ЗАТВЕРДЖЕНО
Науково-методичною радою
Державного університету
«Житомирська політехніка»
Протокол №8 від 24 травня 2023 р.

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
для виконання лабораторних робіт з навчальної дисципліни «Людино-машинний інтерфейс» Частина 1

для здобувачів вищої освіти освітнього ступеня «бакалавр»
спеціальності 121 «Інженерія програмного забезпечення»
освітньо-професійна програма «Інженерія програмного забезпечення»
факультет інформаційно-комп'ютерних технологій
кафедра інженерії програмного забезпечення

Рекомендовано
на засіданні кафедри інженерії програмного забезпечення
03 квітня 2023 р.,
протокол № 4

Розробник: Інна СУГОНЯК, Світлана КРАВЧЕНКО,
Євгеній ГРИШКУН

Житомир
2023

Зміст

Вступ

Метою викладання навчальної дисципліни “ Людино-машинний інтерфейс” є надання майбутнім фахівцям знань про сучасні концепції, методи та засоби створення інтерфейсів прикладного програмного забезпечення автоматизованих інформаційних систем на базі використання різноманітних сучасних програмних засобів.

Основними завданнями вивчення дисципліни “Людино-машинний інтерфейс” є формування сукупності знань та вмінь для створення інтерфейсів прикладного програмного забезпечення автоматизованих інформаційних систем на базі використання різноманітних сучасних програмних засобів.

Програма вивчення навчальної дисципліни “Людино-машинний інтерфейс” складена відповідно до освітньо-професійної програми підготовки за спеціальністю 121 «Інженерія програмного забезпечення».

Програма навчальної дисципліни складається з таких змістових модулів:

1. Психологічні принципи людино-машинної взаємодії
2. Функціональні компоненти та властивості людино-машинного інтерфейсу
3. Засоби розробки людино-машинного інтерфейсу
4. Оцінювання якості людино-машинного інтерфейсу

Методичні рекомендації містять 4 лабораторні роботи. Вони можуть бути використані для самостійного освоєння практичної складової курсу «Людино-машинний інтерфейс», під час підготовки до складання підсумкового контролю з дисципліни та як доповнення до лекційних курсів, що викладаються у вищих навчальних закладах.

Лабораторна робота № 1

Тема роботи: Інтерфейс віконного додатку.

Мета роботи: Дослідження основних підходів та технологій реалізації інтерфейсу універсального додатку Windows.

Обладнання: ПК, Visual Studio не нижче v.2019

1. Завдання на лабораторну роботу

Засобами інтегрованого середовища розробки Visual Studio побудувати текстовий редактор. Потрібно реалізувати наступні функції:

1. Створення, відкриття та збереження файлів
2. Копіювання, переміщення тексту.
3. Виділення тексту та знищення тексту.
4. Відміну останньої дії.
5. Стрічку, меню та меню швидкого доступу для управління функціональними компонентами додатку.
6. Персоналізувати параметри проекту (назву додатку, оригінальну піктограму, вікно AboutBox тощо).
7. Виконати індивідуальне завдання за варіантом.
8. Додатково виконати завдання на самостійну роботу.

Індивідуальні завдання

1. Додати Control та діалогове вікно що дозволяє зміну вирівнювання виділеного фрагмента тексту.
2. Додати Control та діалогове вікно що дозволяє зміну відступу від лівого краю першого рядка виділеного фрагмента тексту.
3. Додати пункт Control та діалогове вікно що дозволяє встановити маркери до виділеного фрагмента тексту.
4. Додати пункт меню та кнопку що дозволяє встановити нумерований список до виділеного фрагмента тексту.
5. Додати пункт меню та кнопку що дозволяє зміну відступу від лівого краю виділеного фрагмента тексту.
6. Додати пункт меню та кнопку що дозволяє зміну відступу від правого краю виділеного фрагмента тексту.
7. Додати пункт меню та кнопку що дозволяє зміну інтервалу перед та після фрагменту тексту.
8. Додати пункт меню та кнопку що дозволяє визначити позицію табуляції в тексті.
9. Додати пункт меню що визначає позицію курсору та відображує її у командному рядку.

10. Додати кнопки що дозволяють вставляти часто використані символи грецького алфавіту в текст (α,β ,μ).
11. Реалізувати пошук символів в тексті.
12. Реалізувати заміну символів в тексті.
13. Колір фону.
14. Колір тексту.
15. Границі тексту.

Індивідуальні завдання підвищеного рівня складності

1. Додати кнопку, що визначає параметри форматування фрагменту тексту та відображує її у діалоговому вікні.
2. Реалізувати друк документів з редактору.
3. Реалізувати кнопку відміни форматування для виділеного фрагменту тексту.
4. Реалізувати журнали операцій для відміни та повернення останньої дії.
5. Вставку об’єкта.
6. Багаторівневий список.
7. Колонтитули.

Завдання на самостійну роботу

1. Реалізувати MDI інтерфейс текстового редактора
2. реалізувати системи контекстних меню додатку
3. реалізувати довідникову систему додатку.

2. Хід роботи

Взаємодія користувача з комп’ютером шляхом використання ним додатків для виконання своїх функціональних задач. Фокусом взаємодії в таких випадках є інтерфейс додатку.

Класичний інтерфейс WIMP додатку має наступні елементи:

- *робоча область:* внутрішня частина вікна, призначена для вдображення інформації та виконання завдань користувача;
- *границі:* рамка, що обмежує вікно із чотирьох сторін. Розміри вікна можна змінювати, переміщаючи границю мишею;
- *заголовок:* рядок безпосередньо під верхньою границею вікна, що містить назву вікна;
- *значок системного меню:* кнопка ліворуч у рядку заголовка відкриває меню переміщення й зміни розмірів вікна;
- *рядок горизонтального меню:* розташовується безпосередньо під заголовком, містить пункти меню, забезпечує доступ до команд;
- *панель інструментів:* розташовується під рядком меню, являє собою набір кнопок, забезпечує швидкий доступ до деяких команд;
- *кнопки Згорнути, Розгорнути/Відновити, Закрити* розташовані у верхній правій частині вікна.

Більш сучасна версія інтерфейсу замість горизонтального меню та панелі інструментів використовує елемент керування *Стрічка (Ribbon)*.

До стандартних можливостей будь-якого додатку є:

- Створення, відкриття та збереження вмісту робочої області.
- Попередній перегляд та друк вмісту робочої області;
- Робота з буфером обміну;

До функціональних можливостей текстових редакторів відносяться:

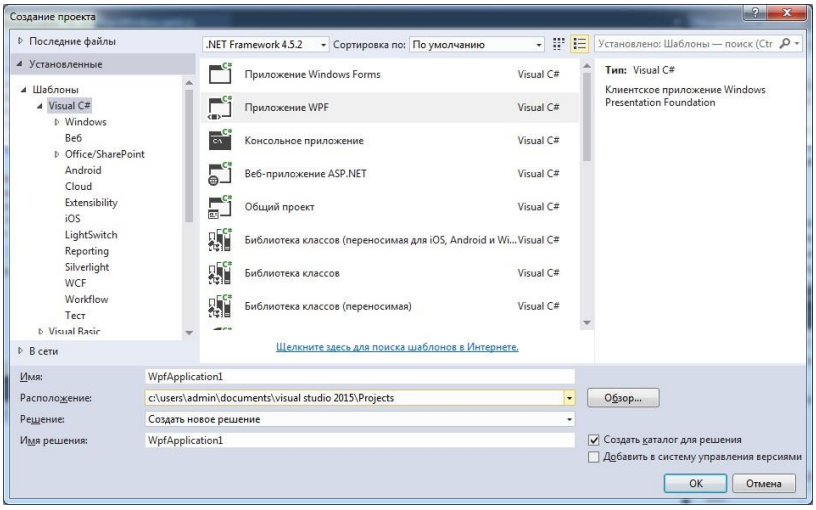
- Керування виглядом текстових символів (керування шрифтами)
- керування розташуванням тексту (форматування абзаців);
- операції пошуку та заміни символів;
- вставка зовнішніх об’єктів (зображень, таблиць, формул тощо)

Будь-який додаток має гнучку систему допомоги що складається із централізованої довідки, підказок та повідомлень в рядку стану.

Додатки поділяються на SPI (однодокументні) та MDI (багатодокументні). Класичний MDI реалізується багатовіконним додатком на основі системи Parent-Child Windows. В більш сучасних в сучасних інтерфейсах використовуються вкладки або технологія псевдоSDI.

Windows Presentation Foundation (WPF [1]) - система для побудови клієнтських додатків для Windows з візуально привабливими можливостями взаємодії з користувачем, графічна (презентаційна) підсистема у складі .NET Framework (починаючи з версії 3.0), яка використовує мову XAML [2].

Для реалізації стандартного додатку необхідно створити новий проект WPF.

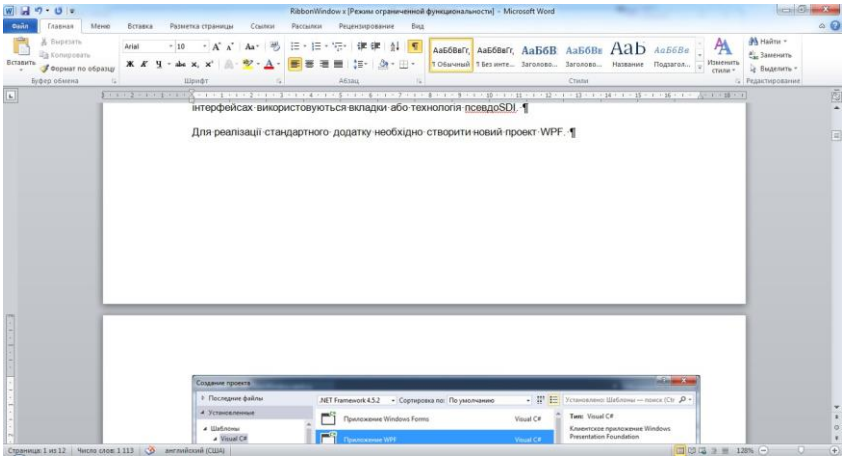


За зразок візьмемо інтерфейс текстового процесора Microsoft Word 2010.

Послідовність створення SDI додатку в нашому випадку наступна:

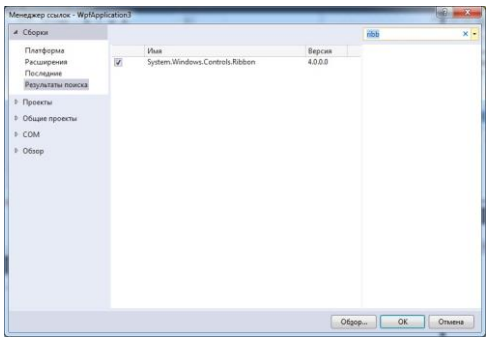
- реалізації системи керування додатком (стрічки);
- реалізація робочої області;

- реалізація функціональних можливостей додатку.

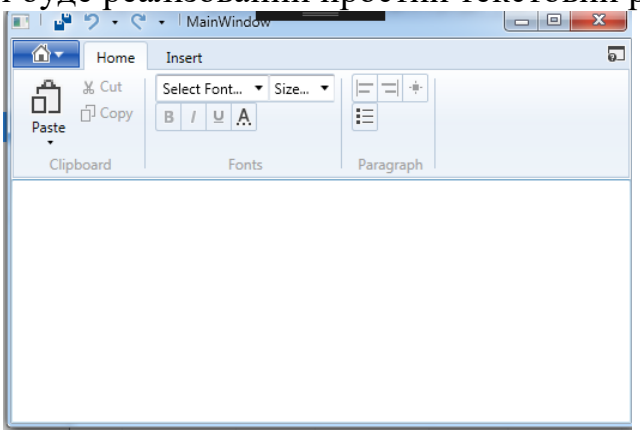


Реалізація стрічки (Ribbon).

Для реалізації системи керування в проект необхідно додати посилання на зборку System.Windows.Controls.Ribbon:



Дизайн інтерфейсу додатку буде виконано із використанням мови XAML. У результаті буде реалізований простий текстовий редактор, типу:



Для цього переходимо у файл з XAML розміткою головного вікна додатку (в даному випадку MainWindow.xaml) та першим кроком робимо заміну типу вікна з Windows на RibbonWindows:

```
<RibbonWindow x:Class="WpfApplication3.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:local="clr-
        namespace:WpfApplication3"
                mc:Ignorable="d"
                Title="MainWindow" Height="350" Width="525">
    <Grid>
    </Grid>
    </RibbonWindow>
```

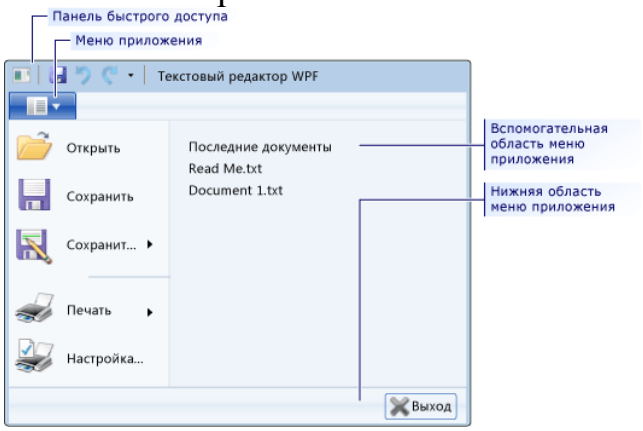
Наступним кроком в межах Grid визначаємо для розміточної сітки 2 рядки – перший для стрічки, другий для робочої області.

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

Далі розташовуємо стрічку в першому рядку:

```
<Ribbon Grid.Row="0" Margin="1,1,1,1">
</Ribbon>
```

Основними елементами стрічки є:



Далі реалізуємо інтерфейс *Довідникового меню*:

```
<Ribbon.HelpPaneContent>
            <RibbonButton SmallImageSource="images\HelpApplication.png" KeyTip="F"/>
        </Ribbon.HelpPaneContent>
```

Для роботи із зображенням кнопки HelpPane та іншими кнопками необхідно виконати наступні підготовчи дії:

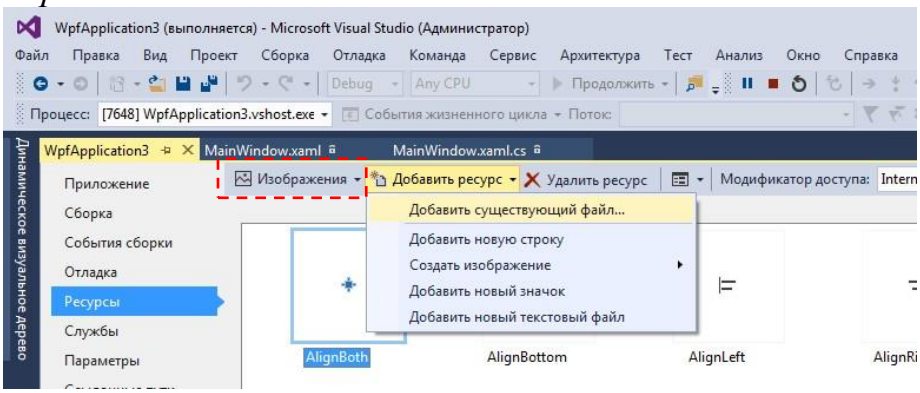
1. Створити в папці проекту папку images.
2. Зберегти в ній піктограми для наступних дій: Create, Open, Close, Save, Propetrty, Print Copy, Paste, Cut, FontColor, Bold, Italic, Underline, AlignLeft, AlignRight, AlignBoth, AlignCenter, BulletList, InsertObject, InsertTable, InsertImage та для індивідуального завдання.

Для виконання даного завдання слід виконати пошук в Інтернеті колекції піктограм або скористатись VSImageLibrary за посиланням https://www.microsoft.com/en-us/download/details.aspx?id=35825 або використати типовий набір кнопок із VSImageLibrary, що використані в даному

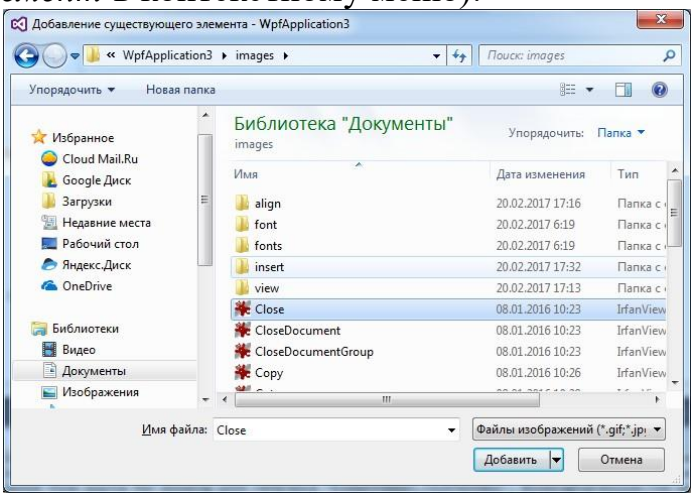
зразку за посиланням

3. Додати проект ресурси зображень

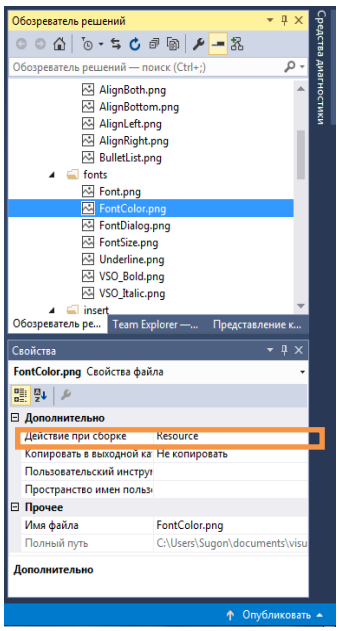
Для цього перейти в меню *Проект->Свойства*: на вкладку *Ресурси* та обрати *Изображения:*



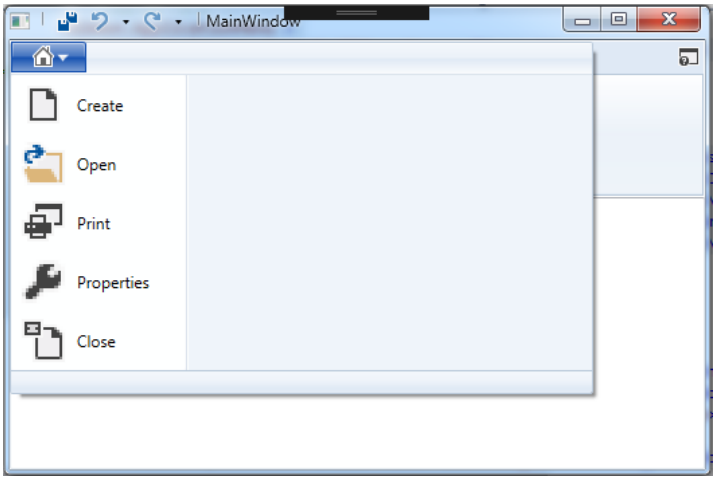
або за допомогою контекстного меню проекта у *Обозреватели решений* створити папку images и добавить зображення (команда *Добавить существующий елемент* в контекстному меню):



Обов’язково перевірити у властивостях всіх рисунків, що параметр *Действие при сборке* має значення Recource.



Наступним кроком є створення *Головного меню* додатку:



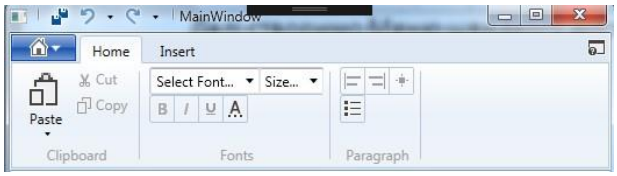

```
<Ribbon.ApplicationMenu >
    <RibbonApplicationMenu SmallImageSource="images\icon-home.png"        >
<RibbonApplicationMenuItem Header="Create" ImageSource="images\Document.png" KeyTip="C" />
<RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png" KeyTip="O" />
<RibbonApplicationMenuItem Header="Print" ImageSource="images\PrintDialog.png" KeyTip="P"/>
    <RibbonApplicationMenuItem Header="Properties" ImageSource="images\Property.png" />
<RibbonApplicationMenuItem      Header="Close"      ImageSource="images\CloseDocument.png"      KeyTip="C"
  Click="btnClose_Click"/>
    </RibbonApplicationMenu>
    </Ribbon.ApplicationMenu>
```

Далі створимо *Меню швидкого доступу*:


```
<Ribbon.QuickAccessToolBar>
        <RibbonQuickAccessToolBar>
            <RibbonButton SmallImageSource="images\SaveAll.png" />
```

```
                <RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" >
                    <RibbonSplitMenuItem Header="Undo1"></RibbonSplitMenuItem>
                </RibbonSplitButton>
                <RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" >
                    <RibbonSplitMenuItem Header="Redo1"></RibbonSplitMenuItem>
                </RibbonSplitButton>
                                </RibbonQuickAccessToolBar>
        </Ribbon.QuickAccessToolBar>
```

Наступним кроком є створення та заповнення вкладки *Home:*



Зверніть увагу, що дана вкладка поділена на три групи: Clipboard, Fonts, Paragraph.

```
<RibbonTab Header="Home">
        <!-- Home/Clipboard group-->
        <RibbonGroup Header="Clipboard">
          <RibbonMenuButton LargeImageSource="Images\paste.png" Label="Paste" KeyTip="V" >
          <RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T" />
<RibbonMenuItem ImageSource="Images\Paste.png" Header="Paste Special..." KeyTip="S"/>
          </RibbonMenuButton>
          <RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X" />
          <RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C" />
        </RibbonGroup>
        <!-- Home/Colors group-->
        <RibbonGroup x:Name="fonts" Header="Fonts" Width="Auto" >
          <RibbonControlGroup>
<ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}" Text="Select Font..."
    IsEditable="True"/>
            <ComboBox x:Name="_fontSize" Text="Size..." IsEditable="True"></ComboBox>
          </RibbonControlGroup>
          <RibbonControlGroup>
            <RibbonButton SmallImageSource="Images\fonts\VSO_Bold.png" KeyTip="B" />
            <RibbonButton SmallImageSource="Images\fonts\VSO_Italic.png" KeyTip="I" />
            <RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U" />
            <RibbonButton SmallImageSource="Images\fonts\FontColor.png" KeyTip="A"/>
          </RibbonControlGroup>
        </RibbonGroup>
        <RibbonGroup x:Name="paragraph" Header="Paragraph">
          <RibbonControlGroup>
            <RibbonButton SmallImageSource="Images\align\AlignLeft.png" />
            <RibbonButton SmallImageSource="Images\align\AlignRight.png" />
            <RibbonButton SmallImageSource="Images\align\AlignBoth.png" />
          </RibbonControlGroup>
          <RibbonControlGroup>
            <RibbonButton SmallImageSource="Images\align\BulletList.png"/>
          </RibbonControlGroup>
        </RibbonGroup>
      </RibbonTab>
```

Для коректного відображення розмірів шрифту у відповідному ComboBox необхідно у файлі коду головного вікна (в даному випадку MainWindow.xaml.cs) визначити структуру даних, приблизно такого вмісту:

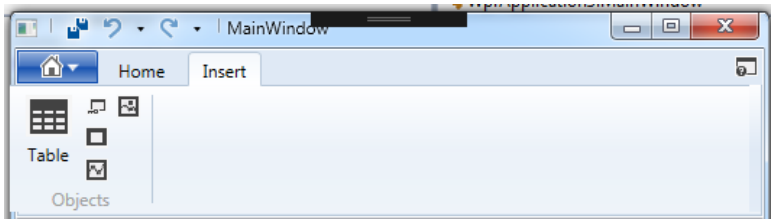
```
public double[] FontSizes
{
```

```
        get
        {
            return new double[] { 3.0, 4.0, 5.0, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0,
9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5,13.0,13.5,14.0, 15.0,16.0, 17.0, 18.0, 19.0,
20.0, 22.0, 24.0, 26.0, 28.0, 30.0,32.0, 34.0, 36.0, 38.0, 40.0, 44.0, 48.0, 52.0, 56.0,
60.0, 64.0, 68.0, 72.0, 76.0,80.0, 88.0, 96.0, 104.0, 112.0, 120.0, 128.0, 136.0, 144.0
};
        }
    }
```

та визначити її як джерело даних для елемента керування ComboBox

```
public MainWindow()
    {
        InitializeComponent();
        _fontSize.ItemsSource = FontSizes;
    }
```

Останнім кроком є створення вкладки Insert:




```
<RibbonTab Header="Insert" Margin="-8,0,8,0" UseLayoutRounding="True"
    ScrollViewer.VerticalScrollBarVisibility="Auto">
        <RibbonGroup x:Name="objects" Header="Objects">
          <RibbonButton LargeImageSource="Images\insert\Table.png" Label="Table"/>
          <RibbonButton SmallImageSource="Images\insert\ApplicationAccess.png"/>
          <RibbonButton SmallImageSource="Images\insert\Rectangle.png"/>
          <RibbonButton SmallImageSource="Images\insert\LineChart.png"/>
          <RibbonButton SmallImageSource="Images\insert\Image.png"/>
        </RibbonGroup>
      </RibbonTab>
```

Компілюємо проект та перевіряємо помилки (сподіваємось, що це вже 3 або 4 раз). Дивимось на результат.

Реалізація робочої області

Для реалізації робочої області використаємо компонет RichTextBox.

```
<RichTextBox x:Name="doc1" Grid.Row="1">
        </RichTextBox>
```

Даний компонент містить вбудований функціонал для підтримки стандартних операцій з текстом, таких як робота з буфером обміну та параметрами шрифту та абзацу.

Для використання вбудованих властивостей компоненту використаємо розширення розмітки x:Static . За його допомоги реалізуємо наступні команди:

- команди рівня додатку: Cut, Copy, Paste, Undo, Rendo.
- команди редагування: Bold, Italic, Underline, AlignLeft, AlignRight, AlignJustify.

Додаємо відповідні дефініції в код розмітки вікна:
в *Меню швидкого доступу*:

```
<RibbonSplitButton x:Name="Undo" SmallImageSource="images\Undo.png" Command="{x:Static ApplicationCommands.Undo}" CommandTarget="{Binding ElementName=_richTextBox}">
...
<RibbonSplitButton x:Name="Redo" SmallImageSource="images\Redo.png" Command="{x:Static ApplicationCommands.Redo}" CommandTarget="{Binding ElementName=_richTextBox}">
```

на вкладку Insert

```
...
    <RibbonMenuItem ImageSource="Images\Paste.png" Header="Keep Text Only" KeyTip="T"
    Command="{x:Static ApplicationCommands.Paste}" CommandTarget="{Binding ElementName=_richTextBox}"/>
...
  <RibbonButton SmallImageSource="Images\Cut.png" Label="Cut" KeyTip="X" Command="{x:Static ApplicationCommands.Cut}" CommandTarget="{Binding ElementName=_richTextBox}"/>
  <RibbonButton SmallImageSource="Images\Copy.png" Label="Copy" KeyTip="C" Command="{x:Static ApplicationCommands.Copy}" CommandTarget="{Binding ElementName=_richTextBox}"/>
```

…

```
<RibbonButton SmallImageSource="Images\fonts\VSO_Bold.png" KeyTip="B" Command="{x:Static EditingCommands.ToggleBold}" CommandTarget="{Binding ElementName=_richTextBox}"/>
  <RibbonButton SmallImageSource="Images\fonts\VSO_Italic.png" KeyTip="I" Command="{x:Static EditingCommands.ToggleItalic}" CommandTarget="{Binding ElementName=_richTextBox}"/>
  <RibbonButton SmallImageSource="Images\fonts\Underline.png" KeyTip="U" Command="{x:Static EditingCommands.ToggleUnderline}" CommandTarget="{Binding ElementName=_richTextBox}"/>
```

…

```
<RibbonButton SmallImageSource="Images\align\AlignLeft.png" Command="{x:Static EditingCommands.AlignLeft}"
    CommandTarget="{Binding ElementName=_richTextBox}"/>
<RibbonButton SmallImageSource="Images\align\AlignRight.png" Command="{x:Static EditingCommands.AlignRight}"
    CommandTarget="{Binding ElementName=_richTextBox}"/>
<RibbonButton SmallImageSource="Images\align\AlignBoth.png" Command="{x:Static EditingCommands.AlignJustify}"
    CommandTarget="{Binding ElementName=_richTextBox}"/>
```

Реалізація основних функцій додатку

Для реалізації функції відкриття файлів визначимо та реалізуємо процедуру btnOpen_Click:

```
<RibbonApplicationMenuItem Header="Open" ImageSource="images\OpenFolder.png" KeyTip="O" Click="btnOpen_Click"/>
```

реалізація має наступний вид:

```
private void btnOpen_Click(object sender, RoutedEventArgs e)
{
  OpenFileDialog dlg = new OpenFileDialog(); dlg.Filter = "Document files (*.rtf)|*.rtf"; var result = dlg.ShowDialog();
  if (result.Value)
  {
  TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd); FileStream file = new FileStream(dlg.FileName, FileMode.Open);
  t.Load(file, System.Windows.DataFormats.Rtf);
  }
```

```
}
```

Для реалізації функції збереження файлів визначимо та реалізуємо процедуру btnSave_Click:

```
<RibbonButton SmallImageSource="images\SaveAll.png" Click="btnSave_Click"/>
```

реалізація має наступний вид:

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

  SaveFileDialog savefile = new SaveFileDialog();
  // set a default file name savefile.FileName = "unknown.doc";
  // set filters - this can be done in properties as well savefile.Filter = "Document files (*.rtf)|*.rtf";
  if (savefile.ShowDialog() == true)
  {
    TextRange t = new TextRange(doc1.Document.ContentStart, doc1.Document.ContentEnd); this.Title = this.Title + " " + savefile.FileName;
    FileStream file = new FileStream(savefile.FileName, FileMode.Create); t.Save(file, System.Windows.DataFormats.Rtf);
    file.Close();
  }

  IsSaved = true;

}
```

Змінна IsSaved визначається як глобальна і в подальшому використовується для контролю збереження файлу перед закриттям додатку.

```
public partial class MainWindow : RibbonWindow
  {
      public bool IsSaved = false;
```

визначення процедури закриття додатку:

```
<RibbonApplicationMenuItem Header="Close" ImageSource="images\CloseDocument.png" KeyTip="C" Click="btnClose_Click"/>
```

реалізація:

```
private void btnClose_Click(object sender, RoutedEventArgs e)
{ if (IsSaved == false)
if (MessageBox.Show("Do you want save changes ?", "Message", MessageBoxButton.YesNo) == MessageBoxResult.Yes)
  //Если была нажата кнопка Yes, вызываем метод Save
  {
    this.btnSave_Click(sender, e);
  }

  this.Close();
}
```

Останнім кроком є реалізація процедури визначення типу та розміру шрифта. Для цього визначимо наступні процедури:

```
<ComboBox ItemsSource="{Binding Source={x:Static Fonts.SystemFontFamilies}}"
SelectionChanged="FontFamili_SelectionChange" Text="Select Font..." IsEditable="True"/>
<ComboBox SelectionChanged="FontSize_SelectionChange" x:Name="_fontSize" Text="Size..."
IsEditable="True"></ComboBox>
```

та реалізуємо їх наступним чином:

```
void ApplyPropertyValueToSelectedText(DependencyProperty formattingProperty, object value)
    {
if (value == null) return;
      doc1.Selection.ApplyPropertyValue(formattingProperty, value);
    }

    private void FontFamili_SelectionChange(object sender, SelectionChangedEventArgs e)
    {
      try
      {
        FontFamily editValue = (FontFamily)e.AddedItems[0];
 ApplyPropertyValueToSelectedText(TextElement.FontFamilyProperty, editValue);
      }
      catch (Exception) { }
    }

    private void FontSize_SelectionChange(object sender, SelectionChangedEventArgs e)
    {

      try
      {


        ApplyPropertyValueToSelectedText(TextElement.FontSizeProperty, e.AddedItems[0]);
      }
      catch (Exception) { }
    }
  }
  }
```

Компілюємо, дивимось на результат та переходимо до виконання індивідуального завдання.

Контрольні питання

1. Людино-машинна(комп'ютерна) взаємодія

а) вивчає методи розробки, оцінки та впровадження інтерактивних комп'ютерних систем, призначених для використання людиною;

б) вивчає методи прийняття рішень при цьому переважно розглядаються формалізовані задачі;

в) займається дослідженням процесів збереження, накопичення, перетворення, передачі даних та інформації із застосуванням комп'ютерної техніки;

г) вивчає системи зі зворотнім зв'язком і аспект керування інформацією в цих системах, розглядаючи при цьому строго формалізовані задачі.

2. Задачі людино-комп'ютерної взаємодії:

а) методологією і розвитком проектування інтерфейсів (тобто, виходячи з вимог і класу користувачів, проектування найкращого інтерфейсу в заданих рамках, оптимізація під необхідні властивості, такі як здатність до навчання і ефективність використання);

б) методами реалізації інтерфейсів (наприклад, програмні інструментарії, бібліотеки та раціональні алгоритми);

в) методами для оцінки та порівняння таких інтерфейсів;

г) розробкою нових інтерфейсів і технологій взаємодії;

д) розвитком описових і прогнозованих моделей, і теорією взаємодії.

3. Вузол взаємодії включає в себе кілька аспектів:

а) Область завдань;

б) Область машини;

в) Області інтерфейсу;

г) Вхідний потік;

д) Вихідний потік;

е) Зворотній зв'язок;

ж) Геометрічні параметри;

з) Фонове оформлення;

и) Тематичний напрямок;

к) Довідникова система.

4. Область завдань:

а) умови і цілі, орієнтовані на користувача;

б) середовизе з яким взаємодіє комп'ютер;

в) непересічні області, що стосуються процесів людини і комп'ютера, не відносяться до сфери взаємодії;

г) потік інформації, який починається в області завдань, коли користувач має кілька завдань, які вимагають використання комп'ютера;

д) потік інформації, який виникає в машині;

е) вузли взаємодії, що проходять через інтерфейс, оцінюються, модеруються та підтверджуються.

5. Типи діалогів. поставте відповідність

а) Структура діалогу типу «питання-відповідь» (Q & A);

б) заснована на аналогії зі звичайним інтерв'ю. Система бере на себе роль інтерв'юера і отримує інформацію від користувача у вигляді відповідей на питання;

в) Діалог на основі меню;

г) є найбільш природним механізмом для роботи з пристроями вказівки і вибору: складається із зображень тих об'єктів, які вибираються користувачем;

д) Діалог на основі екранних форм;

е) допускає обробку на одному кроці кількох відповідей. Використовуються в основному там, де облік будь-якої діяльності вимагає введення досить стандартного набору даних;

ж) діалог на основі командного мови;

з) програмна система не виводить нічого, крім постійної підказки (запрошення на введення команди), яка означає готовність системи до роботи. Кожну команду вводять з нового рядка і зазвичай закінчують натисканням клавіші «введення».

Лабораторна робота № 2

Тема роботи: Класифікація інтерфейсу.

Мета роботи: аналіз ефективності інтерфейсів із застосуванням закону Фітса.

Обладнання: ПК, Visual Studio не нижче v.2019 (.Net Framework 4.5)

1. Завдання на лабораторну роботу

1) створення алгоритму і програми для дослідження особливостей застосування закону Фітса в дизайні інтерфейсів;

2) проведення експериментів по дослідженню швидкості і точності фізичних дій користувача;

3) побудова і інтерполяція графіків залежності тривалості фізичних дій користувача і кількості помилок від параметрів інтерфейсу засобами Microsoft Excel.

Індивідуальні завдання

У всіх варіантах необхідно в середовищі візуального програмування розробити програми для рішення приведених в них задач.

Пропонуються наступні групи задач в різних варіантах:

1. Визначення залежності часу досягнення об'єкту від його розміру і дистанції до нього.

2. Визначення залежності числа помилок, пов'язаних з промахами при досягненні (за обмежений час) дрібних об'єктів від розміру об'єкта і дистанції до нього.

3. Визначення впливу різних сполучень кольорів фону і об'єкта (розмір, що сприймається суб'єктивно) на швидкість його досягнення.

4. Порівняльний аналіз нескінченних і звичайних кнопок по швидкості їх досягнення.

Перша група задач має на увазі три серії експериментів:

1. При фіксованому розмірі D, заданому в табл. 1, визначити вплив дистанції S на час досягнення об'єкта. Для дистанцій S = 0, 20, 40, 60, 100, 150, 200, 250, 300, 350 (в пікселях) необхідно провести по 5 експериментів (натискань) і визначити середній час досягнення об'єкта при кожній дистанції.

2. При фіксованій дистанції S (див. табл. 1) визначити вплив розміру D на час досягнення об'єкта. Для розмірів D = 8, 10, 12, 15, 20, 30, 50, 70, 100 (в пікселях) необхідно провести по 5 експериментів (натискань) і визначити середній час досягнень об'єкта при кожному розмірі.

3. При змінюючому розмірі D і дистанції S (2 вкладених цикла), які задані в попередніх серіях експериментів, визначити для кожного відношення S/D середній час досягнення об'єкта по 3 натисканням.

Таблиця 1

Варіанти завдань для визначення залежності часу досягнення об'єкта від його розміру і дистанції до нього

№ вар.	Об'єкт	Розмір (D), пікс.	Дистанція (S), пікс.	Колір об'єкта	Колір фону
1	Прямокутник D x 2D	7	300	чорний	білий
2	Прямокутник D x 2,5D	8	350	сірий	білий
3	Прямокутник D x 3D	9	400	синій	білий
4	Кнопка D x 2D	10	500	сірий	синій
5	Кнопка D x 3D	12	550	сірий	чорний
6	Кнопка D x 4D	15	600	сірий	синій
7	Квадрат	8	200	чорний	синій
8	Квадрат	10	250	сірий	синій
9	Квадрат	12	300	жовтий	синій
10	Квадрат	15	350	сірий	жовтий
11	Квадрат	17	400	чорний	жовтий
12	Круг	8	300	червоний	синій
13	Круг	10	400	червоний	білий
14	Круг	12	500	синій	білий
15	Круг	15	600	сірий	чорний
16	Круг	17	650	жовтий	синій
17	Еліпс D x 2D	9	500	жовтий	сірий
18	Еліпс D x 2,5D	11	400	червоний	сірий
19	Еліпс D x 3D	12	300	синій	сірий
20	Еліпс D x 3,5D	14	200	білий	зелений
21	Еліпс D x 4D	16	150	сірий	синій

На основі отриманих середніх значень часу досягнення об'єктів в MS Excel повинні бути побудовані 3 графіка: т (S), т (D), т (S / D). До кожного графіку необхідно побудувати апроксимуючу криву - логарифмічну лінію тренда - і вивести рівняння графіка функції.

У другій групі завдань необхідно провести 2 досліди, час на виконання яких обмежений:

1. При D фіксованому розмірі, заданому в табл. 2, визначити вплив дистанції S на число помилок досягнення об'єкта. Для дистанцій S = 0, 20, 40, 60, 100, 150, 200, 250, 300, 350 (в пікселях) необхідно провести по 8 експериментів (успішних натискань) і визначити сумарне число помилкових натискань досягнення об'єкта для кожної дистанції.

2. При S фіксованою дистанції (див. табл. 2) визначити вплив розміру D на число помилок досягнення об'єкта. Для розмірів D = 3, 4, 5, 6, 7, 8, 10, 12, 14 (в

пікселях) необхідно провести по 8 експериментів (успішних натискань) і визначити сумарне число помилкових натискань досягнення об'єкта для кожного розміру.

Таблиця 2

Варіанти завдань для визначення залежності числа помилок при досягненні об'єктів від їх розмірів і дистанції до них

№ вар.	Об'єкт	Час, (t), мс	Розмір (D), пікс.	Дистанція (S), пікс.	Колір об'єкта	Колір фону
1.	Прямокутник D x 2D	1200	4	100	чорний	синій
2.	Прямокутник D x 3D	1500	3	200	жовтий	синій
3.	Кнопка D x 2D	1800	3	250	сірий	зелений
4.	Кнопка D x 3D	1300	4	300	сірий	жовтий
5.	Квадрат	1400	4	400	синій	білий
6.	Квадрат	1500	5	550	жовтий	сірий
7.	Квадрат	1800	3	600	червоний	сірий
8.	Круг	1000	5	100	синій	сірий
9.	Круг	1500	4	150	білий	зелений
10.	Круг	1600	4	250	чорний	білий
11.	Еліпс D x 2D	1200	5	300	сірий	білий
12.	Еліпс D x 2,5D	1500	4	350	синій	білий
13.	Еліпс D x 3D	1800	3	400	чорний	сірий

На основі отриманих значень числа помилок при досягнені об'єкта в MS Excel повинні бути побудовані 2 графіка: N (S), N (D), де N - число помилок. Для кожного графіка необхідно побудувати апроксимуючу криву - логарифмічну лінію тренда.

У третій групі завдань необхідно провести 2 досліди:

1. При D фіксованому розмірі, S дистанції і кольорі об'єкта, заданому в табл. 3, визначити вплив кольору фону на час досягнення об'єкта. Для кожного з 15 кольорів стандартної 16-кольоровий палітри (виключаючи колір об'єкта) необхідно провести по 10 експериментів (натискань) і визначити середній час досягнення об'єкта при кожному кольорі фону.

2. При D фіксованому розмірі, S дистанції і кольорі фону (див. табл. 3) визначити вплив кольору об'єкта на час його досягнення. Для кожного з 15 кольорів стандартної 16-колірної гами (виключаючи колір фону) необхідно провести по 10 (експериментів) натискань і визначити середній час досягнення об'єкта при кожному кольорі фону.

Таблиця 3

Варіанти завдань для визначення залежності швидкості роботи від кольору фону і об'єкта

№ вар.	Об'єкт	Час, (t), мс	Розмір (D), пікс.	Колір об'єкта	Колір фону
1.	Прямокутник D x 2D	4	150	сірий	синій
2.	Прямокутник D x 3D	3	200	жовтий	синій
3.	Квадрат	5	450	сірий	чорний
4.	Квадрат	4	500	жовтий	синій
5.	Квадрат	3	600	червоний	сірий
6.	Круг	5	100	синій	сірий
7.	Круг	4	250	чорний	білий
8.	Еліпс D x 2,5D	4	350	зелений	білий
9.	Еліпс D x 3D	3	400	чорний	сірий
10.	Еліпс D x 4D	4	500	сірий	сірий

На основі отриманих значень в MS Excel повинні бути побудовані 2 стовпчастих: t(СФ), t(Со), де СФ - колір фону, Со - колір об'єкта.

Всім дослідам і варіантам відповідають наступні вимоги і рекомендації:
1. Типовий алгоритм роботи програми повинен бути наступним: а) вибрати номер досвіду з меню; б) намалювати об'єкт встановити курсор в задані позиції; в) по об'єкту клацнути, відраховуючи час від моменту початку руху; г) зберегти проміжні результати; д) перемалювати об'єкт в новому місці вікна, встановити курсор в початкову позицію і т.д.

2. Методи підвищення доступності кнопки

Із закону Фітса можна зробити висновок, що кращий спосіб підвищити доступність кнопки полягає у тому, щоб робити її великою і розташовувати ближче до курсора.

У цього правила є два слідства. Щоб «нескінченно» прискорити натискання кнопки, її, по-перше, можна зробити нескінченного розміру і, по-друге, дистанцію до неї можна зробити нульовою.

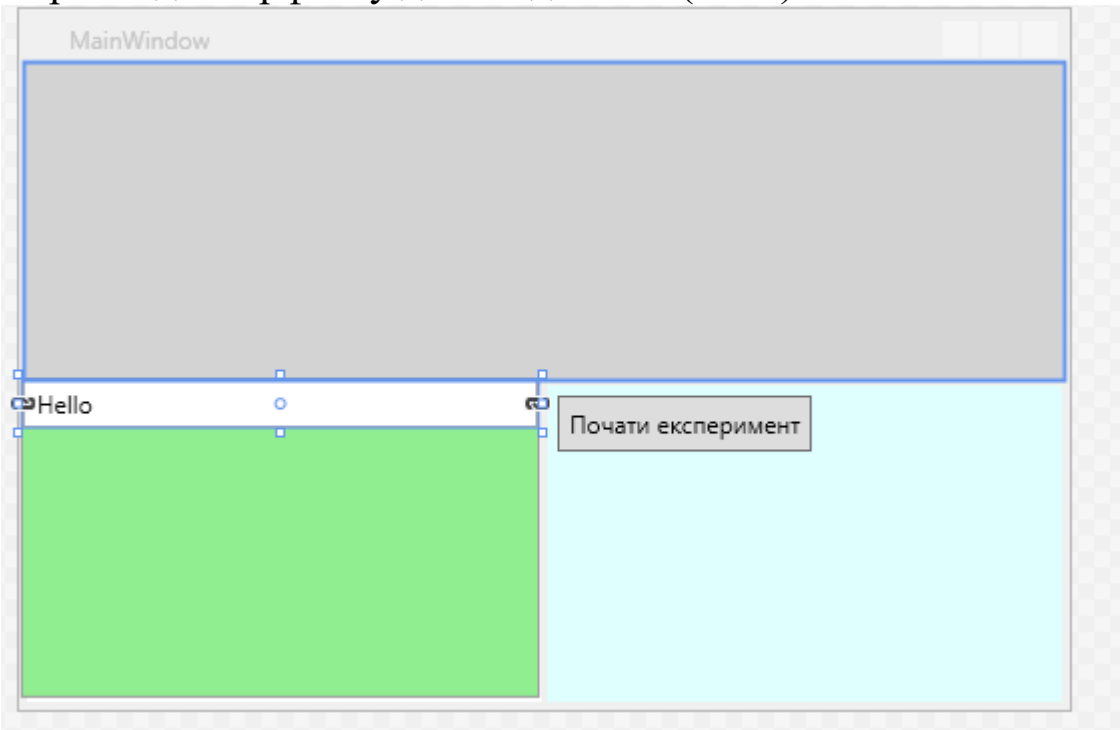
Варіант 1. Кнопка нескінченного розміру. При підведенні курсору до краю екрану він зупиняється, навіть якщо рух миші триває. Це означає, що кнопка, росташована впритул до верхнього або нижнього краю екрану, має нескінченну висоту (так само як кнопка біля лівого або правого краю має нескінченну ширину). Таким чином, швидкість досягнення такої кнопки залежить тільки від відстані до неї і точності вибору початкового напрямку руху.

Отже, кнопка, розташована в кутку екрану (наприклад, кнопка виклику головного меню Windows, «Пуск»), має ще більшу перевагу. Для досягнення такої кнопки від користувача потрібно всього лише перемістити мишу в потрібному напрямку, не піклуючись про її швидкості і не роблячи спроб зупинити її в потрібному місці.

Варіант 2. Нульова дистанція до кнопки. Контекстне меню, яке викликається після натискання правої кнопки миші, завжди відкривається під курсором, відповідно відстань до будь-якого його елемента завжди мінімально. Тому контекстне меню є одним з найшвидших і ефективних елементів управління.

Зменшувати відстані до цілі можна і в діалогових вікнах, які також є контекстно-залежними. За замовчуванням вони відкриваються в центрі екрану, але виходячи з принципу скорочення відстані до їх кнопок, відкривати їх під курсором набагато ефективніше (якщо вони не будуть перекривати важливу інформацію на екрані).

Приклад інтерфейсу для завдання 1 (WPF)



Хід роботи

XAML-розмітка компонентів

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:WpfApplication4"

        Title="MainWindow" Height="350" Width="525">
    <Grid ShowGridLines="False">
        <Grid.RowDefinitions>
            <RowDefinition Height ="*"/>
            <RowDefinition Height ="Auto"/>
            <RowDefinition Height ="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="*"/>

        </Grid.ColumnDefinitions>
        <GridSplitter Grid.Column="1"  Grid.Row="3" ShowsPreview="False" Width="3"
                    HorizontalAlignment="Center" VerticalAlignment="Stretch" />
        <GridSplitter x:Name="split" Grid.Row="1" Grid.ColumnSpan="3" Height="3"
                    HorizontalAlignment="Stretch" VerticalAlignment="Center" />

        <InkCanvas Grid.ColumnSpan="3" Grid.Row="0" Background="LightGray" x:Name="icDrow"
MinHeight="0" MinWidth="0" MouseLeftButtonUp="icUp"/>

        <StackPanel x:Name="Charts" Grid.Column="0" Grid.Row="1" Background="LightGreen"
RenderTransformOrigin="0.593,0.386" Margin="-2,0,2,3" Grid.RowSpan="2">
            <ListBox x:Name="lbRez">
                <ListBoxItem Content="Hello"></ListBoxItem>
            </ListBox>



        </StackPanel>
        <Canvas Grid.Column="2" Grid.Row="2" Background="#dfffff">
            <Button  x:Name="button1"  HorizontalAlignment="Stretch" Content="Почати
експеримент" Click="ButtonStart_Click" Padding="5" Margin="5"  />
        </Canvas>
    </Grid>
</Window>
```

Реалізація основних функцій
Перевірити підключені ресурси:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Timers;
```

Змінні, які потрібні:

```
#region peremennie

        Random x = new Random();
        Random y = new Random();
        public Point p = new Point();
        public Point[] pi = new Point[2];
        int Status_exp = 0;
        DateTime Start;
        DateTime Stoped;
        TimeSpan Elapsed = new TimeSpan();
        public Point s;
        #endregion
```

Початок експерименту – натискання кнопки:

```
  private void ButtonStart_Click(object sender,  RoutedEventArgs e)
        {
            lbRez.Items.Clear();
            lbRez.Items.Add("Результати");
            expir();

        }
```

Промальовування фігур - тут маленькі червоні квадратики:

```
  private Point DrowObject(Point p) {
            Rectangle el = new Rectangle();
            el.Width = 10;
            el.Height = 10;
            p.X=x.Next(Convert.ToInt32(this.Width - el.Width));
            p.Y=y.Next(Convert.ToInt32(icDrow.ActualHeight - el.Height));
            el.Fill = Brushes.Red;
            InkCanvas.SetLeft(el, p.X);
            InkCanvas.SetTop(el, p.Y);
            icDrow.Children.Add(el);
            return p;

        }
```

Проведення 1 вимірювання - для постійних розмірів і згенерованих координат:

```
            public void expir() {
                   int[] rez_arr = new int[2];
                   Array.Clear(pi,0,pi.Length-1);
                   Start= new DateTime(0);
                   icDrow.Children.Clear();
                   icDrow.Strokes.Clear();
                   for (int i = 0; i < 2; i++)
                   {
                      pi[i]= DrowObject(p);
                   }
                   Status_exp = Status_exp+1;
                         }
```

Оброблювач подій миші - вимір часу + перехід до наступного виміру:

```
        private void icUp(object sender, MouseButtonEventArgs e)
        {
            if (Status_exp >0)
            {Point[]n= Array.FindAll(pi, element => (Math.Abs(element.X -
e.GetPosition(this).X) < 20) && (Math.Abs(element.Y - e.GetPosition(this).Y) < 20));
                if (Array.Exists(pi, element => (Math.Abs(element.X - e.GetPosition(this).X)
< 20) && (Math.Abs(element.Y - e.GetPosition(this).Y) < 20)))
                {
                    if (!((Math.Abs(s.X - e.GetPosition(this).X) < 10) && (Math.Abs(s.Y -
e.GetPosition(this).Y) < 10)))
                    {
                        s = e.GetPosition(this);

                        {
                            if (Start.Ticks == 0) { Start = DateTime.Now; }
                            else
                            {
                                Stoped = DateTime.Now;
                                Elapsed = Stoped.Subtract(Start);
                                long elapsedTicks = Stoped.Ticks - Start.Ticks;
                                TimeSpan elapsedSpan = new TimeSpan(elapsedTicks);
                                double rez = Math.Sqrt(Math.Pow(pi[0].X - pi[1].X, 2) +
Math.Pow(pi[0].Y - pi[1].Y, 2));
                                lbRez.Items.Add("Експеримент"+ Status_exp.ToString()+"
Час:"+elapsedSpan.Milliseconds.ToString()+ " Відстань "+rez);

                                if (Status_exp<100) expir();

                            }

                        }


                    }
                }
            }
        }
```

Контрольні питання

1. Інтерфейс складається:

а) набір завдань користувача, які він вирішує за допомогою системи;
б) використовувана системою метафора (наприклад, робочий стіл в MS Windows тощо);
в) елементи управління системою;
г) навігація між блоками системи;
д) візуальний (і не тільки) дизайн екранів програми.

2. Види інтерфейсів:

а) командний інтерфейс;

б) графічний користувальницький інтерфейс;

в) SILK-інтерфейс;

г) семантичний інтерфейс;

д) об'єктно-орієнтований інтерфейс;

е) з віддаленим керуванням.

3. SILK-інтерфейс

а) користувач дає команди комп'ютеру, який їх виконує і видає результат користувачеві;

б) даний підхід полягає в символічному зображенні доступних дій у вигляді ікон (icons) на екрані і надання користувачеві можливості вибирати дії за допомогою миші або іншого координатного пристрою введення;

в) Інтерфейс найбільш наближений до звичайної, людської формі спілкування. При цьому комп'ютер визначає команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд комп'ютер перетворює в зрозумілу людині форму.

4. Графічний користувальницький інтерфейс

а) користувач дає команди комп'ютеру, який їх виконує і видає результат користувачеві

б) даний підхід полягає в символічному зображенні доступних дій у вигляді ікон (icons) на екрані і надання користувачеві можливості вибирати дії за допомогою миші або іншого координатного пристрою введення *

в) Інтерфейс найбільш наближений до звичайної, людської формі спілкування. При цьому комп'ютер визначає команди, аналізуючи людську мову і знаходячи в ній ключові фрази. Результат виконання команд комп'ютер перетворює в зрозумілу людині форму

Лабораторна робота № 3

Тема роботи: Створення макету сайта
Мета роботи: Розробка макету сайту з використанням Figma
Обладнання: ПК, Figma

1. Завдання на лабораторну роботу
Завдання 1. В лабораторній роботі повинно бути спроектованно макет сайту. Наприклад, Figma. Щоб створити свій макет, заререєстуємось (Sign up).
Як альтернативу, можемо використати Gimp, Adobe XD, Adobe Photoshop.
2. Створення шаблону
Створемо 2 фрейма (Desktop)

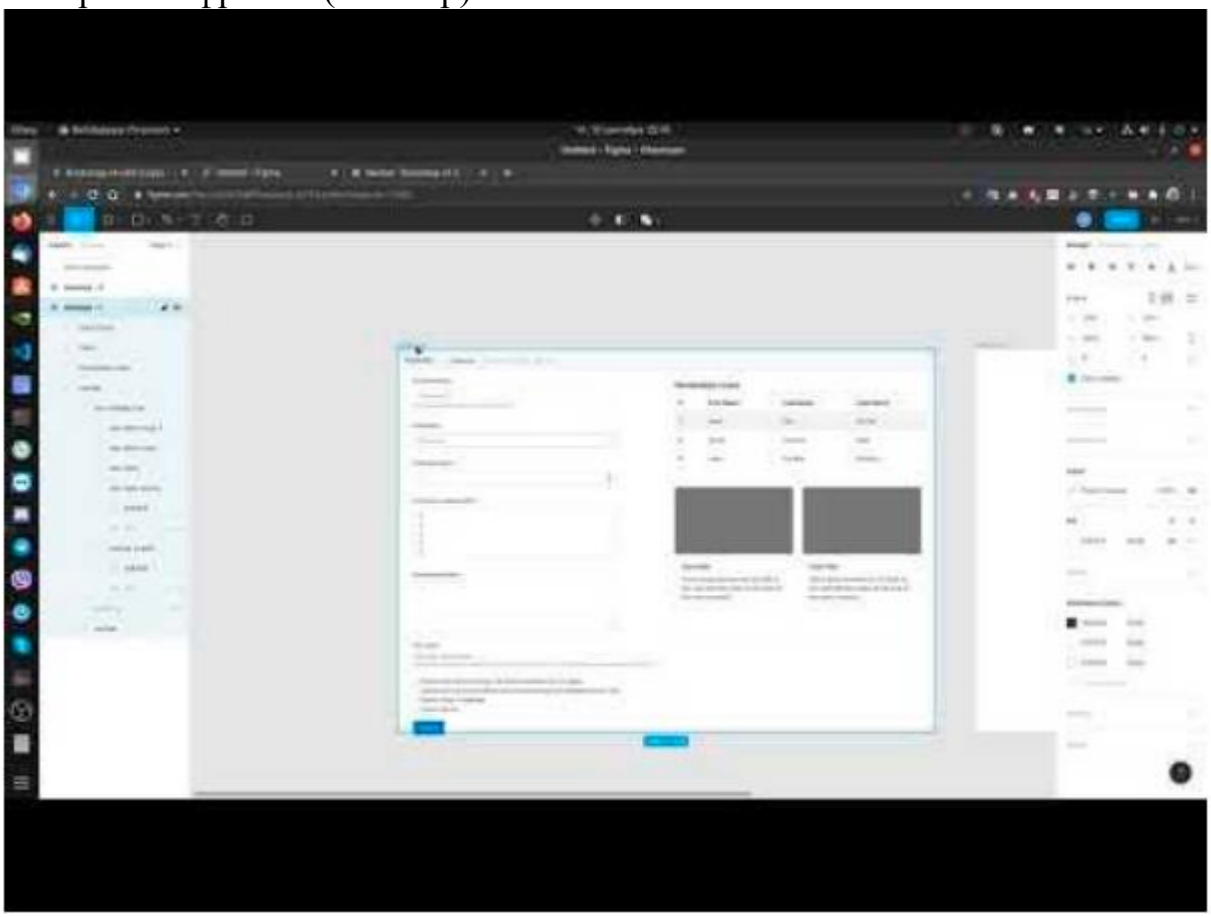
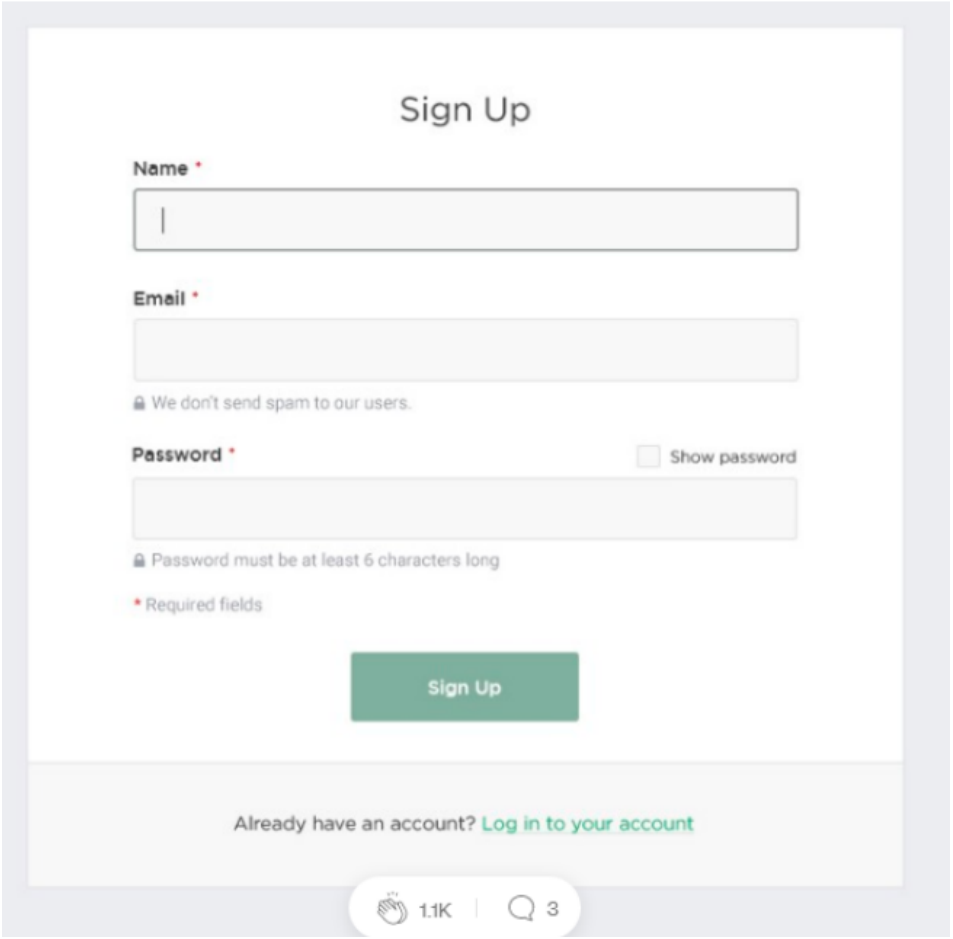


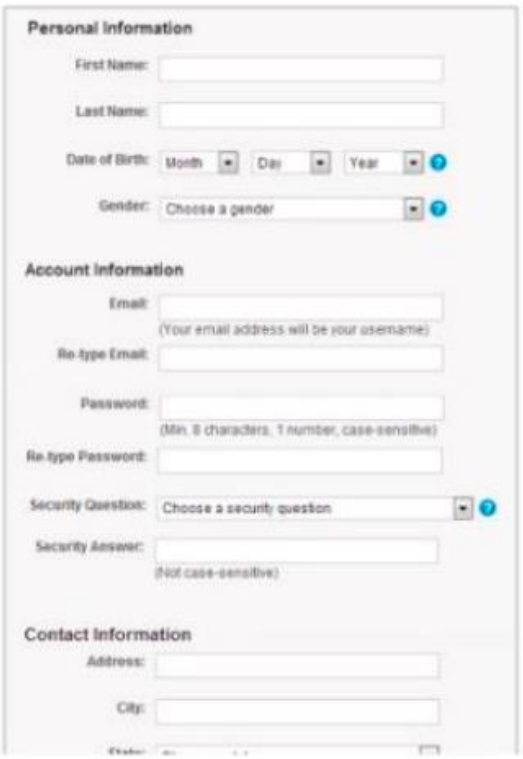
Рисунок 2. Фрейм

3. **Створення меню**
4. Створення форми
Вимоги до макету:
- Повинна бути стилізована головна сторінка;
- Повинна бути стилізовна як мінімум одна сторінка з формою (наприклад, форма замовлення, або створення чи редагування);
- Повинна бути стилізована як мінімум одна сторінка зі списком (список клієнтів, товарів, розходів тощо)
- Повинен бути розміщений логототип;

Також рекомендується ознакомиться з бібліотекою для побудови сайтів Bootstrap 4.

Елементи інтерфейсу з цієї бібліотеки для Figma можна забрати до себе звідси.





Don't.

Do.

Індивідуальні завдання

1. Портал медичного закладу
2. Книжний інтернет-магазин.
3. Автосервіс.
4. Електроний щоденник школяра.
5. Електроний журнал в ЗВО.
6. Облік продукції на складі.
7. Домашній бюджет.
8. Готель.
9. Облік обчислювальної техніки.
10. Суші-бар.
11. Будівельна фірма.
12. Фірма по ремонту комп’ютерної техніки.
13. Бібліотека.
14. Відділ кадрів.
15. Дошка об’яв.
16. Фотогалерея.
17. Форум.
18. Блог новин.
19. Облік стану здоров’я співробітників фірми.
20. Комп’ютерна гра.

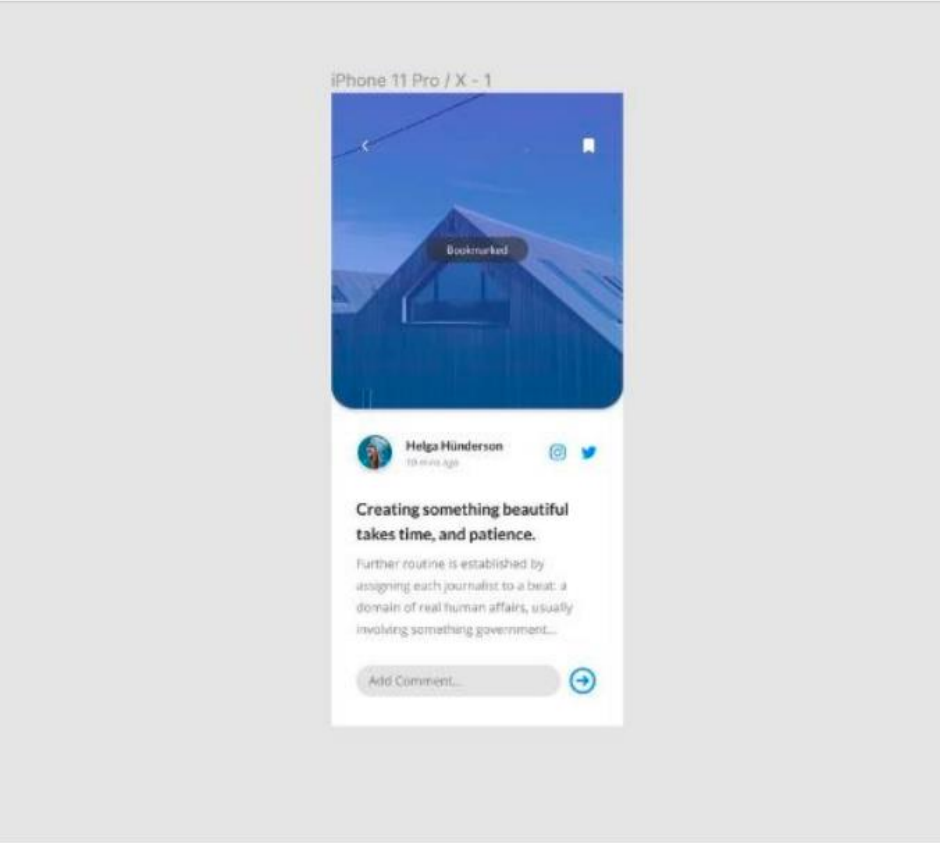
Контрольні запитання

1. Різниця між front-end і back-end. Основи front-end розробки
2. Основи UX-дизайну та UX/UI-дизайну

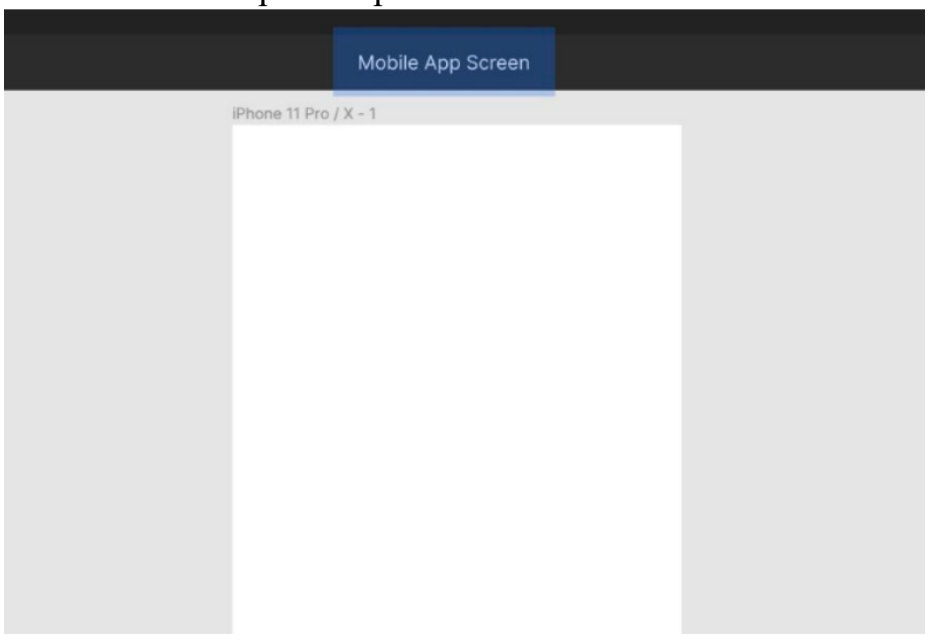
3. TOP-10 навичок UX-дизайнера та UI-дизайнера
4. Закони UX-дизайну
5. Тренди UX-дизайну
6. Поширені помилки UX/UI-дизайну

Лабораторна робота № 4

Тема роботи: Проєктування інтерфейсу мобільного додатку
Мета роботи: Створення інтерфейсу мобільного додатку
Обладнання: ПК, Figma
Як зразок взяти наступний інтерфейс мобільного додатку:



Вибрати варіант iPhone 11 Pro / X

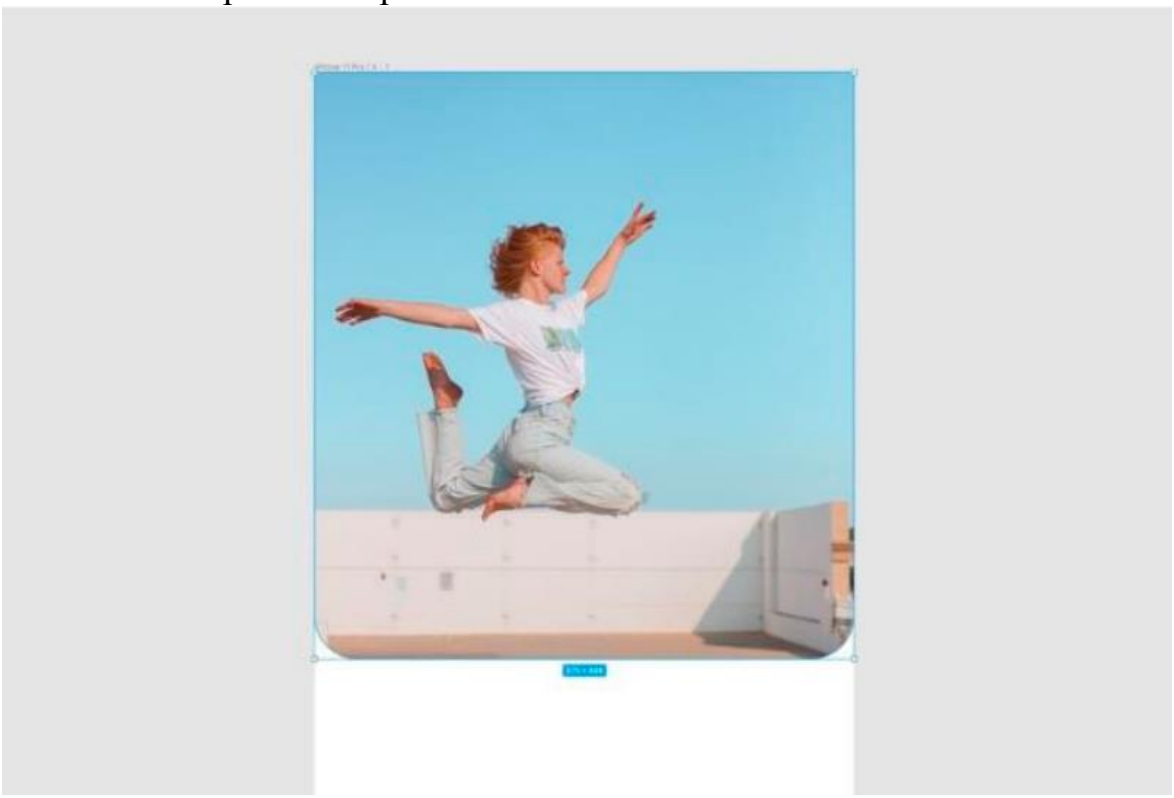


Мініатюра картки

Вибираємо інструмент «прямокутик» (R), малюємо фігуру приблизно 375px в ширину і 406px в висоту розміщуємо її в верхній частині фрейма.

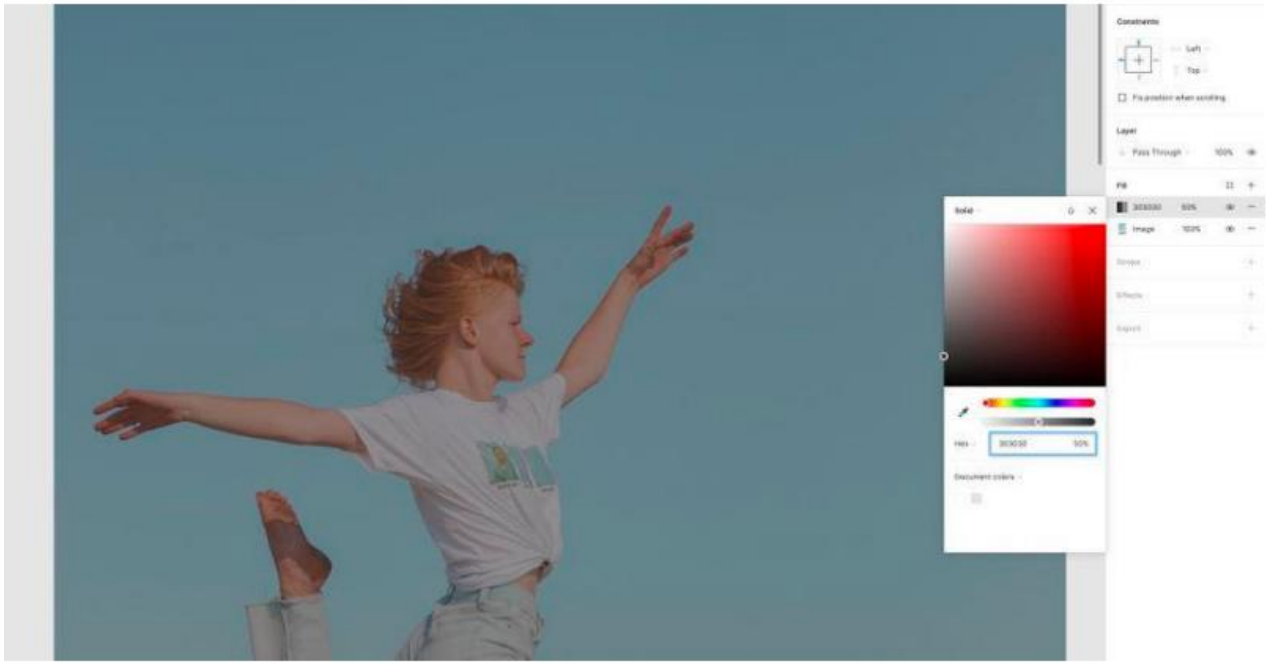
Перевіряємо чи встановлено значення 0 по осям X і Y на панелі властивостей. На панелі властивостей вибираємо по конці Independent Corners выберите 30 для радіуів зкруглення лівого нижнього і правого нижнього кітів прямомокутника.

Зображення розмістимо всередині нашої фігури, використовуємо функцію «Place Image». Використовуючи співчитання клавиш Shift + Ctrl + K или Shift + Cmd + K, вибіріть файл preview.jpg зз папки Images, а потім клікните по фігурі, щоб вставити зображенні прямо в неї.

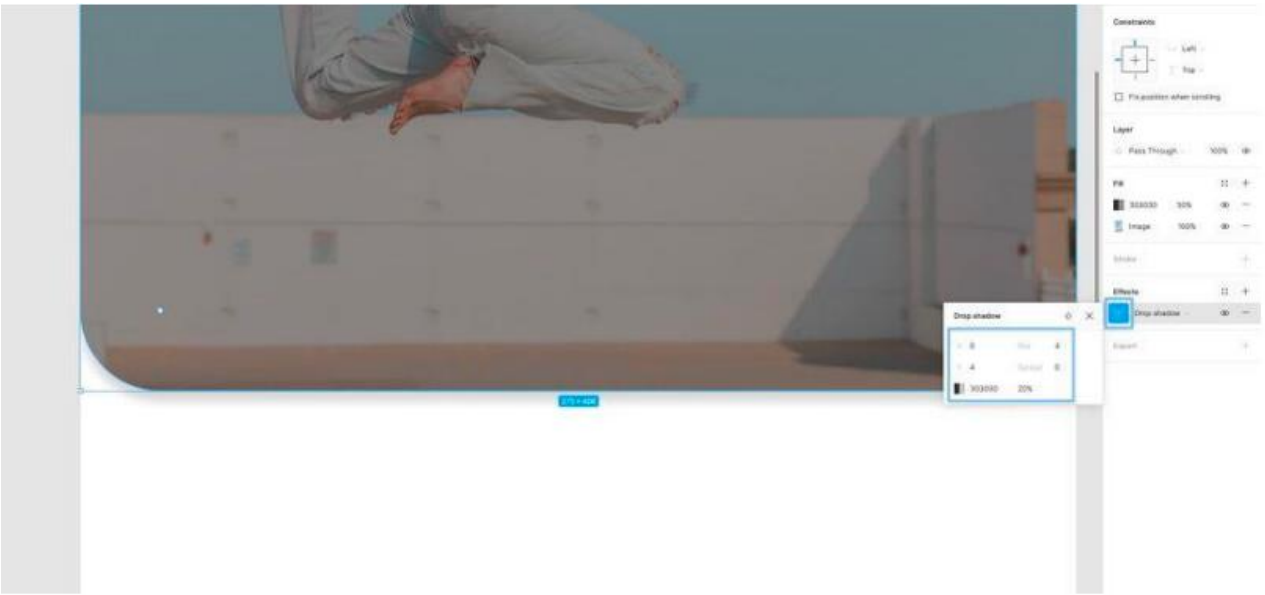


Давайте застосуємо кольорове накладання до нашого зображення, щоб додати трохи контрасту між ним та елементами, які ми незабаром розмістимо поверх нього. На панелі «Заливка» (Fill) клацніть піктограму «Плюс» (+), щоб додати нову заливку. Переконайтеся, що вибрано параметр Solid.

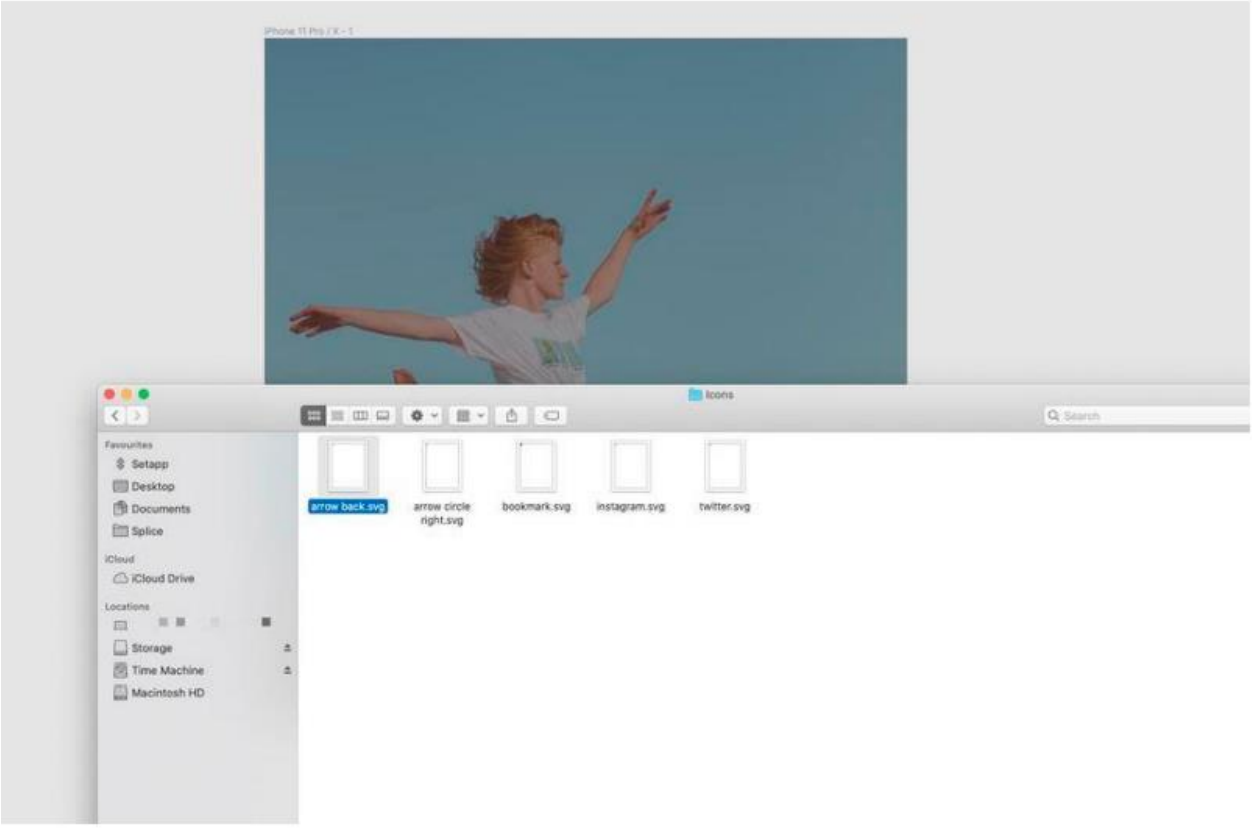
У спливаючому меню введіть наступне значення Hex (#): 303030 і встановіть непрозорість (opacity) на 50%.



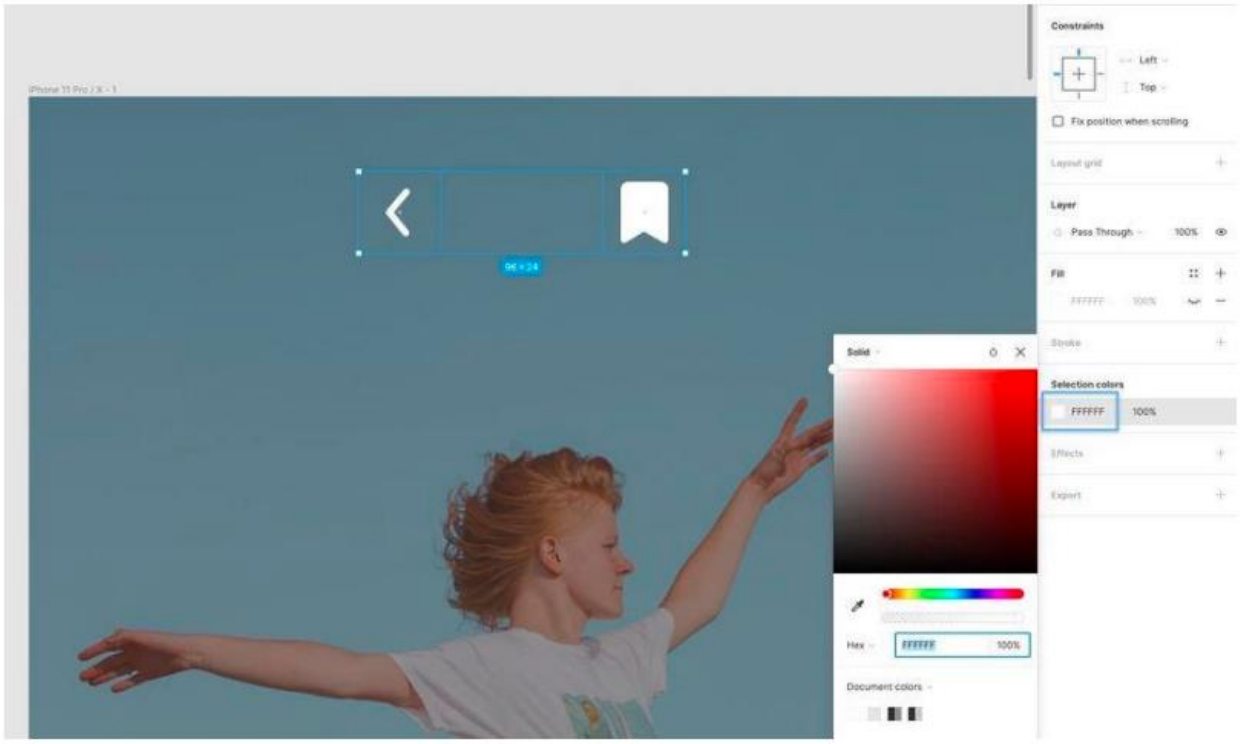
Нарешті, давайте додамо легку тінь (drop shadow) до мініатюри і додамо глибини нашому дизайну. На панелі «Effects» клацніть значок плюс (+), виберіть «Drop Shadow» у меню вибору, а потім налаштуйте наступні параметри ефекту. X - 0 Y - 4 Розмиття (Blur) - 4 Колір (Color) - #303030 Непрозорість (Opacity) – 20%



Панель навігації (Navigation Bar) Для навігаційної панелі ми вставимо пару іконок. Одну для повернення до попереднього екрану, а іншу для додавання статті до закладок. Відкривши папку Icons, просто перетягніть на кадр наступні іконки, по одній... -стрілка тому (arrow back) - закладка (bookmark)



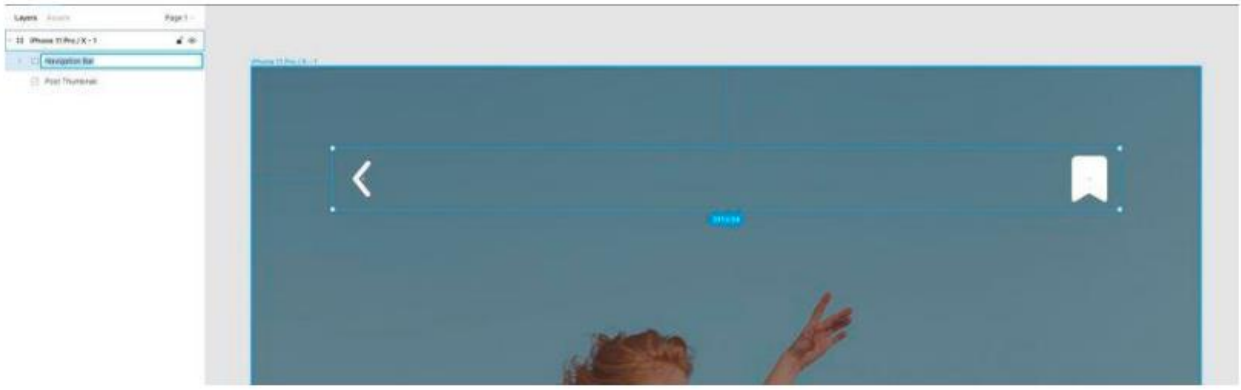
Виділивши обидві іконки, перейдіть на панель «Selection Colors» та змініть значення Hex-значення кольору на білий (FFFFFF).



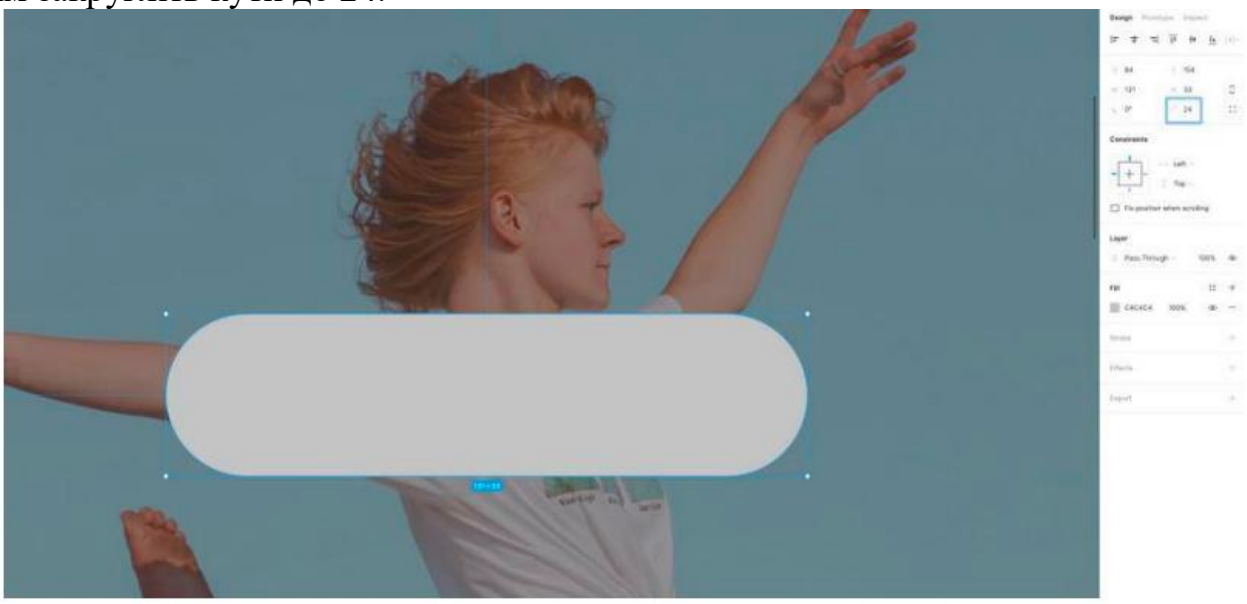
Утримуючи клавішу Alt, щоб виміряти відстань між елементами дизайну, помістіть кожну іконку на 32px від лівого та правого країв кадру.

Потім, вибравши обидві іконки, перейдіть на панель властивостей та оберіть "Align Vertical Centers" (Alt + V).

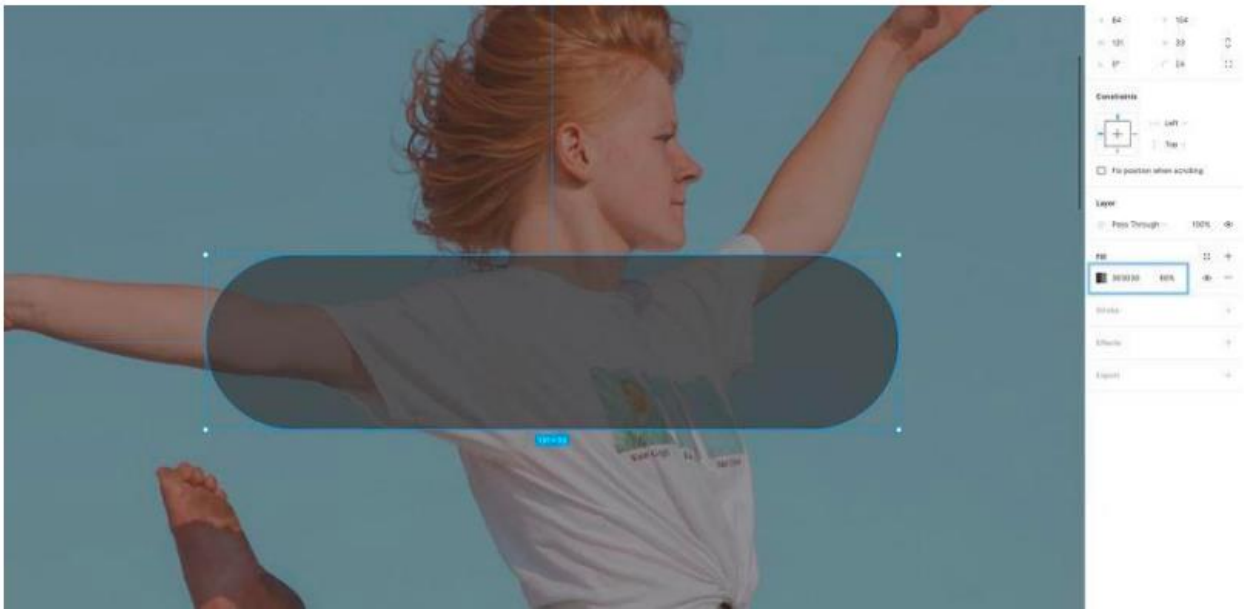
Коли обидві іконки все ще виділені, використовуйте клавіші Ctrl + G або Cmd + G, щоб згрупувати їх, і Ctrl + R або Cmd + R, щоб перейменувати. Це просто додасть акуратного та організованого вигляду на панелі шарів.



Завжди перейменовуйте та групуйте по ходу справи. Це заощаджує купу часу. Тепер, утримуючи Alt, помістіть цю нову групу 56px від верхнього краю основного кадру. Сповіщення Як бачите, для іконки закладки ми вибрали версію із заливкою, ніби користувач щойно додав публікацію до закладок. Давайте додамо просте повідомлення, яке може відображатись користувачеві при натисканні на іконку закладки. Ненав'язливий візуальний сигнал, що показує, що дія (натискання на іконку) була розпізнана та виконана. Інструментом «Прямокутник» (R) намалюйте фігуру розміром 131px x 33px, а потім закругліть кути до 24.

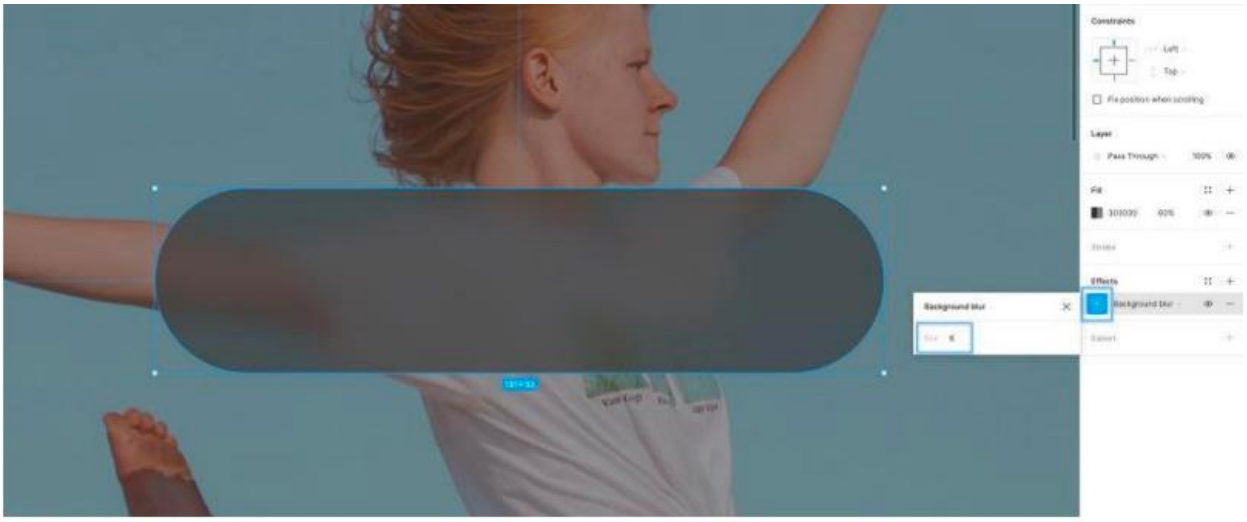


На панелі «Заливка» (Fill) задайте Hex-значение (#) 303030 та зменшіть його непрозрачність до 60%.

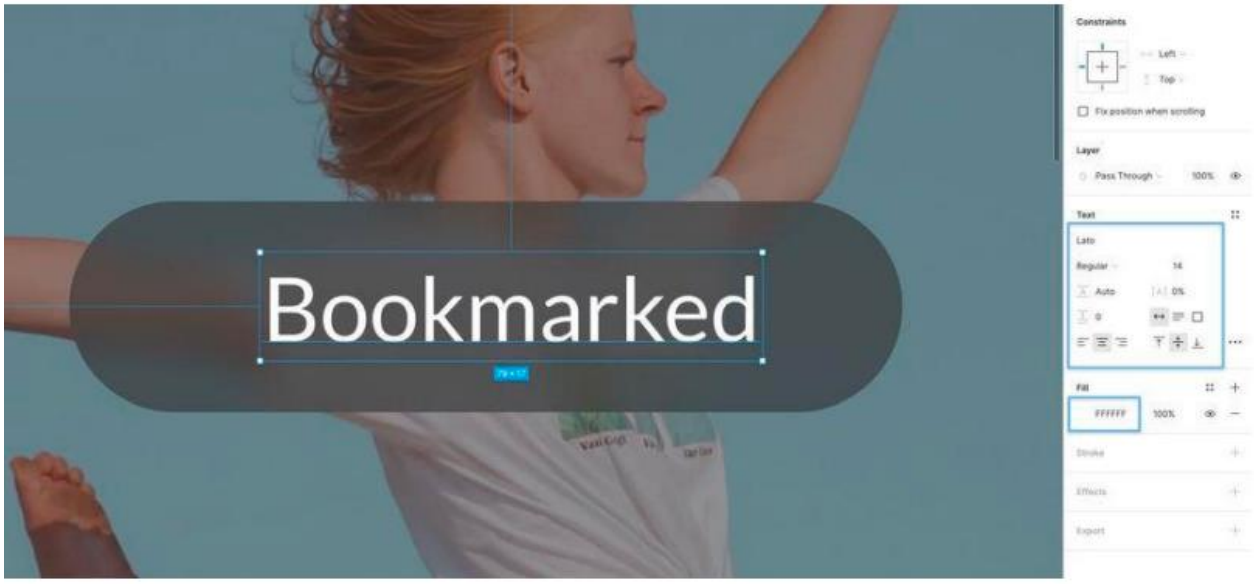


На панель «Ефекти» (Effects) додамо цьому маленькому повідомленню легке розмиття фону, щоб додати візуальний інтерес і трохи більшого розмаїття між зображенням та текстом, який ми скоро розмістимо в ньому. На панелі «Ефекти» клацніть значок плюс (+), а потім виберіть Background Blur (Розмиття фону) у меню вибору.

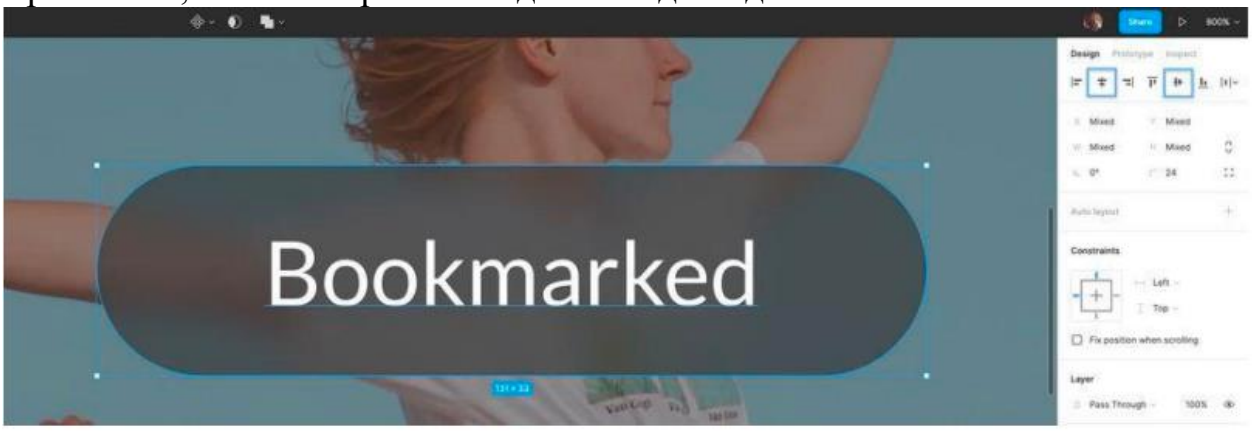
Налаштування на (Effect Settings) і зміна силу розмиття (Blur Strength) на 6.



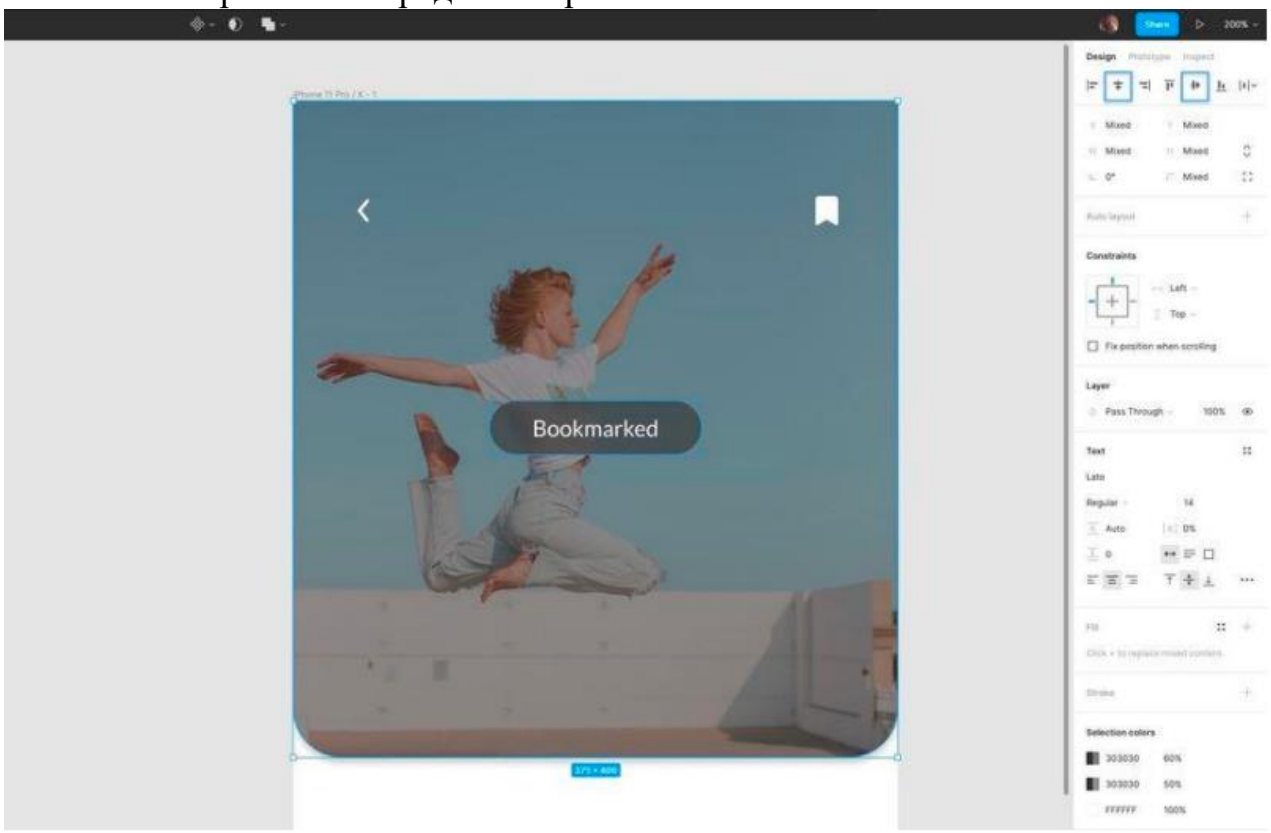
Настав час написати трохи тексту та налаштувати ще кілька властивостей, чи не так? Використовуючи інструмент «Текст» (T), додайте новий текстовий елемент та надрукуйте слово «Bookmarked» (додано до закладок). Потім на панелі властивостей тексту (Text Properties) застосуйте таке… Шрифт (Font) - Lato Вага (Weight) - Regular Розмір (Size) – 14 Вирівнювання тексту (Text Align) - Center Вирівнювання (Align) - Middle Заливка (Fill) - #FFFFFF



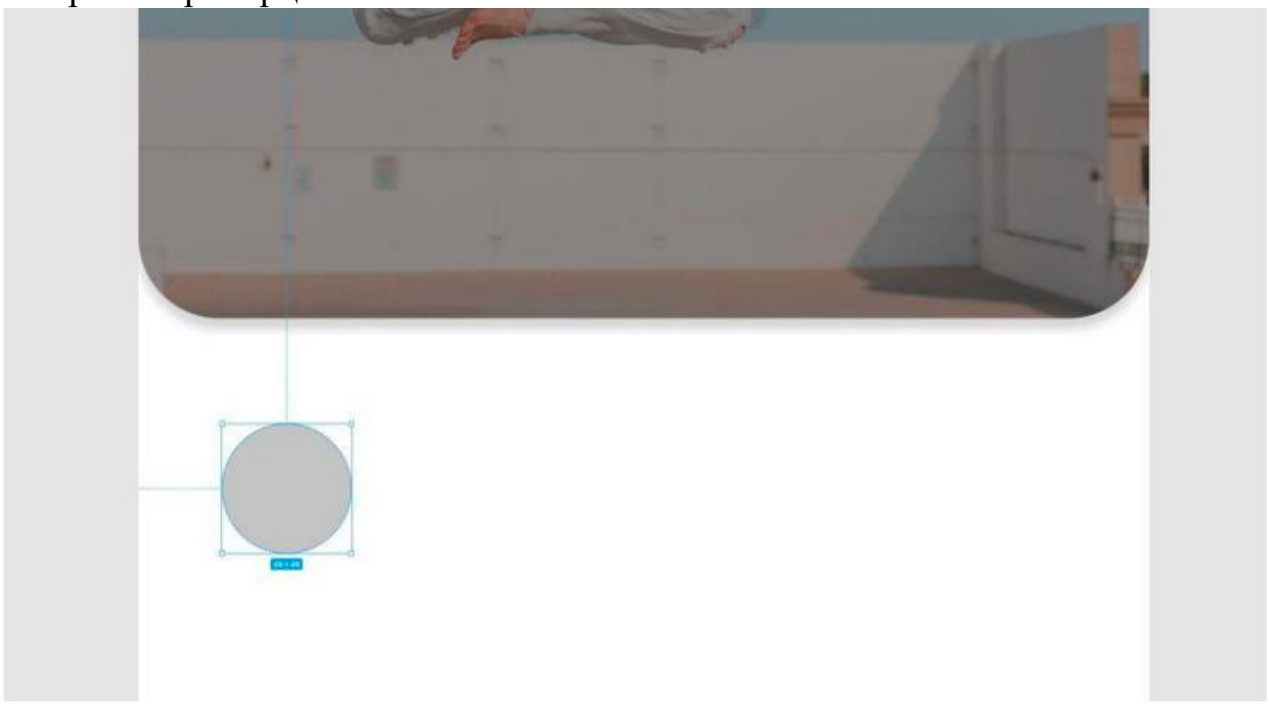
Вибравши текст та шар фігури, використовуйте параметри вирівнювання (Alignment) у верхній частині панелі властивостей, щоб вирівняти елементи як по горизонталі, так і по вертикалі відносно один одного.



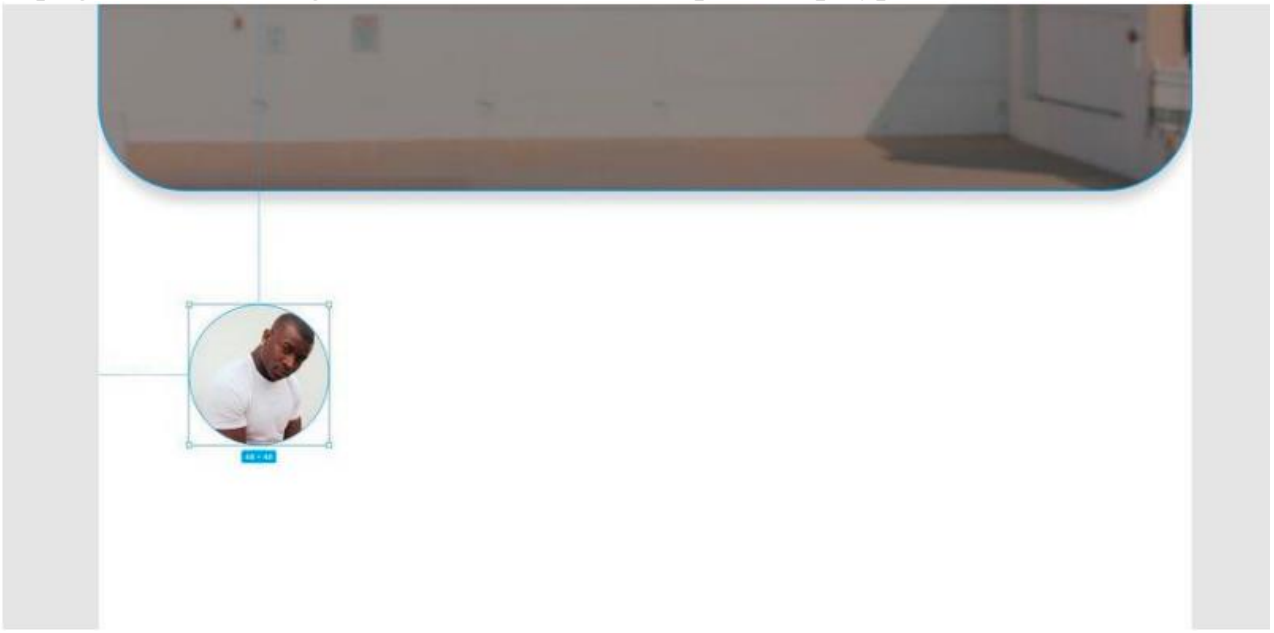
Використовуйте клавіші Ctrl + G або Cmd + G, щоб згрупувати їх, і Ctrl + R або Cmd + R, щоб перейменувати. Потім, нарешті, використовуйте Ctrl+Click або Cmd+Click, щоб вибрати групу повідомлень та зображення, а потім використовуйте параметри вирівнювання, щоб вирівняти повідомлення по горизонталі та вертикалі всередині зображення.



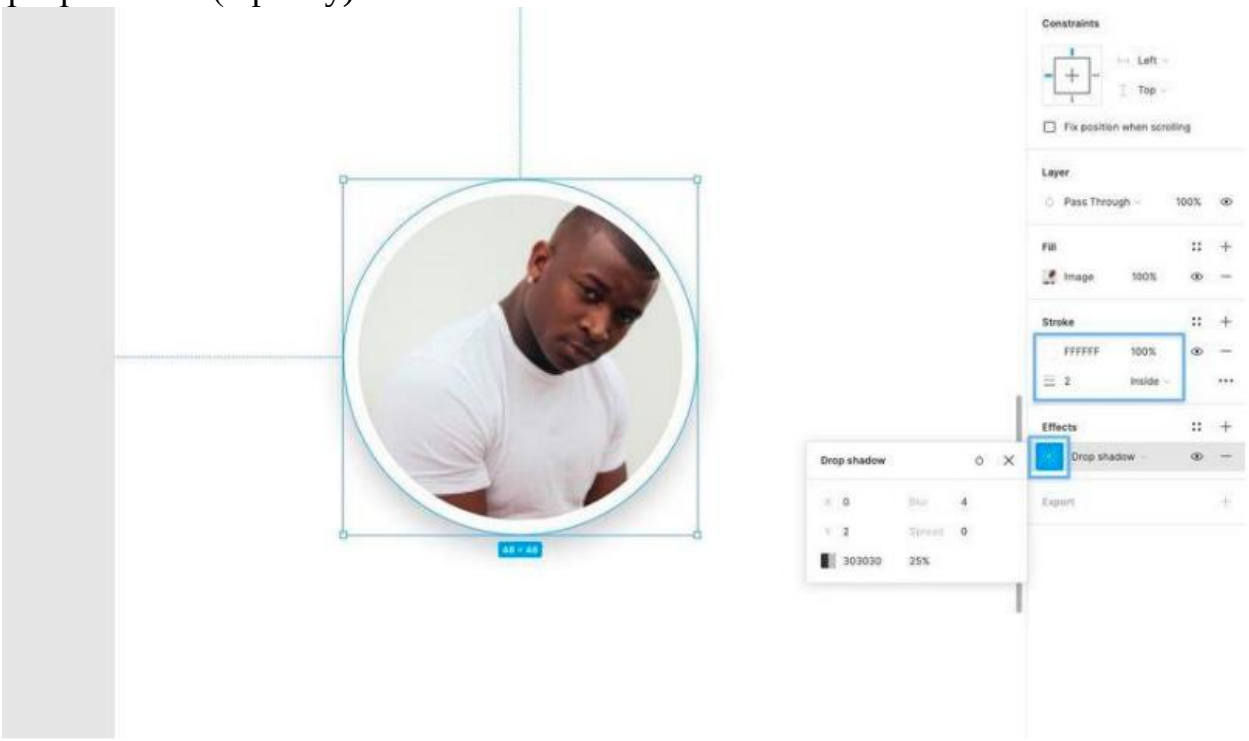
Аватар + іконки соціальних мереж Давайте розмістимо аватар, ім'я автора та іконки соціальних мереж на свої місця. Виберіть інструмент Еліпс (O) та намалюйте фігуру розміром 48 x 48px. Також можна утримувати клавішу Shift, щоб зберегти пропорції.



Використовуючи клавіші Shift+Ctrl+K або Shift+Cmd+K, щоб використовувати інструмент «Помістити зображення» (Place Image), виберіть avatar.png з папки «Images» і помістіть його всередині фігури.



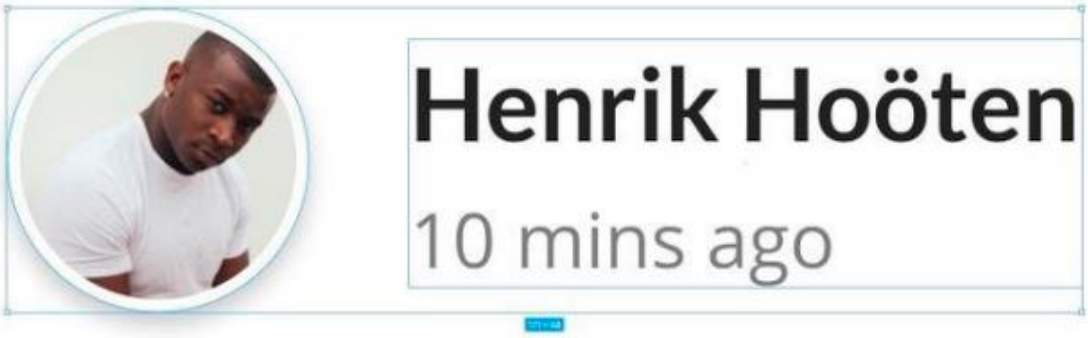
Виділивши фігуру, перейдіть на панель властивостей і додайте наступне… Stroke – Колір (Color) - #FFFFFF Ширина (Width) - 2 Положення (Position) - Inside Drop Shadow – X - 0 Y - 2 Розмиття (Blur) - 4 Колір (Color) - #303030 • Непрозрачность (Opacity) - 25%



Давайте додамо ще кілька текстових шарів як для імені автора, так і для дати публікації статті. Використовуючи інструмент «Текст» (T), створіть 2 нових текстових шари, відредагуйте текст відповідним чином, а потім застосуйте до імені автора такі властивості. Шрифт (Font) - Lato Вага (Weight) - Bold Розмір (Size) – 16 Вирівнювання тексту (Text Align) - Left Заливання (Fill) - #212121

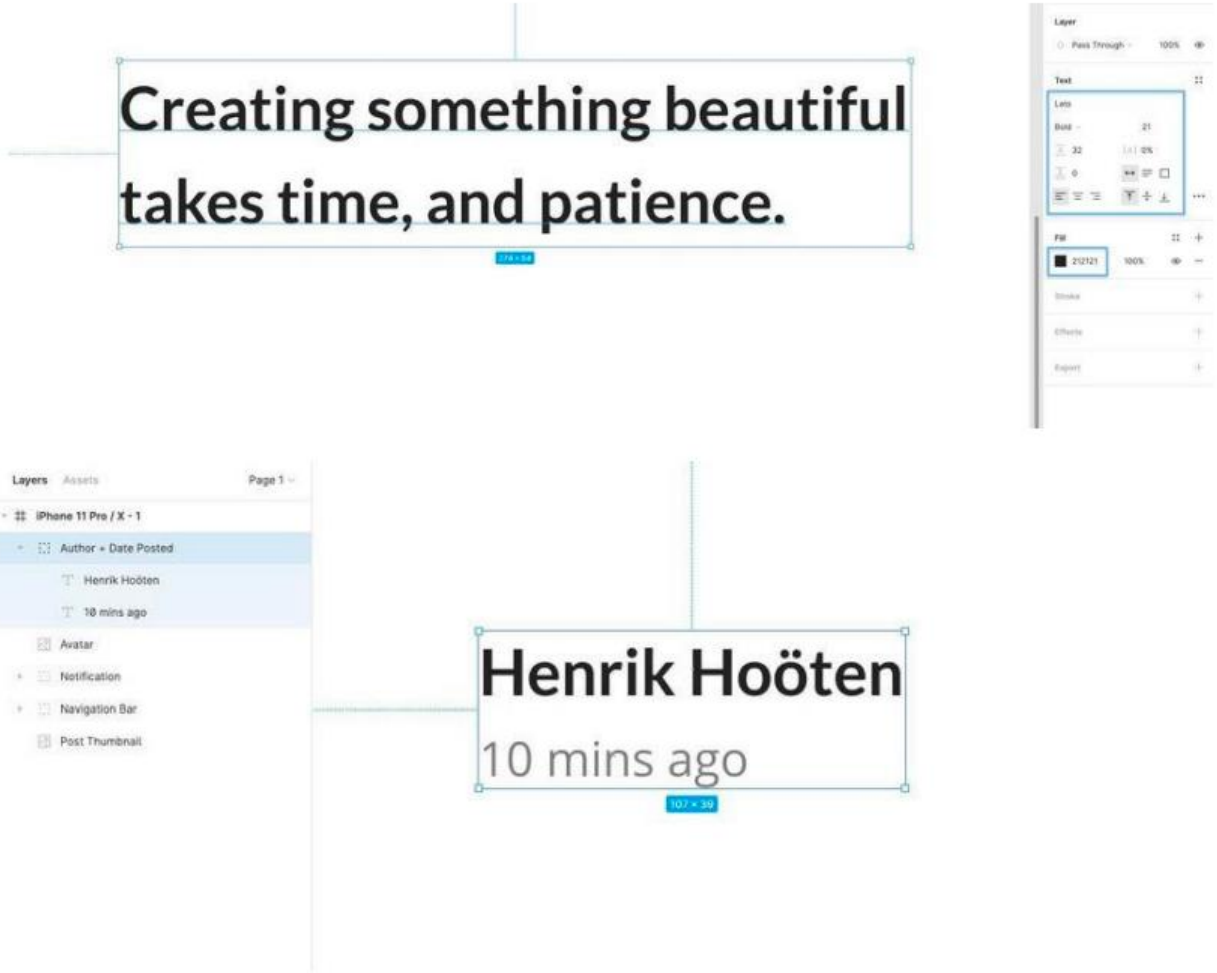
А для дати публікації використовуйте наступне… Шрифт (Font) - Open Sans Вага (Weight) - Regular Розмір (Size) - 12 Вирівнювання тексту (Text Align) – Left Заливання (Fill) - #7D7D7D

Використовуючи клавішу Alt, розташуйте текстові шари відносно один одного, даючи їм невеликий простір. Потім використовуйте інструменти вирівнювання, щоб вони були вирівняні лівим краєм відносно один одного, і, нарешті, згрупуйте їх (Ctrl + G або Cmd + G).

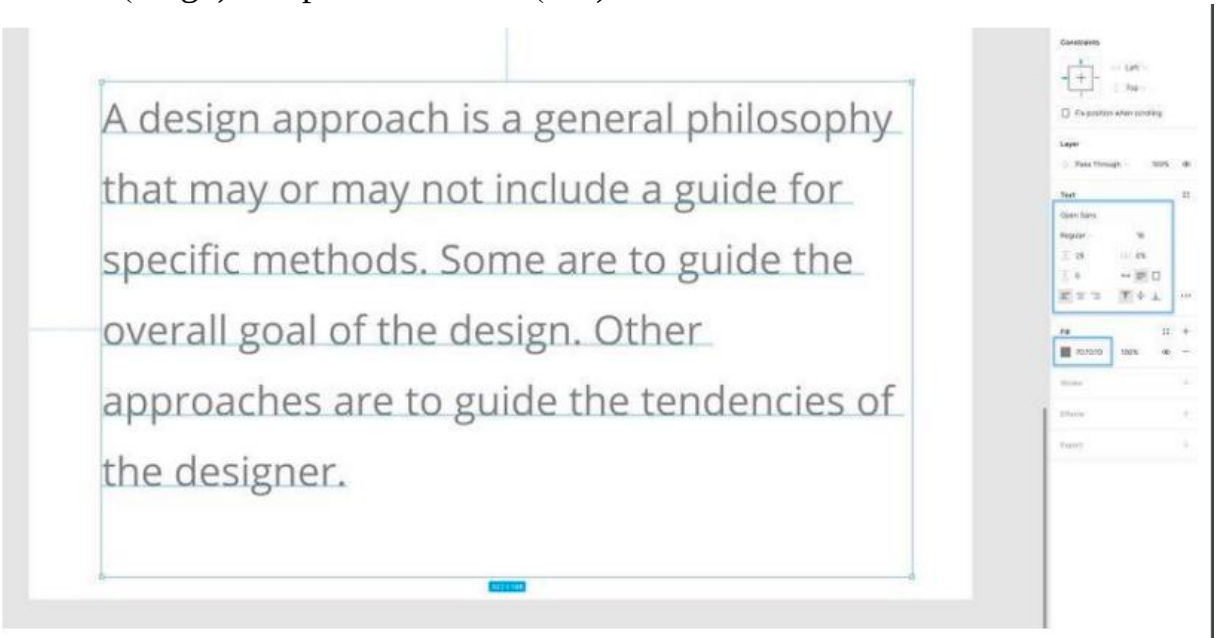


Давайте додамо пару іконок соціальних мереж. Як і раніше, просто перетягніть по одній іконці Instagram і Twitter у свій дизайн і, використовуючи комбінацію інструментів вирівнювання та клавіші Alt, вирівняйте їх по вертикалі один до одного (Alt + V) і розмістіть їх. На відстані 16px.

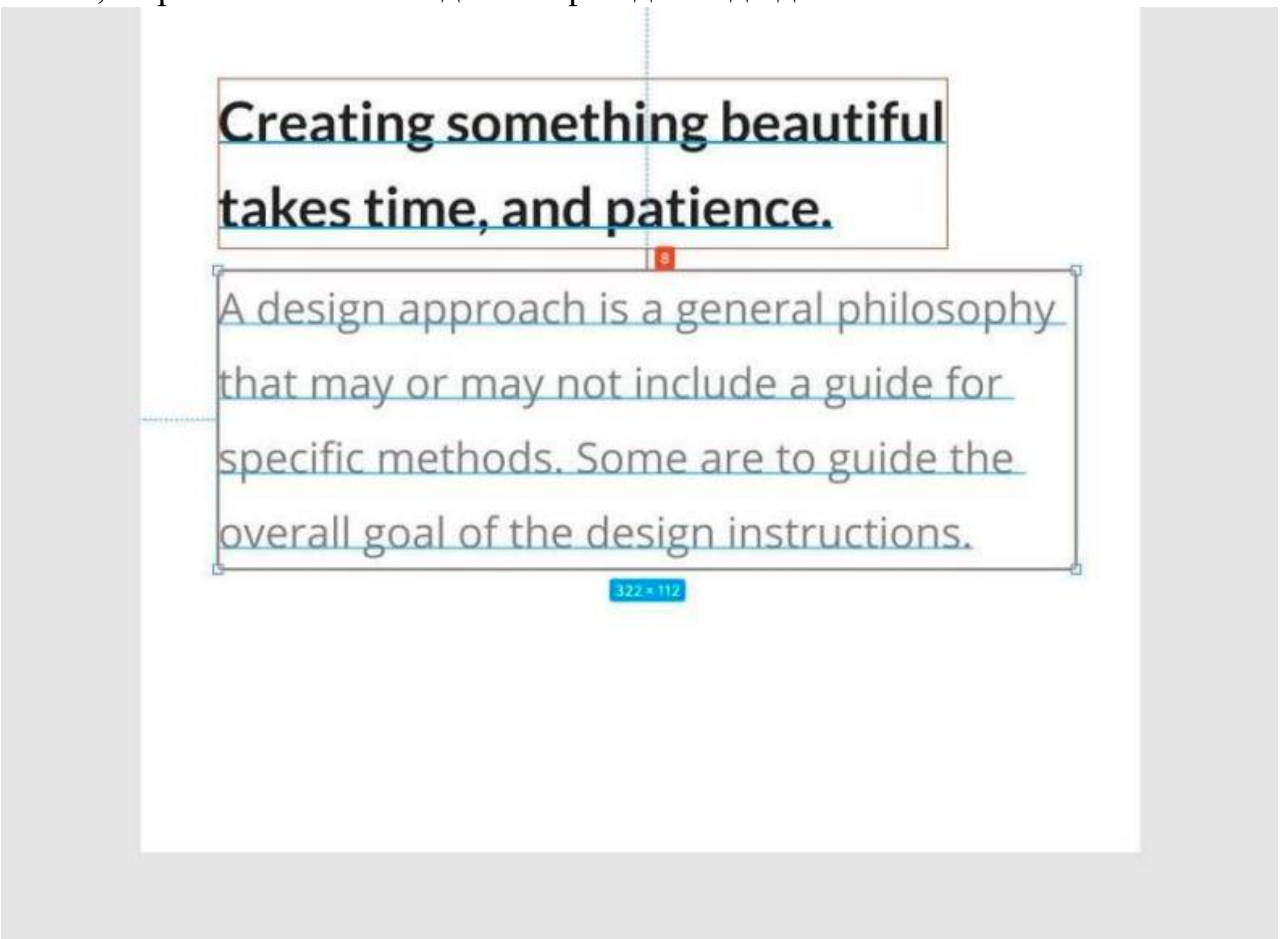
Згрупуйте іконки (Ctrl + G або Cmd + G) та перейменуйте (Ctrl + R або Cmd + R). Виділивши групу іконок соціальних мереж, групу текстових шарів та аватар, використовуйте Alt+V, щоб вирівняти їх по вертикалі щодо один одного. Давайте додамо невеликий заголовок та фрагмент контенту, щоб ще трохи доповнити наш дизайн. Використовуючи інструмент «Текст» (T), створіть простий заголовок, а потім на панелі властивостей застосуйте наступне… • • Шрифт (Font) - Lato • • Вага (Weight) – Bold • • Розмір (Size) – 21 • • Висота рядка (Line Height) – 32 • • Вирівнювання тексту (Text Align) – Left • • Вирівнювання (Align) – Top • • Заливання (Fill) - #212121



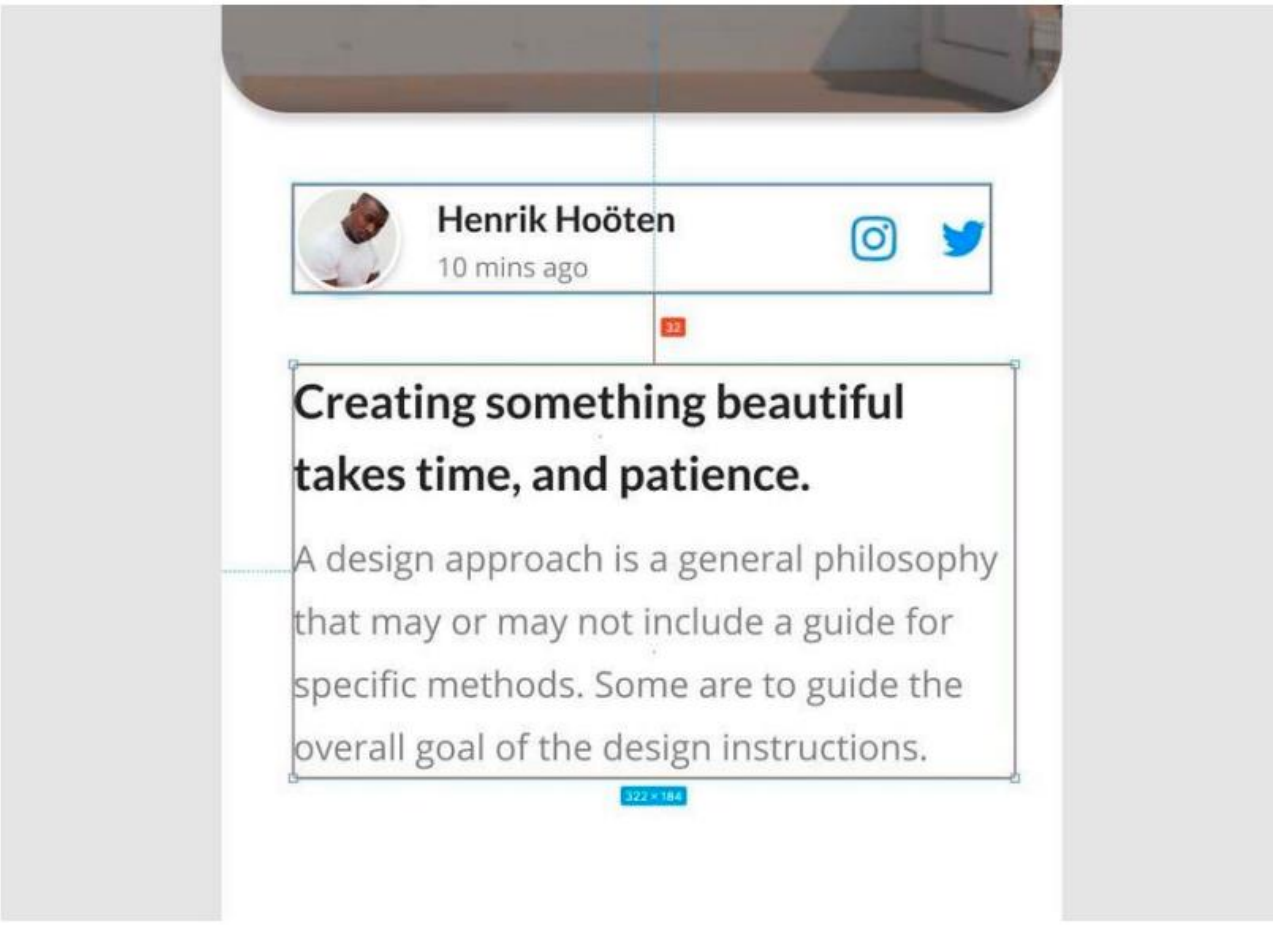
Потім, використовуючи інструмент «Текст» (T) ще раз, створіть абзац контенту, який відображатиметься під заголовком з такими властивостями. • Шрифт (Font) - Open Sans • Вага (Weight) - Regular • Розмір (Size) - 16 • Висота рядка (Line Height) – 28 • Вирівнювання тексту (Text Align) - Left • Вирівнювання (Align) - Top • Заливання (Fill) - #7D7D7D



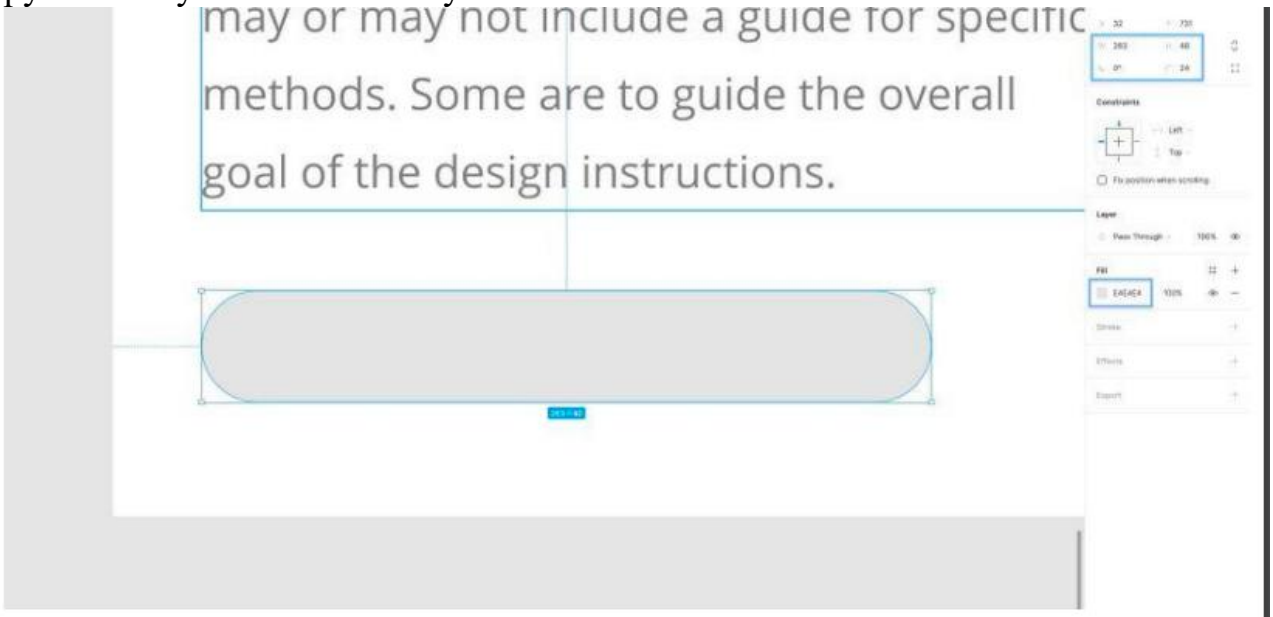
Ми встановили колір цього блоку тексту на темно-сірий (#7D7D7D), щоб він красиво контрастував із заголовком, додав трохи візуального інтересу, але все ще був доступний на білому тлі для користувачів з ослабленим зором. Утримуючи клавішу Alt, наведіть курсор на два шари, щоб побачити відстань між ними, та розмістіть їх на відстані 8px один від одного.



Не забудьте згрупувати (Ctrl + G або Cmd + G) та перейменувати (Ctrl + R або Cmd + R), а потім розмістіть цю нову групу на 32px від лівого краю кадру та на 32px від нижньої частини аватара групи.

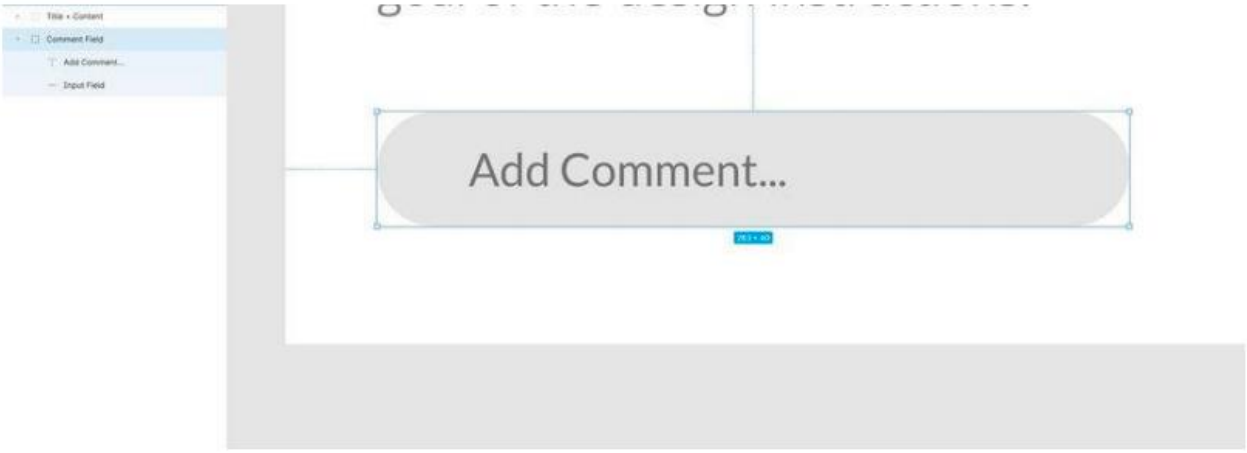


Форма коментаря ОК. Давайте підіб'ємо підсумки і додамо в наш дизайн невелику форму / поле коментаря. Знову викликавши інструмент «Прямокутник» (R), намалюйте фігуру розміром 263 x 40px, задайте їй радіус заокруглення кутів 24 та заливку #E4E4E4.

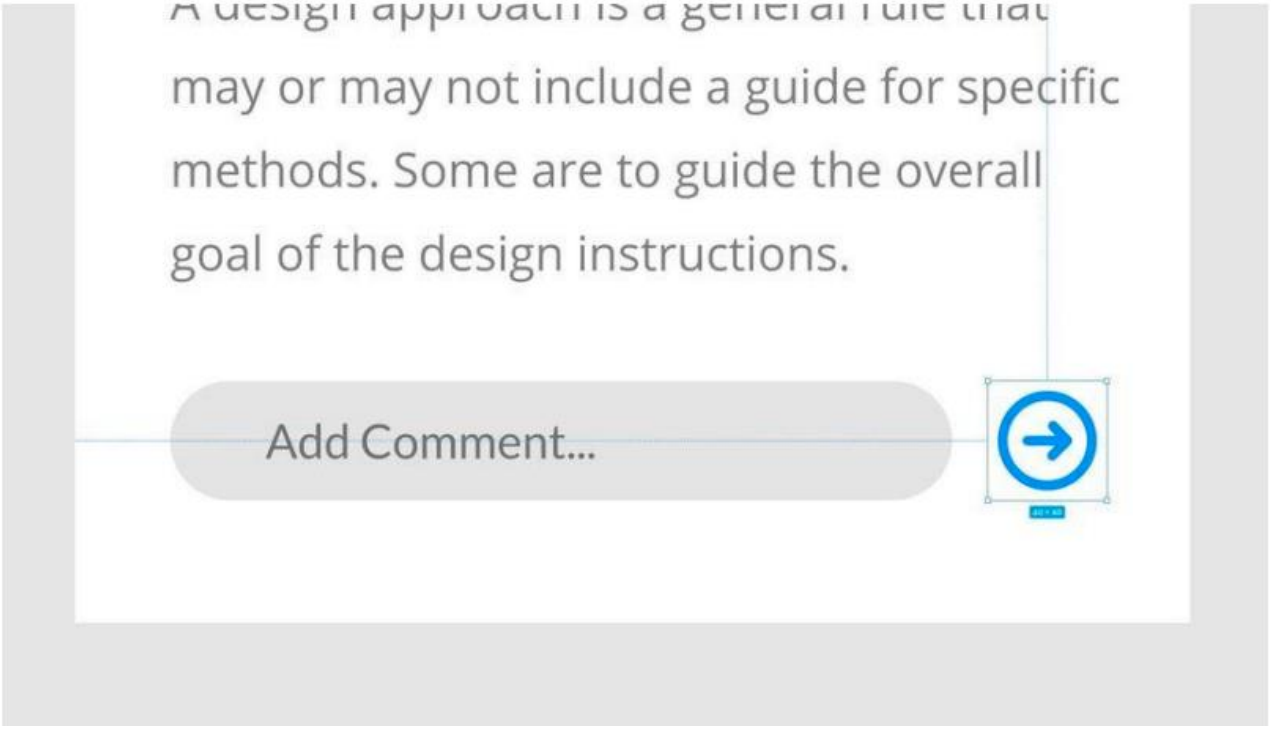


Вставте текстовий шар (T), змініть формулювання (наприклад, «Додати коментар…»), а потім додайте такі властивості… • Шрифт (Font) - Lato • Вага (Weight) - Regular • Розмір (Size) – 16 • Вирівнювання тексту (Text Align) - Left • Вирівнювання (Align) - Middle • Заливання (Fill) - #707070

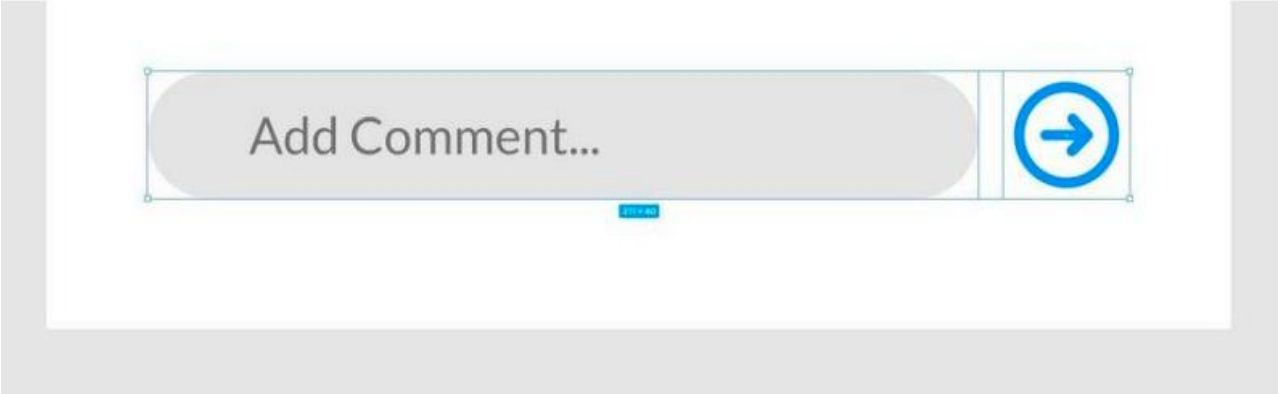
Виділивши і фігуру, і текстовий шар, вирівняйте їх по вертикалі відносно один одного (Alt + V), а потім згрупуйте (Ctrl + G або Cmd + G) їх і перейменуйте (Ctrl + R або Cmd + R).



Перетягніть у свій дизайн кружок зі стрілкою вправо, а потім за допомогою інструмента «Scale» (K) змініть його розмір пропорційно до 40 x 40px, щоб він мав ті самі розміри за висотою, що й поле, яке ми щойно створили.



Помістіть іконку на 8px праворуч від поля коментаря, а потім, вибравши групу полів та іконку, вирівняйте їх по вертикалі один до одного (Alt + V).



Згрупуйте 2 елементи (поле коментаря та іконку). І помістіть цю нову групу на 32px від нижньої частини тексту, а також від лівого та правого країв кадру.

Список літератури

.

1. Ткачук М.В. Уніфіковани програмні сервіси та візуальні інтерфейси в інтранет-системах управління технологічними процесами - Системні дослідження та інформаційні технології -№1 - 2004.

2. Доценко С. І. Людино-машинний інтерфейс: навч. посібник. – Харків: УкрДУЗТ, 2022. – 135 с.

3. Інтерфейс "Користувач-комп'ютер": Навчальний посібник / В.П. Майданюк, А.М. Пєтух. - Вінниця: ВДТУ, 1999. - 66 с.

4. Пупена О.М. Розроблення людино-машинних інтерфейсів та систем збирання даних з використанням програмних засобів SCADA/HMI. Навчальний посібник / Пупена О.М. - Видавництво: Ліра-К., 2020. - 594 с.

Н а в ч а л ь н о - м е т о д и ч н е в и д а н н я

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З ДИСЦИПЛІНИ

“Людино-машинний інтерфейс”. Частина 1

Підготували
Сугоняк Інна Іванівна, Кравченко Світлана Миколаївна,
Гришкун Євгеній Олександрович

Технічний редактор – **І.І.Сугоняк**
Комп’ютерне верстання **– Є.О.Гришкун**

Підписано до друку 15.02.23. Формат 60×84/16.
Ум. друк. арк. 3.016. Зам. офс.