Workshop Guidelines

Part 1: UI

Components

Grid

ColumnDefinitions = *,*

RowDefinitions = *, auto

CollectionView

Grid.Row="0"

Grid.Column="0"

Grid.ColumnSpan="2"

HorizontalOptions="FillAndExpand"

VerticalOptions="Center"

CollectionViewEmptyView

Image

HeightRequest="200"

Source="nodata.png"

WidthRequest="200"

Inside Collectionview

Grid - Padding = 10

Border - HeightRequest = 125

Grid - Padding = 0 ColumnDefinitions= 125, *

Image

Aspect="AspectFill"

HeightRequest="125"

WidthRequest="125"

VerticalStackLayout

Grid.Column="1"

Padding="10"

VerticalOptions="Center"

Inside Vertical Stacklayout

Two labels

Button

Grid.Row="1"

Grid.Column="0"

Grid.ColumnSpan="2"

Margin="10,0,10,10"

BackgroundColor="#038aff"

FontSize="16"

Text="Get Monkeys"

TextColor="White"

ActivityIndicator

Grid.RowSpan="2"

Grid.ColumnSpan="2"

HorizontalOptions="Fill"

IsRunning="False"

IsVisible="False"

VerticalOptions="Center"

Color="#038aff"

Shell

Shell.BackgroundColor="#038aff"

Shell.NavBarIsVisible="True"

Shell.TitleColor="White"

Part 2: MVVM

Create folders

- Models
- ViewModels
- Views

Move MainPage to Views folder

Create MonkeyModel class

Create Monkey Properties

- public string? Name { get; set; }
- public string? Location { get; set; }
- public string? Details { get; set; }
- public string? Image { get; set; }
- public int Population { get; set; }
- public double Latitude { get; set; }
- public double Longitude { get; set; }

Create MonkeysViewModel class

Inherit ObservableObject

Create ObservableCollection property

- public ObservableCollection<Monkey> Monkeys { get; } = new();

Create isBusy property

- [ObservableProperty]

  bool isBusy;

MainPage

Declare References

- xmlns:model="clr-namespace:MonkeyApp.maui.Models"

- xmlns:viewmodel="clr-namespace:MonkeyApp.maui.ViewModels"
- x:DataType="viewmodel:MonkeysViewModel"

MainPage.xaml.cs

Depedency Injection

Inject MonkeysViewModel

- public MainPage(MonkeysViewModel monkeysViewModel)

{

InitializeComponent();

BindingContext = monkeysViewModel;

}

MauiProgram.cs

Register MainPage and MonkeysViewModel

- builder.Services.AddTransient<MonkeysViewModel>();
- builder.Services.AddTransient<MainPage>();

Part 3: API

Create Services folder

- Services

Create MonkeyService class inside Services folder

Create MonkeyService

```
public class MonkeyService
  {
    HttpClient client;
    public MonkeyService()
    {
      client = new HttpClient();
```

```
        }
        List<Monkey> monkeyList = new();
        public async Task<List<Monkey>> GetMonkeys()
        {
            if (monkeyList?.Count > 0)
                return monkeyList;

            var url = "https://raw.githubusercontent.com/jamesmontemagno/app-
monkeys/master/MonkeysApp/monkeydata.json";

            var response = await client.GetAsync(url);

            if (response.IsSuccessStatusCode)
            {
                monkeyList = await
response.Content.ReadFromJsonAsync<List<Monkey>>();
            }

            return monkeyList;
        }
    }
```

MonkeysViewModel

Inject MonkeyService

```
public MonkeysViewModel(MonkeyService monkeyService)
        {
            this.monkeyService = monkeyService;
        }
```

Create GetMonkeyCommand

```
    [RelayCommand]
    async Task GetMonkeyAsync()
    {
        try
        {
            IsBusy = true;
            var monkeys = await monkeyService.GetMonkeys();
            if (Monkeys.Count != 0)
                Monkeys.Clear();

            foreach (var monkey in monkeys)
                Monkeys.Add(monkey);

            IsBusy = false;
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex);
            await Shell.Current.DisplayAlert("Error", $"Unable to get monkeys: {ex.Message}", "Ok");
            IsBusy = false;
        }
    }
}
```

MainPage

Bind ObservableCollection Monkey in CollectionView

ItemsSource="{Binding Monkeys}"

Set CollectionView DataTemplate

- x:DataType="model:Monkey"

Bind Image to Image and label to Name and Location

- <Image Source="{Binding Image}"/>
- <Label Text="{Binding Name}" />
- <Label Text="{Binding Location}" />

Bind button to GetMonkeyCommand

- Command="{Binding GetMonkeyCommand}"

Bind ActivityIndicator to IsBusy Property

- IsRunning="{Binding IsBusy}"
- IsVisible="{Binding IsBusy}"

Part 4: Navigation

In Views folder, create DetailsPage

Create DetailsPage UI

```
<ScrollView>
    <Grid RowDefinitions="Auto,Auto,*">
        <BoxView
            Grid.RowSpan="2"
            BackgroundColor="#038aff"
            HorizontalOptions="Fill"
            VerticalOptions="Fill" />

        <Border
            Margin="0,8,0,0"
```

```
            HeightRequest="172"
            HorizontalOptions="Center"
            Stroke="White"
            StrokeShape="RoundRectangle 80"
            StrokeThickness="6"
            VerticalOptions="Center"
            WidthRequest="172">
            <Image
                Aspect="AspectFill"
                HeightRequest="160"
                HorizontalOptions="Center"
                Source="{Binding Monkey.Image}"
                VerticalOptions="Center"
                WidthRequest="160" />
        </Border>

        <VerticalStackLayout
            Grid.Row="2"
            Padding="10"
            Spacing="10">
            <Label FontSize="20" Text="{Binding Monkey.Name}" />
            <Label FontSize="14" Text="{Binding Monkey.Details}" />
            <Label FontSize="14" Text="{Binding Monkey.Location,
StringFormat='Location: {0}'}" />
            <Label FontSize="14" Text="{Binding Monkey.Population,
StringFormat='Population: {0}'}" />
        </VerticalStackLayout>

    </Grid>
```

```
</ScrollView>
```

Create DetailsViewModel Class

Set QueryParameter and create monkey property

```
[QueryProperty(nameof(Monkey), nameof(Monkey))]
public partial class DetailsViewModel : ObservableObject
{
    [ObservableProperty]
    Monkey monkey;

    public DetailsViewModel()
    {

    }
}
```

Set references, title of the page, and content page attributes

- xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
- xmlns:model="clr-namespace:MonkeyApp.maui.nav.Models"
- xmlns:viewmodel="clr-namespace:MonkeyApp.maui.nav.ViewModels"
- Title="Detail Page"
- x:DataType="viewmodel:DetailsViewModel"
- Shell.BackgroundColor="#038aff"
- Shell.NavBarIsVisible="True"
- Shell.TitleColor="White"

In MonkeyViewModel, Create NavigateToDetailsPage Command

```
[RelayCommand]
async Task NavigateToDetailsPage(Monkey monkey)
{
    await Shell.Current.GoToAsync(nameof(DetailsPage), true, new
Dictionary<string, object>
    {
        {"Monkey", monkey }
    });
}
```

MainPage

Create Border tap gesture

```
<Border HeightRequest="125">
    <Border.GestureRecognizers>
                <TapGestureRecognizer Command="{Binding
Source={RelativeSource AncestorType={x:Type viewmodel:MonkeysViewModel}},
x:DataType=viewmodel:MonkeysViewModel, Path=NavigateToDetailsPageCommand}"
CommandParameter="{Binding .}" />
    </Border.GestureRecognizers>
```

Dependency Injection

Inject DetailsViewModel

```
public DetailsPage(DetailsViewModel detailsViewModel)
{
    InitializeComponent();
    BindingContext = detailsViewModel;
}
```

MauiProgram.cs

Register DetailsPage and DetailsViewModel

```
builder.Services.AddTransient<DetailsViewModel>();
builder.Services.AddTransient<DetailsPage>();
```

Register DetailsPage Route

```
Routing.RegisterRoute(nameof(DetailsPage), typeof(DetailsPage));
```