT1` on the bottom row in normal mode (2x1 grid).

- When a deadline approaches (within `VisualAlertMinutes`), the corresponding rectangle becomes large and moves to the top row, with the other remaining small on the bottom row, consistent with the current alert mode behavior.

- This will require a new setting in `OptionsViewModel` and updates to `MainViewModel` and `MainWindow.xaml` to handle the single-deadline mode.

2. \*\*Sefirat HaOmer Display\*\*:

- During the Sefirat HaOmer period (from 16 Nisan to 6 Sivan in the Jewish calendar), display a single large rectangle in the main content area of `MainWindow.xaml`.

- The rectangle should:

- Use a new background image (`haomerBackground.png`) from the `Assets` folder.

- Display the Hebrew text for the current Omer count (e.g., for the 10th day: "היום עשרה ימים שהם שבוע אחד ושלשה ימים לעומר").

- This should replace the normal `EOS`/`EOT` display during the Omer period but maintain the header and footer.

### Approach

- \*\*Feature 1\*\*: Add a new setting (`UseSingleDeadline`) to `OptionsViewModel` and persist it in `Properties.Settings`. Update `MainViewModel` to filter `TimeSlots` to include only `EOS1` and `EOT1` when this setting is enabled. Modify `MainWindow.xaml` to adjust the `ItemsControl` layout for a 2x1 grid in normal mode when using single deadlines.

- \*\*Feature 2\*\*: Add logic in `MainViewModel` to detect the Sefirat HaOmer period using the Hebrew calendar and calculate the Omer count. Update `MainWindow.xaml` to conditionally show a new `Border` with the Omer text and background when in the Omer period, hiding the `EOS`/`EOT` display.

- \*\*Code Style\*\*: I’ll reuse your existing converters (`BoolToBrushConverter`, `ResourceToImageSourceConverter`), styles (`TimeBoxStyle`, `BigTimeBoxStyle`), and MVVM structure. I’ll also ensure logging via `Logger.cs` and proper integration with `Properties.Settings`.

### Changes Overview

I’ll modify the following files:

1. \*\*Properties/Settings.settings\*\* (and `Settings.Designer.cs`): Add new settings for `UseSingleDeadline` and `SefiratHaOmerEnabled` (optional, for toggling Omer display).

2. \*\*ViewModels/OptionsViewModel.cs\*\*: Add `UseSingleDeadline` property and UI bindings for the new setting.

3. \*\*Views/OptionsWindow.xaml\*\*: Add a checkbox for the single-deadline option.

4. \*\*ViewModels/MainViewModel.cs\*\*: Filter `TimeSlots` for single deadlines and add logic for Sefirat HaOmer detection and text generation.

5. \*\*Views/MainWindow.xaml\*\*: Adjust the `ItemsControl` for a 2x1 grid in single-deadline mode and add a new `Border` for the Sefirat HaOmer display.

### Detailed Changes

#### 1. Properties/Settings.settings

- \*\*Purpose\*\*: Add new settings for `UseSingleDeadline` (to toggle single-deadline mode) and `SefiratHaOmerEnabled` (to allow users to enable/disable Omer display).

- \*\*Changes\*\*:

- Add two boolean settings to `Settings.settings` (manually edited in Visual Studio or the XML file).

- The generated `Settings.Designer.cs` will include these properties.

```xml

<?xml version='1.0' encoding='utf-8'?>

<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings" CurrentProfile="(Default)" GeneratedClassNamespace="EOTReminder.Properties" GeneratedClassName="Settings">

<Profiles />

<Settings>

<Setting Name="ExcelFilePath" Type="System.String" Scope="User">

<Value Profile="(Default)" />

</Setting>

<Setting Name="FirstAlertMinutes" Type="System.Int32" Scope="User">

<Value Profile="(Default)">10</Value>

</Setting>

<Setting Name="SecondAlertMinutes" Type="System.Int32" Scope="User">

<Value Profile="(Default)">3</Value>

</Setting>

<Setting Name="EOS1FirstAlertPath" Type="System.String" Scope="User">

<Value Profile="(Default)" />

</Setting>

<Setting Name="EOS1SecondAlertPath" Type="System.String" Scope="User">

<Value Profile="(Default)" />

</Setting>

<Setting Name="EOS2FirstAlertPath" Type="System.String" Scope="User">

<Value Profile="(Default)" />

</Setting>

<Setting Name="EOS2SecondAlertPath" Type="System.String" Scope="User">

<Value Profile="(Default)" />

</Setting>

<Setting Name="VisualAlertMinutes" Type="System.Int32" Scope="User">

<Value Profile="(Default)">30</Value>

</Setting>

<Setting Name="AlertOnShabbos" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<!-- New settings -->

<Setting Name="UseSingleDeadline" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">False</Value>

</Setting>

<Setting Name="SefiratHaOmerEnabled" Type="System.Boolean" Scope="User">

<Value Profile="(Default)">True</Value>

</Setting>

</Settings>

</SettingsFile>

```

```x-csharp

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated by a tool.

// Runtime Version:4.0.30319.42000

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

namespace EOTReminder.Properties {

using System;

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator", "17.6.0.0")]

internal sealed partial class Settings : global::System.Configuration.ApplicationSettingsBase {

private static Settings defaultInstance = ((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new Settings())));

public static Settings Default {

get {

return defaultInstance;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("")]

public string ExcelFilePath {

get {

return ((string)(this["ExcelFilePath"]));

}

set {

this["ExcelFilePath"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("10")]

public int FirstAlertMinutes {

get {

return ((int)(this["FirstAlertMinutes"]));

}

set {

this["FirstAlertMinutes"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("3")]

public int SecondAlertMinutes {

get {

return ((int)(this["SecondAlertMinutes"]));

}

set {

this["SecondAlertMinutes"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("")]

public string EOS1FirstAlertPath {

get {

return ((string)(this["EOS1FirstAlertPath"]));

}

set {

this["EOS1FirstAlertPath"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("")]

public string EOS1SecondAlertPath {

get {

return ((string)(this["EOS1SecondAlertPath"]));

}

set {

this["EOS1SecondAlertPath"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("")]

public string EOS2FirstAlertPath {

get {

return ((string)(this["EOS2FirstAlertPath"]));

}

set {

this["EOS2FirstAlertPath"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("")]

public string EOS2SecondAlertPath {

get {

return ((string)(this["EOS2SecondAlertPath"]));

}

set {

this["EOS2SecondAlertPath"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("30")]

public int VisualAlertMinutes {

get {

return ((int)(this["VisualAlertMinutes"]));

}

set {

this["VisualAlertMinutes"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("False")]

public bool AlertOnShabbos {

get {

return ((bool)(this["AlertOnShabbos"]));

}

set {

this["AlertOnShabbos"] = value;

}

}

// New settings

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("False")]

public bool UseSingleDeadline {

get {

return ((bool)(this["UseSingleDeadline"]));

}

set {

this["UseSingleDeadline"] = value;

}

}

[global::System.Configuration.UserScopedSettingAttribute()]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.DefaultSettingValueAttribute("True")]

public bool SefiratHaOmerEnabled {

get {

return ((bool)(this["SefiratHaOmerEnabled"]));

}

set {

this["SefiratHaOmerEnabled"] = value;

}

}

}

}

```

\*\*Purpose\*\*:

- Added `UseSingleDeadline` to enable/disable single-deadline mode.

- Added `SefiratHaOmerEnabled` to allow users to toggle the Sefirat HaOmer display.

- These settings are persisted in `Properties.Settings` and accessible in `OptionsViewModel`.

#### 2. ViewModels/OptionsViewModel.cs

- \*\*Purpose\*\*: Add the `UseSingleDeadline` and `SefiratHaOmerEnabled` properties, load/save them, and update the UI bindings.

- \*\*Changes\*\*:

- Lines 29–40: Added new properties for `UseSingleDeadline` and `SefiratHaOmerEnabled`.

- Lines 108–113: Load the new settings in `LoadSettings`.

- Lines 150–155: Save the new settings in `SaveSettings`.

```x-csharp

using System;

using System.ComponentModel;

using System.IO;

using System.Runtime.CompilerServices;

using System.Windows;

using System.Windows.Input;

using Microsoft.Win32; // For OpenFileDialog

using EOTReminder.Utilities;

namespace EOTReminder.ViewModels

{

public class OptionsViewModel : INotifyPropertyChanged

{

// Existing Settings

private int \_firstAlertMinutes;

public int FirstAlertMinutes

{

get => \_firstAlertMinutes;

set { \_firstAlertMinutes = value; OnPropertyChanged(); }

}

private int \_secondAlertMinutes;

public int SecondAlertMinutes

{

get => \_secondAlertMinutes;

set { \_secondAlertMinutes = value; OnPropertyChanged(); }

}

private string \_excelFilePath;

public string ExcelFilePath

{

get => \_excelFilePath;

set { \_excelFilePath = value; OnPropertyChanged(); }

}

// NEW: UseSingleDeadline setting

private bool \_useSingleDeadline;

public bool UseSingleDeadline

{

get => \_useSingleDeadline;

set { \_useSingleDeadline = value; OnPropertyChanged(); }

}

// NEW: SefiratHaOmerEnabled setting

private bool \_sefiratHaOmerEnabled;

public bool SefiratHaOmerEnabled

{

get => \_sefiratHaOmerEnabled;

set { \_sefiratHaOmerEnabled = value; OnPropertyChanged(); }

}

// Existing Audio Alert Paths

private string \_eos1FirstAlertPath;

public string EOS1FirstAlertPath

{

get => \_eos1FirstAlertPath;

set { \_eos1FirstAlertPath = value; OnPropertyChanged(); }

}

private string \_eos1SecondAlertPath;

public string EOS1SecondAlertPath

{

get => \_eos1SecondAlertPath;

set { \_eos1SecondAlertPath = value; OnPropertyChanged(); }

}

private string \_eos2FirstAlertPath;

public string EOS2FirstAlertPath

{

get => \_eos2FirstAlertPath;

set { \_eos2FirstAlertPath = value; OnPropertyChanged(); }

}

private string \_eos2SecondAlertPath;

public string EOS2SecondAlertPath

{

get => \_eos2SecondAlertPath;

set { \_eos2SecondAlertPath = value; OnPropertyChanged(); }

}

private string \_eot1FirstAlertPath;

public string EOT1FirstAlertPath

{

get => \_eot1FirstAlertPath;

set { \_eot1FirstAlertPath = value; OnPropertyChanged(); }

}

private string \_eot1SecondAlertPath;

public string EOT1SecondAlertPath

{

get => \_eot1SecondAlertPath;

set { \_eot1SecondAlertPath = value; OnPropertyChanged(); }

}

private string \_eot2FirstAlertPath;

public string EOT2FirstAlertPath

{

get => \_eot2FirstAlertPath;

set { \_eot2FirstAlertPath = value; OnPropertyChanged(); }

}

private string \_eot2SecondAlertPath;

public string EOT2SecondAlertPath

{

get => \_eot2SecondAlertPath;

set { \_eot2SecondAlertPath = value; OnPropertyChanged(); }

}

private int \_visualAlertMinutes;

public int VisualAlertMinutes

{

get => \_visualAlertMinutes;

set { \_visualAlertMinutes = value; OnPropertyChanged(); }

}

private bool \_alertOnShabbos;

public bool AlertOnShabbos

{

get => \_alertOnShabbos;

set { \_alertOnShabbos = value; OnPropertyChanged(); }

}

// Commands

public ICommand SaveSettingsCommand { get; }

public ICommand CloseApplicationCommand { get; }

public ICommand CloseSettingsCommand { get; }

public ICommand BrowseExcelCommand { get; }

public ICommand BrowseEOS1FirstAlertCommand { get; }

public ICommand BrowseEOS1SecondAlertCommand { get; }

public ICommand BrowseEOS2FirstAlertCommand { get; }

public ICommand BrowseEOS2SecondAlertCommand { get; }

public ICommand BrowseEOT1FirstAlertCommand { get; }

public ICommand BrowseEOT1SecondAlertCommand { get; }

public ICommand BrowseEOT2FirstAlertCommand { get; }

public ICommand BrowseEOT2SecondAlertCommand { get; }

public OptionsViewModel()

{

LoadSettings();

SaveSettingsCommand = new RelayCommand(SaveSettings);

CloseApplicationCommand = new RelayCommand(CloseApplication);

CloseSettingsCommand = new RelayCommand(CloseSettings);

BrowseExcelCommand = new RelayCommand(BrowseExcelFile);

BrowseEOS1FirstAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOS1FirstAlertPath)));

BrowseEOS1SecondAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOS1SecondAlertPath)));

BrowseEOS2FirstAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOS2FirstAlertPath)));

BrowseEOS2SecondAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOS2SecondAlertPath)));

BrowseEOT1FirstAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOT1FirstAlertPath)));

BrowseEOT1SecondAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOT1SecondAlertPath)));

BrowseEOT2FirstAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOT2FirstAlertPath)));

BrowseEOT2SecondAlertCommand = new RelayCommand(param => BrowseAudioFile(nameof(EOT2SecondAlertPath)));

}

private void LoadSettings()

{

FirstAlertMinutes = Properties.Settings.Default.FirstAlertMinutes;

SecondAlertMinutes = Properties.Settings.Default.SecondAlertMinutes;

ExcelFilePath = Properties.Settings.Default.ExcelFilePath;

EOS1FirstAlertPath = Properties.Settings.Default.EOS1FirstAlertPath;

EOS1SecondAlertPath = Properties.Settings.Default.EOS1SecondAlertPath;

EOS2FirstAlertPath = Properties.Settings.Default.EOS2FirstAlertPath;

EOS2SecondAlertPath = Properties.Settings.Default.EOS2SecondAlertPath;

VisualAlertMinutes = Properties.Settings.Default.VisualAlertMinutes;

AlertOnShabbos = Properties.Settings.Default.AlertOnShabbos;

// NEW: Load new settings

UseSingleDeadline = Properties.Settings.Default.UseSingleDeadline;

SefiratHaOmerEnabled = Properties.Settings.Default.SefiratHaOmerEnabled;

Logger.LogInfo("Application settings loaded.");

}

private void SaveSettings(object parameter)

{

Properties.Settings.Default.FirstAlertMinutes = FirstAlertMinutes;

Properties.Settings.Default.SecondAlertMinutes = SecondAlertMinutes;

Properties.Settings.Default.ExcelFilePath = ExcelFilePath;

Properties.Settings.Default.EOS1FirstAlertPath = EOS1FirstAlertPath;

Properties.Settings.Default.EOS1SecondAlertPath = EOS1SecondAlertPath;

Properties.Settings.Default.EOS2FirstAlertPath = EOS2FirstAlertPath;

Properties.Settings.Default.EOS2SecondAlertPath = EOS2SecondAlertPath;

Properties.Settings.Default.VisualAlertMinutes = VisualAlertMinutes;

Properties.Settings.Default.AlertOnShabbos = AlertOnShabbos;

// NEW: Save new settings

Properties.Settings.Default.UseSingleDeadline = UseSingleDeadline;

Properties.Settings.Default.SefiratHaOmerEnabled = SefiratHaOmerEnabled;

Properties.Settings.Default.Save();

Logger.LogInfo("Application settings saved successfully.");

if (parameter is Window window)

{

window.Close();

}

}

private void CloseSettings(object parameter)

{

Logger.LogInfo("Settings window close requested.");

if (parameter is Window window)

{

window.Close();

}

}

private void CloseApplication(object parameter)

{

Logger.LogInfo("Application close requested from options window.");

if (parameter is Window window)

{

window.Close();

}

Application.Current.Shutdown();

}

private void BrowseExcelFile(object parameter)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Excel Files (\*.xlsx;\*.xls)|\*.xlsx;\*.xls|All Files (\*.\*)|\*.\*";

openFileDialog.InitialDirectory = GetInitialDirectory(ExcelFilePath);

try

{

if (openFileDialog.ShowDialog() == true)

{

ExcelFilePath = openFileDialog.FileName;

Logger.LogInfo($"Excel file path set to: {ExcelFilePath}");

}

}

catch (Exception ex)

{

Logger.LogError($"Error opening file dialog for Excel file: {ex.Message}", ex);

}

}

private void BrowseAudioFile(string propertyName)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "WAV Audio Files (\*.wav)|\*.wav|All Files (\*.\*)|\*.\*";

string currentPath = GetPropertyValue(propertyName) as string;

openFileDialog.InitialDirectory = GetInitialDirectory(currentPath);

try

{

if (openFileDialog.ShowDialog() == true)

{

SetPropertyValue(propertyName, openFileDialog.FileName);

Logger.LogInfo($"Audio file path for {propertyName} set to: {openFileDialog.FileName}");

}

}

catch (Exception ex)

{

Logger.LogError($"Error opening file dialog for audio file ({propertyName}): {ex.Message}", ex);

}

}

private string GetInitialDirectory(string currentPath)

{

if (!string.IsNullOrWhiteSpace(currentPath) && File.Exists(currentPath))

{

return Path.GetDirectoryName(currentPath);

}

if (!string.IsNullOrWhiteSpace(currentPath) && Directory.Exists(currentPath))

{

return currentPath;

}

return Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

}

private object GetPropertyValue(string propertyName)

{

return GetType().GetProperty(propertyName)?.GetValue(this);

}

private void SetPropertyValue(string propertyName, object value)

{

GetType().GetProperty(propertyName)?.SetValue(this, value);

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

public class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Predicate<object> \_canExecute;

public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)

{

\_execute = execute ?? throw new ArgumentNullException(nameof(execute));

\_canExecute = canExecute;

}

public bool CanExecute(object parameter) => \_canExecute == null || \_canExecute(parameter);

public void Execute(object parameter) => \_execute(parameter);

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

}

}

```

#### 3. Views/OptionsWindow.xaml

- \*\*Purpose\*\*: Add checkboxes for `UseSingleDeadline` and `SefiratHaOmerEnabled` in the options UI.

- \*\*Changes\*\*:

- Lines 53–56: Added new rows for the checkboxes.

- Lines 332–341: Added `TextBlock` and `CheckBox` for `UseSingleDeadline`.

- Lines 342–351: Added `TextBlock` and `CheckBox` for `SefiratHaOmerEnabled`.

```xml

<Window x:Class="EOTReminder.Views.OptionsWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:vm="clr-namespace:EOTReminder.ViewModels"

mc:Ignorable="d"

Title="Application Options" Height="650" Width="600"

WindowStartupLocation="CenterScreen"

WindowStyle="ToolWindow"

ResizeMode="NoResize">

<Window.DataContext>

<vm:OptionsViewModel/>

</Window.DataContext>

<Grid Margin="15">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="10"/>

<RowDefinition Height="Auto"/>

<!-- NEW: Added rows for new settings -->

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="Excel File Path:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="0" Grid.Column="1" Text="{Binding ExcelFilePath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="0" Grid.Column="2" Content="Browse..." Command="{Binding BrowseExcelCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="3" Grid.Column="0" Text="First Audio Alert (minutes before):" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="3" Grid.Column="1" Text="{Binding FirstAlertMinutes, UpdateSourceTrigger=PropertyChanged}" Margin="5" Width="150" HorizontalAlignment="Left"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="Second Audio Alert (minutes before):" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="4" Grid.Column="1" Text="{Binding SecondAlertMinutes, UpdateSourceTrigger=PropertyChanged}" Margin="5" Width="150" HorizontalAlignment="Left"/>

<TextBlock Grid.Row="5" Grid.Column="0" Text="Visual Alert (minutes before):" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="5" Grid.Column="1" Text="{Binding VisualAlertMinutes, UpdateSourceTrigger=PropertyChanged}" Margin="5" Width="150" HorizontalAlignment="Left"/>

<TextBlock Grid.Row="7" Grid.Column="0" Text="EOS1 First Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="7" Grid.Column="1" Text="{Binding EOS1FirstAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="7" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOS1FirstAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="9" Grid.Column="0" Text="EOS1 Second Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="9" Grid.Column="1" Text="{Binding EOS1SecondAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="9" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOS1SecondAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="12" Grid.Column="0" Text="EOS2 First Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="12" Grid.Column="1" Text="{Binding EOS2FirstAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="12" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOS2FirstAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="14" Grid.Column="0" Text="EOS2 Second Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="14" Grid.Column="1" Text="{Binding EOS2SecondAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="14" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOS2SecondAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="17" Grid.Column="0" Text="EOT1 First Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="17" Grid.Column="1" Text="{Binding EOT1FirstAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="17" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOT1FirstAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="19" Grid.Column="0" Text="EOT1 Second Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="19" Grid.Column="1" Text="{Binding EOT1SecondAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="19" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOT1SecondAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="22" Grid.Column="0" Text="EOT2 First Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="22" Grid.Column="1" Text="{Binding EOT2FirstAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="22" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOT2FirstAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="24" Grid.Column="0" Text="EOT2 Second Alert Audio:" Margin="5" VerticalAlignment="Center"/>

<TextBox Grid.Row="24" Grid.Column="1" Text="{Binding EOT2SecondAlertPath, UpdateSourceTrigger=PropertyChanged}" Margin="5"/>

<Button Grid.Row="24" Grid.Column="2" Content="Browse..." Command="{Binding BrowseEOT2SecondAlertCommand}" Margin="5" Width="80"/>

<TextBlock Grid.Row="27" Grid.Column="0" Text="Alert on Shabbos:" Margin="5" VerticalAlignment="Center"/>

<CheckBox Grid.Row="27" Grid.Column="1" IsChecked="{Binding AlertOnShabbos}" Margin="5" VerticalAlignment="Center" HorizontalAlignment="Left"/>

<!-- NEW: Single Deadline Checkbox -->

<TextBlock Grid.Row="28" Grid.Column="0" Text="Use Single Deadline (EOS1 & EOT1 only):" Margin="5" VerticalAlignment="Center"/>

<CheckBox Grid.Row="28" Grid.Column="1" IsChecked="{Binding UseSingleDeadline}" Margin="5" VerticalAlignment="Center" HorizontalAlignment="Left"/>

<!-- NEW: Sefirat HaOmer Checkbox -->

<TextBlock Grid.Row="29" Grid.Column="0" Text="Show Sefirat HaOmer:" Margin="5" VerticalAlignment="Center"/>

<CheckBox Grid.Row="29" Grid.Column="1" IsChecked="{Binding SefiratHaOmerEnabled}" Margin="5" VerticalAlignment="Center" HorizontalAlignment="Left"/>

<Grid Grid.Row="31" Grid.ColumnSpan="3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Orientation="Horizontal" HorizontalAlignment="Left" Margin="0,10,0,0">

<Button Content="Save Settings"

Command="{Binding SaveSettingsCommand}"

CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"

Margin="0,0,10,0" Padding="10,5"/>

<Button Content="Cancel"

Command="{Binding CloseSettingsCommand}"

CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}"

Padding="10,5"/>

</StackPanel>

<StackPanel Grid.Column="1" Orientation="Horizontal" HorizontalAlignment="Right" Margin="0,10,0,0">

<Button Content="Close Application" Command="{Binding CloseApplicationCommand}" CommandParameter="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Window}}}" Padding="10,5"/>

</StackPanel>

</Grid>

</Grid>

</Window>

```

#### 4. ViewModels/MainViewModel.cs

- \*\*Purpose\*\*:

- Filter `TimeSlots` to include only `EOS1` and `EOT1` when `UseSingleDeadline` is true.

- Add logic to detect the Sefirat HaOmer period and generate the Omer count text in Hebrew.

- \*\*Changes\*\*:

- Line 35: Added `IsSefiratHaOmer` property for binding to control Omer display visibility.

- Line 36: Added `SefiratHaOmerText` property for the Omer count text.

- Lines 115–121: Added `HebrewNumberWords` dictionary for converting numbers to Hebrew words.

- Lines 586–589: Modified `LoadFromExcel` to filter slots based on `UseSingleDeadline`.

- Lines 665–692: Added `IsInSefiratHaOmerPeriod` to check if the current date is in the Omer period.

- Lines 693–752: Added `GetSefiratHaOmerText` to generate the Hebrew Omer count text.

- Lines 753–757: Modified `LoadMock` to include Sefirat HaOmer logic.

- Lines 758–762: Modified `AddSlot` to notify `IsSefiratHaOmer` changes.

- Lines 763–767: Modified `UpdateSlotCollections` to notify `IsSefiratHaOmer` changes.

```x-csharp

using EOTReminder.Models;

using EOTReminder.Utilities;

using ExcelDataReader;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Media;

using System.Net;

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Text;

using System.Timers;

using System.Windows;

namespace EOTReminder.ViewModels

{

public class MainViewModel : INotifyPropertyChanged

{

public ObservableCollection<TimeSlot> TimeSlots { get; set; } = new ObservableCollection<TimeSlot>();

public ObservableCollection<TimeSlot> TopSlots { get; } = new ObservableCollection<TimeSlot>();

public ObservableCollection<TimeSlot> BottomSlots { get; } = new ObservableCollection<TimeSlot>();

private bool \_isAlertActive;

private DateTime \_lastExcelReloadDate = DateTime.MinValue;

private bool \_hasReloadedForCurrentSunriseCycle = false;

private DateTime \_currentSunriseForReloadCheck = DateTime.MinValue;

private bool \_isSefiratHaOmer; // NEW: For Sefirat HaOmer visibility

private string \_sefiratHaOmerText; // NEW: For Sefirat HaOmer text

public bool IsAlertActive

{

get => \_isAlertActive;

set { \_isAlertActive = value; OnPropertyChanged(); }

}

public bool IsAlertNotActive

{

get => !\_isAlertActive;

set { \_isAlertNotActive = value; OnPropertyChanged(); }

}

public bool IsSefiratHaOmer // NEW: Property to control Omer display

{

get => \_isSefiratHaOmer;

set { \_isSefiratHaOmer = value; OnPropertyChanged(); }

}

public string SefiratHaOmerText // NEW: Property for Omer count text

{

get => \_sefiratHaOmerText;

set { \_sefiratHaOmerText = value; OnPropertyChanged(); }

}

public string TodayDate => DateTime.Now.ToString("dd/MM/yyyy");

public string CurrentTime => DateTime.Now.ToString("HH:mm:ss");

private DateTime \_internalSunriseTime;

private DateTime \_internalMiddayTime;

private DateTime \_internalSunsetTime;

private string \_hebrewDateString;

public string HebrewDate

{

get => \_hebrewDateString;

private set { \_hebrewDateString = value; OnPropertyChanged(); }

}

public string Sunrise

{

get => \_internalSunriseTime == DateTime.MinValue ? "N/A" : \_internalSunriseTime.ToString("HH:mm:ss");

private set { /\* Setter is not used \*/ }

}

public string Midday

{

get => \_internalMiddayTime == DateTime.MinValue ? "N/A" : \_internalMiddayTime.ToString("HH:mm:ss");

private set { /\* Setter is not used \*/ }

}

public string Sunset

{

get => \_internalSunsetTime == DateTime.MinValue ? "N/A" : \_internalSunsetTime.ToString("HH:mm:ss");

private set { /\* Setter is not used \*/ }

}

public event PropertyChangedEventHandler PropertyChanged;

private Timer \_timer;

private string \_currentLang = "he";

private readonly Dictionary<string, Dictionary<string, string>> \_translations =

new Dictionary<string, Dictionary<string, string>>()

{

["en"] = new Dictionary<string, string>()

{

["a2EOS1"] = "End of Shema 1",

["a1EOS2"] = "End of Shema 2",

["b2EOT1"] = "End of Prayer 1",

["b1EOT2"] = "End of Prayer 2",

["Passed"] = "Passed"

},

["he"] = new Dictionary<string, string>()

{

["a2EOS1"] = "סו\"ז קר\"ש מג\"א",

["a1EOS2"] = "סו\"ז קר\"ש תניא גר\"א",

["b2EOT1"] = "סו\"ז תפילה מג\"א",

["b1EOT2"] = "סו\"ז תפילה תניא גר\"א",

["Passed"] = "עבר זמנו",

}

};

// NEW: Dictionary for Hebrew number words

private readonly Dictionary<int, string> \_hebrewNumberWords = new Dictionary<int, string>

{

{ 1, "אחד" }, { 2, "שניים" }, { 3, "שלשה" }, { 4, "ארבעה" }, { 5, "חמשה" },

{ 6, "ששה" }, { 7, "שבעה" }, { 8, "שמונה" }, { 9, "תשעה" }, { 10, "עשרה" },

{ 11, "אחד עשר" }, { 12, "שניים עשר" }, { 13, "שלשה עשר" }, { 14, "ארבעה עשר" },

{ 15, "חמשה עשר" }, { 16, "ששה עשר" }, { 17, "שבעה עשר" }, { 18, "שמונה עשר" },

{ 19, "תשעה עשר" }, { 20, "עשרים" }, { 30, "שלושים" }, { 40, "ארבעים" }, { 50, "חמישים" }

};

public MainViewModel()

{

System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);

LoadFromExcel();

InitTimer();

}

public void InitializeData()

{

}

private void InitTimer()

{

\_timer = new Timer(1000);

\_timer.Elapsed += (s, e) =>

{

Application.Current.Dispatcher.Invoke(() =>

{

foreach (var slot in TimeSlots)

{

slot.Countdown = slot.Time - DateTime.Now;

int firstAlertMin = Properties.Settings.Default.FirstAlertMinutes;

int secondAlertMin = Properties.Settings.Default.SecondAlertMinutes;

int visualAlertMin = Properties.Settings.Default.VisualAlertMinutes;

if (!slot.IsPassed && slot.Countdown <= TimeSpan.Zero)

{

slot.Highlight = false;

slot.IsPassed = true;

slot.CountdownText = "";

slot.ShowSandClock = false;

slot.IsIn30MinAlert = false;

slot.AlertFlags["30"] = false;

slot.AlertFlags["10"] = false;

slot.AlertFlags["3"] = false;

IsAlertActive = false;

}

else if (!slot.IsPassed)

{

if (slot.Countdown.TotalMinutes <= visualAlertMin && !slot.AlertFlags["30"])

{

IsAlertActive = true;

slot.IsIn30MinAlert = true;

slot.Highlight = true;

slot.ShowSandClock = true;

slot.AlertFlags["30"] = true;

}

else if (slot.Countdown.TotalMinutes > visualAlertMin && slot.AlertFlags["30"])

{

IsAlertActive = false;

slot.IsIn30MinAlert = false;

slot.Highlight = false;

slot.ShowSandClock = false;

slot.AlertFlags["30"] = false;

}

slot.CountdownText = string.Format("{0:D2}:{1:D2}",

(int)Math.Floor(slot.Countdown.TotalMinutes),

slot.Countdown.Seconds);

if (firstAlertMin > 0 &&

slot.Countdown.TotalMinutes <= firstAlertMin &&

slot.Countdown.TotalMinutes > (firstAlertMin - 1) &&

!slot.AlertFlags["10"])

{

if (DateTime.Today.DayOfWeek != DayOfWeek.Saturday || Properties.Settings.Default.AlertOnShabbos)

PlayAlert(slot.Id, "10");

slot.AlertFlags["10"] = true;

}

if (secondAlertMin > 0 &&

slot.Countdown.TotalMinutes <= secondAlertMin &&

slot.Countdown.TotalMinutes > (secondAlertMin - 1) &&

!slot.AlertFlags["3"])

{

if (DateTime.Today.DayOfWeek != DayOfWeek.Saturday || Properties.Settings.Default.AlertOnShabbos)

PlayAlert(slot.Id, "3");

slot.AlertFlags["3"] = true;

}

if (\_internalSunriseTime.Date != DateTime.Today)

{

\_hasReloadedForCurrentSunriseCycle = false;

\_currentSunriseForReloadCheck = \_internalSunriseTime;

Logger.LogInfo($"New Gregorian day detected. Excel data reloaded to update current day's times. Sunrise: {\_internalSunriseTime:HH:mm:ss}");

}

DateTime reloadTriggerTime = \_internalSunriseTime.Subtract(TimeSpan.FromMinutes(72));

if (DateTime.Now >= reloadTriggerTime && !\_hasReloadedForCurrentSunriseCycle)

{

Logger.LogInfo($"Triggering scheduled daily Excel reload. Current Time: {DateTime.Now:HH:mm:ss}, Reload Trigger Time: {reloadTriggerTime:HH:mm:ss}");

LoadFromExcel();

\_hasReloadedForCurrentSunriseCycle = true;

\_currentSunriseForReloadCheck = \_internalSunriseTime;

}

}

}

IsAlertNotActive = !IsAlertActive;

UpdateSlotCollections();

OnPropertyChanged(nameof(CurrentTime));

// NEW: Update Sefirat HaOmer status

IsSefiratHaOmer = Properties.Settings.Default.SefiratHaOmerEnabled && IsInSefiratHaOmerPeriod(DateTime.Today);

if (IsSefiratHaOmer)

{

SefiratHaOmerText = GetSefiratHaOmerText(DateTime.Today);

}

});

};

\_timer.Start();

}

private void LoadFromExcel()

{

string path = Properties.Settings.Default.ExcelFilePath;

if (!File.Exists(path))

{

Logger.LogWarning($"Excel file '{path}' not found. Loading mock data.");

LoadMock();

return;

}

try

{

using (var stream = File.Open(path, FileMode.Open, FileAccess.Read))

{

using (var reader = ExcelReaderFactory.CreateReader(stream))

{

var dataSet = reader.AsDataSet(new ExcelDataSetConfiguration()

{

ConfigureDataTable = \_ => new ExcelDataTableConfiguration()

{

UseHeaderRow = true

}

});

var table = dataSet.Tables[0];

if (table == null)

{

Logger.LogWarning("No data tables found in the Excel file. Loading mock data.");

LoadMock();

return;

}

var today = DateTime.Today;

DataRow todayRow = null;

int dateColumnIndex = -1;

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

{

if (table.Columns[i].ColumnName.Equals("Date", StringComparison.OrdinalIgnoreCase))

{

dateColumnIndex = i;

break;

}

}

if (dateColumnIndex == -1)

{

Logger.LogWarning("'Date' column not found in Excel. Loading mock data.");

LoadMock();

return;

}

foreach (DataRow row in table.Rows)

{

if (row[dateColumnIndex] != DBNull.Value && DateTime.TryParse(row[dateColumnIndex].ToString(), out DateTime excelDate))

{

if (excelDate.Date == today.Date)

{

todayRow = row;

break;

}

}

}

if (todayRow == null)

{

Logger.LogWarning($"No entry found for today's date ({today.ToShortDateString()}) in '{path}'. Loading mock data.");

LoadMock();

return;

}

int GetColumnIndex(string columnName)

{

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

{

if (table.Columns[i].ColumnName.Equals(columnName, StringComparison.OrdinalIgnoreCase))

{

return i;

}

}

return -1;

}

DateTime ParseTimeFromCell(DataRow row, string columnName)

{

int colIndex = GetColumnIndex(columnName);

if (colIndex != -1 && row[colIndex] != DBNull.Value)

{

string timeString = row[colIndex].ToString();

if (TimeSpan.TryParse(timeString, out TimeSpan timeSpan))

{

return today.Add(timeSpan);

}

else if (DateTime.TryParse(timeString, out DateTime dateTimeFromCell))

{

return today.Add(dateTimeFromCell.TimeOfDay);

}

}

return DateTime.MinValue;

}

TimeSlots.Clear();

// Modified: Add only EOS1 and EOT1 if UseSingleDeadline is true

if (Properties.Settings.Default.UseSingleDeadline)

{

AddSlot("a2EOS1", ParseTimeFromCell(todayRow, "EOS1"));

AddSlot("b2EOT1", ParseTimeFromCell(todayRow, "EOT1"));

}

else

{

AddSlot("a1EOS2", ParseTimeFromCell(todayRow, "EOS2"));

AddSlot("a2EOS1", ParseTimeFromCell(todayRow, "EOS1"));

AddSlot("b1EOT2", ParseTimeFromCell(todayRow, "EOT2"));

AddSlot("b2EOT1", ParseTimeFromCell(todayRow, "EOT1"));

}

TimeSlots.OrderBy(s => s.Id);

\_internalSunriseTime = ParseTimeFromCell(todayRow, "Sunrise");

\_internalMiddayTime = ParseTimeFromCell(todayRow, "Midday");

\_internalSunsetTime = ParseTimeFromCell(todayRow, "Sunset");

OnPropertyChanged(nameof(Sunrise));

OnPropertyChanged(nameof(Midday));

OnPropertyChanged(nameof(Sunset));

HebrewDate = GetHebrewJewishDateString(today, false);

OnPropertyChanged(nameof(HebrewDate));

if (TimeSlots.Any(s => s.Time == DateTime.MinValue) ||

\_internalSunriseTime == DateTime.MinValue || \_internalMiddayTime == DateTime.MinValue || \_internalSunsetTime == DateTime.MinValue)

{

Logger.LogWarning("Some times could not be parsed from Excel. Using mock data for missing values.");

}

}

}

}

catch (Exception ex)

{

Logger.LogWarning($"An error occurred while reading the Excel file: {ex.Message}\nLoading mock data instead.");

LoadMock();

}

foreach (var slot in TimeSlots)

{

slot.AlertFlags = new Dictionary<string, bool>() { ["30"] = false, ["10"] = false, ["3"] = false };

}

// NEW: Update Sefirat HaOmer status

IsSefiratHaOmer = Properties.Settings.Default.SefiratHaOmerEnabled && IsInSefiratHaOmerPeriod(DateTime.Today);

if (IsSefiratHaOmer)

{

SefiratHaOmerText = GetSefiratHaOmerText(DateTime.Today);

}

}

private void LoadMock()

{

TimeSlots.Clear();

var now = DateTime.Now;

if (Properties.Settings.Default.UseSingleDeadline)

{

AddSlot("a2EOS1", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

AddSlot("b2EOT1", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

}

else

{

AddSlot("a2EOS1", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

AddSlot("a1EOS2", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

AddSlot("b2EOT1", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

AddSlot("b1EOT2", DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture));

}

\_internalSunriseTime = DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture);

\_internalMiddayTime = DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture);

\_internalSunsetTime = DateTime.ParseExact("00:00", "HH:mm", CultureInfo.InvariantCulture);

HebrewDate = GetHebrewJewishDateString(now, false);

// NEW: Update Sefirat HaOmer status for mock data

IsSefiratHaOmer = Properties.Settings.Default.SefiratHaOmerEnabled && IsInSefiratHaOmerPeriod(DateTime.Today);

if (IsSefiratHaOmer)

{

SefiratHaOmerText = GetSefiratHaOmerText(DateTime.Today);

}

OnPropertyChanged(nameof(Sunrise));

OnPropertyChanged(nameof(Midday));

OnPropertyChanged(nameof(Sunset));

OnPropertyChanged(nameof(HebrewDate));

}

private void AddSlot(string id, DateTime time)

{

TimeSlots.Add(new TimeSlot

{

Id = id,

Description = \_translations[\_currentLang][id],

PassedText = \_translations[\_currentLang]["Passed"],

Time = time,

IsPassed = false,

CountdownText = "",

ShowSandClock = false,

Highlight = false,

IsIn30MinAlert = false,

AlertFlags = new Dictionary<string, bool>() { ["30"] = false, ["10"] = false, ["3"] = false }

});

// NEW: Notify Sefirat HaOmer status after adding slots

OnPropertyChanged(nameof(IsSefiratHaOmer));

}

private void PlayAlert(string slotId, string minutesBefore)

{

string fileName = String.Empty;

string extFileName = String.Empty;

if (slotId == "a2EOS1" &&

minutesBefore == Properties.Settings.Default.FirstAlertMinutes.ToString() &&

!string.IsNullOrEmpty(Properties.Settings.Default.EOS1FirstAlertPath))

extFileName = Properties.Settings.Default.EOS1FirstAlertPath;

else if (slotId == "a2EOS1" &&

minutesBefore == Properties.Settings.Default.SecondAlertMinutes.ToString() &&

!string.IsNullOrEmpty(Properties.Settings.Default.EOS1SecondAlertPath))

extFileName = Properties.Settings.Default.EOS1SecondAlertPath;

else if (slotId == "a1EOS2" &&

minutesBefore == Properties.Settings.Default.FirstAlertMinutes.ToString() &&

!string.IsNullOrEmpty(Properties.Settings.Default.EOS2FirstAlertPath))

extFileName = Properties.Settings.Default.EOS2FirstAlertPath;

else if (slotId == "a1EOS2" &&

minutesBefore == Properties.Settings.Default.SecondAlertMinutes.ToString() &&

!string.IsNullOrEmpty(Properties.Settings.Default.EOS2SecondAlertPath))

extFileName = Properties.Settings.Default.EOS2SecondAlertPath;

else

fileName = $"alert{slotId}\_{minutesBefore}.wav";

try

{

SoundPlayer player = null;

if (!string.IsNullOrEmpty(extFileName))

{

player = new SoundPlayer(extFileName);

System.Diagnostics.Debug.WriteLine($"Playing resource from settings");

}

else if (!string.IsNullOrEmpty(fileName))

{

string resourceKey = Path.GetFileNameWithoutExtension(fileName);

Stream stream = Properties.Resources.ResourceManager.GetStream(resourceKey);

if (stream != null)

{

player = new SoundPlayer(stream);

}

System.Diagnostics.Debug.WriteLine($"Playing resource from Resources.resx: {resourceKey}");

}

else

{

System.Diagnostics.Debug.WriteLine($"Resource not found in Resources.resx. and settings not set for {slotId} alert {minutesBefore}");

return;

}

player.Play();

}

catch (Exception ex)

{

System.Diagnostics.Debug.WriteLine($"Error playing embedded sound: {ex.Message}");

}

}

private void UpdateSlotCollections()

{

var alertSlot = TimeSlots.FirstOrDefault(slot => slot.IsIn30MinAlert && !slot.IsPassed);

TopSlots.Clear();

BottomSlots.Clear();

ObservableCollection<TimeSlot> temp = new ObservableCollection<TimeSlot>();

if (alertSlot != null)

{

IsAlertActive = true;

TopSlots.Add(alertSlot);

foreach (var slot in TimeSlots.Where(s => s != alertSlot))

{

temp.Add(slot);

}

foreach (var slot in temp.OrderByDescending(s => s.Time))

{

BottomSlots.Add(slot);

}

}

else

{

IsAlertActive = false;

}

OnPropertyChanged(nameof(TopSlots));

OnPropertyChanged(nameof(BottomSlots));

// NEW: Notify Sefirat HaOmer status

OnPropertyChanged(nameof(IsSefiratHaOmer));

}

internal void StopTimer()

{

if (\_timer != null)

{

\_timer.Stop();

\_timer.Dispose();

\_timer = null;

Logger.LogInfo("Timer stopped and disposed.");

}

}

private string GetHebrewJewishDateString(DateTime anyDate, bool addDayOfWeek)

{

StringBuilder stringBuilder = new StringBuilder();

CultureInfo cultureInfo = CultureInfo.CreateSpecificCulture("he-IL");

cultureInfo.DateTimeFormat.Calendar = new HebrewCalendar();

if (addDayOfWeek)

{

stringBuilder.Append(anyDate.ToString("dddd", cultureInfo) + " ");

}

stringBuilder.Append(anyDate.ToString("dd", cultureInfo) + " ");

stringBuilder.Append(anyDate.ToString("y", cultureInfo) ?? "");

return stringBuilder.ToString();

}

// NEW: Method to check if the date is in Sefirat HaOmer period

private bool IsInSefiratHaOmerPeriod(DateTime date)

{

var hebrewCalendar = new HebrewCalendar();

int hebrewYear = hebrewCalendar.GetYear(date);

int hebrewMonth = hebrewCalendar.GetMonth(date);

int hebrewDay = hebrewCalendar.GetDayOfMonth(date);

DateTime nisan16 = hebrewCalendar.ToDateTime(hebrewYear, 1, 16, 0, 0, 0, 0);

DateTime sivan6 = hebrewCalendar.ToDateTime(hebrewYear, 3, 6, 0, 0, 0, 0);

if (hebrewMonth == 1 && hebrewDay >= 16)

{

return true;

}

else if (hebrewMonth == 2)

{

return true;

}

else if (hebrewMonth == 3 && hebrewDay <= 6)

{

return true;

}

return false;

}

// NEW: Method to generate Sefirat HaOmer text

private string GetSefiratHaOmerText(DateTime date)

{

var hebrewCalendar = new HebrewCalendar();

int hebrewYear = hebrewCalendar.GetYear(date);

DateTime nisan16 = hebrewCalendar.ToDateTime(hebrewYear, 1, 16, 0, 0, 0, 0);

int omerDay = (date.Date - nisan16.Date).Days + 1;

if (omerDay < 1 || omerDay > 49)

{

return "";

}

int weeks = omerDay / 7;

int days = omerDay % 7;

string weeksText = weeks > 0 ? $"{\_hebrewNumberWords[weeks]} שבועות" : "";

string daysText = days > 0 ? $"{\_hebrewNumberWords[days]} ימים" : "";

string conjunction = weeks > 0 && days > 0 ? " ו" : "";

string omerText = $"היום {\_hebrewNumberWords[omerDay]} ימים שהם {weeksText}{conjunction}{daysText} לעומר";

if (weeks == 0)

{

omerText = $"היום {\_hebrewNumberWords[omerDay]} ימים לעומר";

}

if (days == 0 && weeks > 0)

{

omerText = $"היום {\_hebrewNumberWords[omerDay]} ימים שהם {weeksText} לעומר";

}

return omerText;

}

public void SwitchLanguage(string lang)

{

\_currentLang = lang;

foreach (var slot in TimeSlots)

{

if (\_translations[lang].TryGetValue(slot.Id, out var trans))

slot.Description = trans;

}

foreach (var slot in TimeSlots.Where(s => s.IsPassed))

{

}

OnPropertyChanged(nameof(TimeSlots));

}

private void OnPropertyChanged([CallerMemberName] string name = null) =>

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

}

```

#### 5. Views/MainWindow.xaml

- \*\*Purpose\*\*:

- Adjust the `ItemsControl` for a 2x1 grid when `UseSingleDeadline` is true.

- Add a new `Border` for the Sefirat HaOmer display, visible only when `IsSefiratHaOmer` is true.

- \*\*Changes\*\*:

- Lines 104–107: Added a new style for the Sefirat HaOmer rectangle.

- Lines 295–310: Modified the normal `ItemsControl` to use a 2x1 `UniformGrid` when `UseSingleDeadline` is true.

- Lines 311–330: Added a new `Border` for the Sefirat HaOmer display, bound to `IsSefiratHaOmer` and `SefiratHaOmerText`.

```xml

<Window x:Class="EOTReminder.Views.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:vm="clr-namespace:EOTReminder.ViewModels"

xmlns:conv="clr-namespace:EOTReminder.Converters"

xmlns:local="clr-namespace:EOTReminder.Views"

mc:Ignorable="d"

Title="Zmanim Reminder"

Width="950" Height="650"

Background="Transparent"

WindowStartupLocation="CenterScreen"

WindowStyle="None"

WindowState="Maximized"

Topmost="True"

ResizeMode="NoResize">

<Window.Resources>

<BooleanToVisibilityConverter x:Key="BoolToVisibility"/>

<conv:ResourceToImageSourceConverter x:Key="ImageResourceConverter"/>

<Style x:Key="TimeBoxStyle" TargetType="Border">

<Setter Property="Background" Value="#CCFFFFFF"/>

<Setter Property="CornerRadius" Value="15"/>

<Setter Property="Padding" Value="2"/>

<Setter Property="BorderBrush" Value="#999"/>

<Setter Property="BorderThickness" Value="5"/>

<Setter Property="Effect">

<Setter.Value>

<DropShadowEffect Color="Black" BlurRadius="5" ShadowDepth="2" Opacity="0.3"/>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="BigTimeBoxStyle" TargetType="Border" BasedOn="{StaticResource TimeBoxStyle}">

<Setter Property="Padding" Value="20"/>

<Setter Property="BorderThickness" Value="6"/>

<Setter Property="Background" Value="#CCFFF3CD"/>

<Setter Property="Effect">

<Setter.Value>

<DropShadowEffect ShadowDepth="4" Opacity="0.4" BlurRadius="8" Color="#FFD700"/>

</Setter.Value>

</Setter>

</Style>

<!-- NEW: Style for Sefirat HaOmer rectangle -->

<Style x:Key="OmerBoxStyle" TargetType="Border" BasedOn="{StaticResource BigTimeBoxStyle}">

<Setter Property="Background">

<Setter.Value>

<ImageBrush ImageSource="{Binding Converter={StaticResource ImageResourceConverter}, ConverterParameter=haomerBackground}" Stretch="UniformToFill"/>

</Setter.Value>

</Setter>

<Setter Property="Padding" Value="20"/>

<Setter Property="BorderThickness" Value="6"/>

</Style>

<Style x:Key="DescriptionTextStyle" TargetType="TextBlock">

<Setter Property="FontWeight" Value="Normal"/>

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="TextWrapping" Value="Wrap"/>

<Setter Property="Foreground" Value="#333333"/>

</Style>

<Style x:Key="TimeTextStyle" TargetType="TextBlock">

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="Foreground" Value="#333333"/>

</Style>

<Style x:Key="CountdownTextStyle" TargetType="TextBlock">

<Setter Property="FontWeight" Value="Normal"/>

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="Foreground" Value="Red"/>

</Style>

<Style x:Key="StatusTextStyle" TargetType="TextBlock">

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="Foreground" Value="Red"/>

</Style>

</Window.Resources>

<Window.DataContext>

<vm:MainViewModel />

</Window.DataContext>

<Grid>

<Grid.Background>

<ImageBrush ImageSource="{Binding Converter={StaticResource ImageResourceConverter}, ConverterParameter=background}" Stretch="UniformToFill"/>

</Grid.Background>

<Grid.RowDefinitions>

<RowDefinition Height="0.11\*"/>

<RowDefinition Height="0.04\*"/>

<RowDefinition Height="0.73\*"/>

<RowDefinition Height="0.04"/>

<RowDefinition Height="0.12\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.05\*"/>

<ColumnDefinition Width="0.9\*"/>

<ColumnDefinition Width="0.05\*"/>

</Grid.ColumnDefinitions>

<Border Grid.Row="0" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Top"

Width="100" Height="100" Background="Transparent"

MouseLeftButtonDown="HiddenOptionsButton\_MouseLeftButtonDown"

Panel.ZIndex="100">

</Border>

<Border Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="3" Background="#F0F0F0" Padding="5">

<Grid Margin="5,5" HorizontalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.25\*"/>

<ColumnDefinition Width="0.125\*"/>

<ColumnDefinition Width="0.25\*"/>

<ColumnDefinition Width="0.125\*"/>

<ColumnDefinition Width="0.25\*"/>

</Grid.ColumnDefinitions>

<Viewbox Grid.Column="0">

<StackPanel Orientation="Vertical" HorizontalAlignment="Center">

<Viewbox MaxHeight="40">

<TextBlock Text="שקיעה" FontSize="12"/>

</Viewbox>

<Viewbox MaxHeight="50">

<TextBlock Text="{Binding Sunset}" FontSize="12" FontWeight="Bold"/>

</Viewbox>

</StackPanel>

</Viewbox>

<Viewbox Grid.Column="1">

<Image Width="48" Height="48" Margin="10,0" Source="{Binding Converter={StaticResource ImageResourceConverter}, ConverterParameter=clock}" VerticalAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Column="2">

<StackPanel Orientation="Vertical" HorizontalAlignment="Center">

<Viewbox MaxHeight="40">

<TextBlock Text="חצות" FontSize="12"/>

</Viewbox>

<Viewbox MaxHeight="50">

<TextBlock Text="{Binding Midday}" FontSize="12" FontWeight="Bold"/>

</Viewbox>

</StackPanel>

</Viewbox>

<Viewbox Grid.Column="3">

<Image Width="48" Height="48" Margin="10,0" Source="{Binding Converter={StaticResource ImageResourceConverter}, ConverterParameter=clock}" VerticalAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Column="4">

<StackPanel Orientation="Vertical" HorizontalAlignment="Center">

<Viewbox MaxHeight="40">

<TextBlock Text="נץ החמה" FontSize="12"/>

</Viewbox>

<Viewbox MaxHeight="50">

<TextBlock Text="{Binding Sunrise}" FontSize="12" FontWeight="Bold"/>

</Viewbox>

</StackPanel>

</Viewbox>

</Grid>

</Border>

<Grid Grid.Row="2" Grid.Column="1">

<!-- NEW: Sefirat HaOmer display -->

<Border Style="{StaticResource OmerBoxStyle}" Visibility="{Binding IsSefiratHaOmer, Converter={StaticResource BoolToVisibility}}" Margin="10">

<Grid>

<Viewbox>

<TextBlock Text="{Binding SefiratHaOmerText}" FontSize="24" FontWeight="Bold" TextAlignment="Center" TextWrapping="Wrap"/>

</Viewbox>

</Grid>

</Border>

<!-- Normal 2x2 or 2x1 grid - visible only if no alert and not in Sefirat HaOmer -->

<ItemsControl ItemsSource="{Binding TimeSlots}"

Visibility="{Binding IsSefiratHaOmer, Converter={StaticResource BoolToVisibility}, ConverterParameter=False}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns="{Binding Source={x:Static p:Settings.Default}, Path=UseSingleDeadline, Converter={StaticResource BoolToIntConverter}, ConverterParameter='1,2'}" Rows="2"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Border Style="{StaticResource TimeBoxStyle}"

BorderBrush="{Binding Highlight, Converter={StaticResource BoolToBrushConverter}}"

BorderThickness="9" Margin="25">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="0.015\*"/>

<RowDefinition Height="0.21\*"/>

<RowDefinition Height="0.01\*"/>

<RowDefinition Height="0.54\*"/>

<RowDefinition Height="0.01\*"/>

<RowDefinition Height="0.21\*"/>

<RowDefinition Height="0.015\*"/>

</Grid.RowDefinitions>

<Viewbox Grid.Row="1">

<TextBlock Text="{Binding Description}" FontSize="14" FontWeight="Bold" TextAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Row="3" MaxHeight="230">

<TextBlock Text="{Binding Time, StringFormat='HH:mm:ss'}" FontSize="52" TextAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Row="5">

<TextBlock Text="{Binding PassedText}"

Foreground="Red"

FontSize="24"

FontWeight="Bold"

Visibility="{Binding IsPassed, Converter={StaticResource BoolToVisibility}}"

TextAlignment="Center"/>

</Viewbox>

</Grid>

</Border>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

<!-- Alert mode - split top and bottom rows - visible only if alert is active and not in Sefirat HaOmer -->

<Grid Visibility="{Binding IsAlertActive, Converter={StaticResource BoolToVisibility}, ConverterParameter={Binding IsSefiratHaOmer, Converter={StaticResource BoolToVisibility}, ConverterParameter=False}}">

<Grid.RowDefinitions>

<RowDefinition Height="1.5\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<ItemsControl ItemsSource="{Binding TopSlots}" Grid.Row="0" Padding="5">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns="1" Rows="1"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.28\*"/>

<ColumnDefinition Width="0.44\*"/>

<ColumnDefinition Width="0.28\*"/>

</Grid.ColumnDefinitions>

<Viewbox Grid.Column="0" HorizontalAlignment="Right">

<Image Width="180" Height="500"

Source="/Assets/sandclock.png"

Margin="0,0,8,0"

VerticalAlignment="Center"

Visibility="Visible"

Stretch="Uniform"/>

</Viewbox>

<Border Grid.Column="1" Margin="0,0,0,8"

Style="{StaticResource BigTimeBoxStyle}"

BorderBrush="{Binding Highlight, Converter={StaticResource BoolToBrushConverter}}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="0.015\*"/>

<RowDefinition Height="0.15\*"/>

<RowDefinition Height="0.01\*"/>

<RowDefinition Height="0.4\*"/>

<RowDefinition Height="0.01\*"/>

<RowDefinition Height="0.4\*"/>

<RowDefinition Height="0.015\*"/>

</Grid.RowDefinitions>

<Viewbox Grid.Row="1" MaxHeight="72">

<TextBlock Text="{Binding Description}" FontSize="18" FontWeight="Bold" TextAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Row="3" MaxHeight="200">

<TextBlock Text="{Binding Time, StringFormat='HH:mm:ss'}" FontSize="85" TextAlignment="Center"/>

</Viewbox>

<Viewbox Grid.Row="5" MaxHeight="200">

<TextBlock Text="{Binding CountdownText, Mode=OneWay}" FontSize="85" Foreground="Red" TextAlignment="Center"/>

</Viewbox>

</Grid>

</Border>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

<ItemsControl ItemsSource="{Binding BottomSlots}" Grid.Row="1" Padding="2">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<UniformGrid Columns="{Binding Source={x:Static p:Settings.Default}, Path=UseSingleDeadline, Converter={StaticResource BoolToIntConverter}, ConverterParameter='1,3'}" Rows="1"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid>

<Border Margin="5,8,5,8"

Style="{StaticResource TimeBoxStyle}"

BorderBrush="{Binding Highlight, Converter={StaticResource BoolToBrushConverter}}">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height