Name of Experiment: Country Database	**Exp No:** DB2

Background: Student should have basic knowledge of C#.

Summary: Database Management is one of the key factors in any Mobile application development framework. We need to understand each aspect of database at all levels i.e. Physical level, Logical level and View level. In order to do so Windows Phone has MVVM architecture.

Learning Objective: To learn the MVVM architecture in Database using this experiment.

Target Platforms: This experiment is tested on Windows Phone Emulator and Nokia Lumia 800.

Procedure:

Step1. Repeat steps [1-4] as in Experiment no MC1. [Refer Hello World]

Step2. Add Reference for Microsoft.Phone.Controls and add this line `xmlns:controls="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"` to the Phone.ApplicationPage.

Step3. Also, add reference for System.Data.Linq to the current experiment.

Step4. Copy and paste the MainPage.xaml code from the source code section.

Step5. Add New Folder and named it as Images by making right click on the Experiment Name.

Step6. Copy required images from the Windows Phone SDK folder and placed inside the Images folder.

Step7. Set the properties of each images as Build Action= “Content” and Copy to Output Directory= “Copy always”.

Step8. Create two more folders and named it as Model and ViewModel.

Step9. Add new class and named it as CountryDataContext.cs inside the Model folder.[Refer Source Code Section]

Step10.Open CountryDataContext.cs class and add using System, System.ComponentModel, System.Data.Linq and System.Data.Linq.Mapping

Step11.CountryDataContext.cs class contains actual schema for the database. Here, we have two tables namely City and Country and their columns. DataContext interface is implemented in order to create the connectionString to the database. We also, implement INotifyPropertyChanged event in order to track the changes and reflect back to the database. [Refer Source Code Section]

Step12.Similarly, add new class CountryViewModel.cs inside the ViewModel folder.[Refer Source Code Section]

Step13. Open CountryViewModel.cs class and using System.Linq, System.ComponentModel, CountryDatabase.Model, System.Collection.ObjectModel and System.Collection.Generic.

Step14. Here, we initialise the CountryDB as DataContext object in order to link the Views to the Model. This class also contains various functionality like SaveChangesToDB(), LoadCollectionFromDatabase(), AddCity(), SearchCity() and DeleteCity() etc.[Refer Source Code Section]

Step15. Add new page named ItemSearch.xaml, design the UI as given in the snapshot then copy and paste the content inside ItemSearch.xaml.cs [Refer Source Code Section]

Step16. Add one more new page named NewTaskPage.xaml, design the UI part as given in the snapshot then copy and paste the content inside NewTaskPage.xaml.cs class.[Refer Source Code Section]

Step17. Open the App.xaml.cs class, define viewModel object before the constructor of the class. Now, inside the constructor define the DBConnectionString path and load the dropdown list as soon as the app is loaded.

Step18. Now, at last open the MainPage.xaml.cs and copy and paste the content over there.[Refer Source Code Section]

Step19. Save all the changes made to the experiment.

Step20.Press F5, in order to debug the experiment on the Windows Phone Emulator.

Step21. In this manner, we are able to deployed the experiment on the Windows Phone Emulator.

Source Code	Comments

MainPage.xaml

```
<phone:PhoneApplicationPage.Resources>
        <DataTemplate x:Key="ToDoListBoxItemTemplate">

            <Grid HorizontalAlignment="Stretch" Width="420">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="100" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="100" />
                </Grid.ColumnDefinitions>

               <CheckBox
                    IsChecked="False"
                    Grid.Column="0" VerticalAlignment="Top"
Background="DarkOliveGreen"/>

                <TextBlock
                    Text="{Binding CityName}"
                    FontSize="{StaticResource
PhoneFontSizeLarge}"
                    Grid.Column="1" Grid.ColumnSpan="2"
                    VerticalAlignment="Top" Margin="-36, 12,
0, 0"/>

                <Button
```

```
                    Grid.Column="3"
                    x:Name="deleteTaskButton"
                    BorderThickness="0"
                    Margin="0, -18, 0, 0"
                    Click="deleteTaskButton_Click">

                    <Image
                    Source="/Images/appbar.delete.rest.png"
                    Height="75"
                    Width="75"/>

                </Button>
            </Grid>
        </DataTemplate>
    </phone:PhoneApplicationPage.Resources>
    <!--LayoutRoot is the root grid where all page content is
placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application
and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0"
Margin="12,17,0,28">
            <TextBlock x:Name="ExperimentTitle" Text="Exp
No:DB2" TextAlignment="Right" Style="{StaticResource
PhoneTextNormalStyle}"/>
            <TextBlock x:Name="ApplicationTitle"
Text="Country Database" Style="{StaticResource
PhoneTextNormalStyle}"/>
            <!--<TextBlock x:Name="PageTitle" Text="page
name" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>-->
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
   <controls:Pivot Margin="0, -36, 0, 0">

      <controls:PivotItem Header="all">
       <ListBox
                      x:Name="allToDoItemsListBox"
      ItemsSource="{Binding AllCities}"
      Margin="12, 0, 12, 0" Width="440"
                        ItemTemplate="{StaticResource
ToDoListBoxItemTemplate}" />
                </controls:PivotItem>

  <controls:PivotItem Header="india">
   <ListBox
                        x:Name="homeToDoItemsListBox"
ItemsSource="{Binding IndianCity}"
Margin="12, 0, 12, 0" Width="440"
                        ItemTemplate="{StaticResource
ToDoListBoxItemTemplate}" />
    </controls:PivotItem>

   <controls:PivotItem Header="uk">
```

← ExpNo: DB2 (ExperimentTitle)

←Country Database (ApplicationTitle)

←Pivot Page Display having different headers.

←all (Pivot Header)

←india (Pivot Header)

←uk (Pivot Header)

```
  <ListBox
                      x:Name="workToDoItemsListBox"
    ItemsSource="{Binding UKCity}"
   Margin="12, 0, 12, 0" Width="440"
                        ItemTemplate="{StaticResource
ToDoListBoxItemTemplate}" />
    </controls:PivotItem>

   <controls:PivotItem Header="usa">
       <ListBox
                        x:Name="hobbiesToDoItemsListBox"
         ItemsSource="{Binding USACity}"
          Margin="12, 0, 12, 0" Width="440"
                        ItemTemplate="{StaticResource
ToDoListBoxItemTemplate}" />
     </controls:PivotItem>

           </controls:Pivot>
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True"
IsMenuEnabled="False">
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar.new.rest.png" Text="new"
x:Name="newTaskAppBarButton"
Click="newTaskAppBarButton_Click"/>
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar.feature.search.rest.png"
Text="search" x:Name="newSearchAppBarButton"
Click="newSearchAppBarButton_Click"/>

        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>

</phone:PhoneApplicationPage>
```

←usa (Pivot Header)

←new (ApplicationBarButton)

←search(ApplicationBarButton)

NewTaskPage.xaml

```
<!--LayoutRoot is the root grid where all page content is
placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application
and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0"
Margin="12,17,0,28">
<TextBlock x:Name="ApplicationTitle" Text="Country Database"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="city"
Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
```

←Country Database (ApplicationTitle)
←city (PageTitle)

```
        <Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="auto"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="100"/>
                <RowDefinition Height="300"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

                <TextBlock Text="City Name:" Width="100"
Margin="10,30,0,10" Grid.Column="0" Grid.Row="0"/>
                <TextBox x:Name="newTaskNameTextBox"
Height="100" Width="350" Grid.Column="1"  Grid.Row="0"
TextWrapping="Wrap"/>
            <TextBlock Text="Country" Width="100"
Margin="10,30,0,10" Grid.Column="0" Grid.Row="1"/>
            <toolkit:ListPicker
                x:Name="categoriesListPicker"
                ItemsSource="{Binding CountryList}"
                DisplayMemberPath="Name" Grid.Row="1"
Grid.Column="1">

            </toolkit:ListPicker>

        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True"
IsMenuEnabled="False">
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar.check.rest.png" Text="ok"
x:Name="appBarOkButton" Click="appBarOkButton_Click"/>
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar.cancel.rest.png" Text="cancel"
x:Name="appBarCancelButton"
Click="appBarCancelButton_Click"/>

        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>

</phone:PhoneApplicationPage>
```

← CountryList picker

←ok (ApplicationBarButton)

←cancel(ApplicationBarButton)

ItemSearch.xaml

```
 <!--LayoutRoot is the root grid where all page content is
placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application
and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0"
Margin="12,17,0,28">
 <TextBlock x:Name="ApplicationTitle" Text="Country Database"
```

←Country Database (ApplicationTitle)

```
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="search city"
Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0">
            <StackPanel Orientation="Vertical">
                <TextBlock Text="Search"
Style="{StaticResource PhoneTextGroupHeaderStyle}"/>
                <TextBox x:Name="itemToSearch" Height="100"
TextWrapping="Wrap" />
                <Button x:Name="SearchButton" Content="Go!"
Click="SearchButton_Click" Height="75" Width="120"
BorderBrush="Green" BorderThickness="5" Foreground="Wheat"/>
                <ListBox x:Name="responseList"
ScrollViewer.HorizontalScrollBarVisibility="Auto"
ScrollViewer.VerticalScrollBarVisibility="Auto">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <TextBlock Text="{Binding
Result}" Width="350" Height="100"/>
                            </StackPanel>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>
            </StackPanel>
        </Grid>
    </Grid>
```

←search city (PageTitle)

MainPage.xaml.cs

```
using System;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Phone.Controls;
using CountryDatabase.Model;

namespace CountryDatabase
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            this.DataContext = App.ViewModel;
        }
        private void newTaskAppBarButton_Click(object sender,
EventArgs e)
        {
            NavigationService.Navigate(new
Uri("/NewTaskPage.xaml", UriKind.Relative));
        }


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

←Adding namespaces

←NavigateTo NewTaskPage.xaml Page

```
            // Cast the parameter as a button.
            var button = sender as Button;

            if (button != null)
            {
                // Get a handle for the to-do item bound to
the button.
                City deleteCity = button.DataContext as City;

                App.ViewModel.DeleteCity(deleteCity);
            }

            // Put the focus back to the main page.
            this.Focus();
        }

        protected override void
OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs
e)
        {
            // Save changes to the database.
            App.ViewModel.SaveChangesTODB();
        }

        private void newSearchAppBarButton_Click(object
sender, EventArgs e)
        {
            NavigationService.Navigate(new
Uri("/ItemSearch.xaml", UriKind.Relative));
        }

    }
}
```

←NavigateTo ItemSearch.xaml Page

NewTaskPage.xaml.cs

```
using System;
using Microsoft.Phone.Controls;
using CountryDatabase.Model;

namespace CountryDatabase
{
    public partial class NewTaskPage : PhoneApplicationPage
    {
        public NewTaskPage()
        {
            InitializeComponent();
            this.DataContext = App.ViewModel;
        }
        private void appBarOkButton_Click(object sender,
EventArgs e)
        {
            // Confirm there is some text in the text box.
            if (newTaskNameTextBox.Text.Length > 0)
            {
                // Create a new to-do item.
                City newToDoItem = new City
                {
                    CityName = newTaskNameTextBox.Text,
                    country =
(Country)categoriesListPicker.SelectedItem
                };
```

←Add namespaces

```
                // Add the item to the ViewModel.
                App.ViewModel.AddCity(newToDoItem);

                // Return to the main page.
                if (NavigationService.CanGoBack)
                {
                    NavigationService.GoBack();
                }
            }
        }

        private void appBarCancelButton_Click(object sender,
EventArgs e)
        {
            // Return to the main page.
            if (NavigationService.CanGoBack)
            {
                NavigationService.GoBack();
            }
        }
    }
}
```

ItemSearch.xaml.cs

← Add namespaces

```
using System;
using Microsoft.Phone.Controls;
using CountryDatabase.Model;

namespace CountryDatabase
{
    public partial class NewTaskPage : PhoneApplicationPage
    {
        public NewTaskPage()
        {
            InitializeComponent();
            this.DataContext = App.ViewModel;
        }
        private void appBarOkButton_Click(object sender,
EventArgs e)
        {
            // Confirm there is some text in the text box.
            if (newTaskNameTextBox.Text.Length > 0)
            {
                // Create a new to-do item.
                City newToDoItem = new City
                {
                    CityName = newTaskNameTextBox.Text,
                    country =
(Country)categoriesListPicker.SelectedItem
                };

                // Add the item to the ViewModel.
                App.ViewModel.AddCity(newToDoItem);

                // Return to the main page.
                if (NavigationService.CanGoBack)
                {
                    NavigationService.GoBack();
                }
            }
        }
```

←Add new item for database

```
        private void appBarCancelButton_Click(object sender,
EventArgs e)
        {
            // Return to the main page.
            if (NavigationService.CanGoBack)
            {
                NavigationService.GoBack();
            }
        }
    }
}
```

ViewModel/CountryViewModel.cs

←ViewModel Class

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.ComponentModel;
using CountryDatabase.Model;
```

← Add namespaces

```
namespace CountryDatabase.ViewModel {
    public class CountryViewModel : INotifyPropertyChanged {
        private CountryDataContext CountryDB;
        public CountryViewModel(string toDBConnectionString)
{
            CountryDB = new
CountryDataContext(toDBConnectionString);
        }
        public void SaveChangesTODB() {
            CountryDB.SubmitChanges();
        }
        #region INotifyPropertyChanges Members
        public event PropertyChangedEventHandler
PropertyChanged;
        private void NotifyPropertyChanged(string
propertyName) {
            if (PropertyChanged != null) {
                PropertyChanged(this,new
PropertyChangedEventArgs(propertyName));
            }
        }
        #endregion
        private ObservableCollection<City> _allCities;
        public ObservableCollection<City> AllCities {
            get { return _allCities; }
            set { _allCities = value;
            NotifyPropertyChanged("All Cities");
            }
        }
        private ObservableCollection<City> _indianCity;
        public ObservableCollection<City> IndianCity {
            get { return _indianCity; }
            set { _indianCity = value;
            NotifyPropertyChanged("Indian City");
}
        }
        private ObservableCollection<City> _ukCity;
        public ObservableCollection<City> UKCity {
            get { return _ukCity; }
            set { _ukCity = value;
            NotifyPropertyChanged("UK City");
```

←Implement INotifyPropertyChanged interface

←Initialise connectionString object

```
            }
        }
        private ObservableCollection<City> _usaCity;
        public ObservableCollection<City> USACity {
            get { return _usaCity; }
            set { _usaCity = value;
            NotifyPropertyChanged("USA City");
            }
        }
        private List<Country> _countryList;
        public List<Country> CountryList {
            get { return _countryList; }
            set { _countryList = value;
            NotifyPropertyChanged("Country List");
            }

        }
        public void LoadCollectionFromDatabase(){
            var citiesInDB = from City c in CountryDB.Cities
select c;
            AllCities = new
ObservableCollection<City>(citiesInDB);

            var countriesInDB = from Country ct in
CountryDB.Countries select ct;
            foreach(Country country in countriesInDB){
            switch(country.Name){
                case "India": IndianCity = new
ObservableCollection<City>(country.city);
                break;
                case "UK": UKCity = new
ObservableCollection<City>(country.city);
                break;
                case "USA": USACity = new
ObservableCollection<City>(country.city);
                break;
                default:
                break;
            }
            }
            CountryList = CountryDB.Countries.ToList();
        }
        public void AddCity(City newCity) {

            CountryDB.Cities.InsertOnSubmit(newCity);

            CountryDB.SubmitChanges();
            AllCities.Add(newCity);
            switch (newCity.country.Name) {
                case "India": IndianCity.Add(newCity);
                break;
                case "UK": UKCity.Add(newCity);
                break;
                case "USA": USACity.Add(newCity);
                break;
                default:
                break;

            }
        }
        public bool SearchCity(string qCity) {
```

←Adding new city to the database

```
            var query = from item in CountryDB.Cities where
item.CityName.Contains(qCity) select item.CityName;

            if (query.FirstOrDefault()!= null) {
                return true;
            }
            return false;

        }
        public void DeleteCity(City delCity) {
            AllCities.Remove(delCity);
            CountryDB.Cities.DeleteOnSubmit(delCity);
            switch (delCity.country.Name) {
                case "India": IndianCity.Remove(delCity);
                break;
                case "UK": UKCity.Remove(delCity);
                break;
                case "USA": USACity.Remove(delCity);
                break;
                default:
                break;
            }
            CountryDB.SubmitChanges();
        }
    }
}
```

← Deletion of the city from the database.

Model/CountryDataContext.cs

Model Class

```
using System;
using System.Data.Linq.Mapping;
using System.Data.Linq;
using System.ComponentModel;
```

←Add namespace

```
namespace CountryDatabase.Model {
    public class CountryDataContext : DataContext {
        public CountryDataContext(string connectionString) :
base(connectionString) { }
        public Table<City> Cities;
        public Table<Country> Countries;
    }

    [Table]
    public class Country: INotifyPropertyChanged {

        private string name;
        [Column]
        public string Name {
            get { return name;}
            set
            {
                if (name != value)
                {
                    name = value;
                    NotifyPropertyChanged("Name");

                }
            }
        }
        private int id;
        [Column(IsPrimaryKey = true, CanBeNull = false,
IsDbGenerated = true, DbType = "INT NOT NULL IDENTITY",
AutoSync = AutoSync.OnInsert)]
```

←Implement DataContext interface

←Two tables namely, City and Country.

←Implement INotifyPropertyChanged interface

```
        public int ID {
            get { return id; }
            set
            {
                if (id != value)
                {
                    id = value;
                    NotifyPropertyChanged("ID");
                }

            }
        }

        private EntitySet<City> _city;
        [Association(Storage = "_city", OtherKey =
"_countryId", ThisKey = "ID")]
        public EntitySet<City> city {
            get { return this._city; }
            set
            {
                this._city.Assign(value);
            }
        }
        public Country(){
            _city = new EntitySet<City>(new
Action<City>(this.attach_city),
                    new Action<City>(this.dettach_city));
    }
        private void attach_city(City cities) {
            NotifyPropertyChanged("City");
            cities.country = this;
        }
        private void dettach_city(City cities) {
            NotifyPropertyChanged("City");
            cities.country = null;
        }
         #region INotifyPropertyChanged Members

        public event PropertyChangedEventHandler
PropertyChanged;

        private void NotifyPropertyChanged(string
propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
            }
        }
        #endregion

        #region INotifyPropertyChanging Members
        public event PropertyChangingEventHandler
PropertyChanging;
        private void NotifyPropertyChaning(string
propertyName) {
            if (PropertyChanging != null) {
                PropertyChanging(this, new
PropertyChangingEventArgs(propertyName));
            }
        }
```

```
        #endregion
    }
    [Table]
    public class City:
INotifyPropertyChanged,INotifyPropertyChanging{



        private int _cityId;

        [Column(IsPrimaryKey = true, IsDbGenerated = true,
DbType = "INT NOT NULL Identity", CanBeNull = false, AutoSync
= AutoSync.OnInsert)]
        public int CityID
        {
            get { return _cityId; }
            set
            {
                if (_cityId != value)
                {
                    NotifyPropertyChanging("CityID");
                    _cityId = value;
                    NotifyPropertyChanged("CityID");
                }
            }
        }

        private string _cityName;
        [Column]
        public string CityName {
            get { return _cityName; }
            set {
                if (_cityName != value)
                {
                    _cityName = value;
                    NotifyPropertyChanged("CityName");

                }
            }
        }

        [Column]
        internal int _countryId;
        private EntityRef<Country> _country;
        [Association(Storage = "_country", ThisKey =
"_countryId", OtherKey = "ID", IsForeignKey = true)]
        public Country country {
            get { return _country.Entity; }
            set { NotifyPropertyChanged("Country");
            _country.Entity = value;
            if (value != null) {
                _countryId = value.ID;
            }
            }
        }

        #region INotifyPropertyChanged Members

        public event PropertyChangedEventHandler
PropertyChanged;

        private void NotifyPropertyChanged(string
```

←Table City

```
propertyName)
        {
            if (PropertyChanged != null)
            {
                PropertyChanged(this, new
PropertyChangedEventArgs(propertyName));
            }
        }
        #endregion
        #region INotifyPropertyChanging Members
        public event PropertyChangingEventHandler
PropertyChanging;
        private void NotifyPropertyChanging(string
propertyName)
        {
            if (PropertyChanging != null)
            {
                PropertyChanging(this, new
PropertyChangingEventArgs(propertyName));
            }
        }
        #endregion
    }
}
```

Screenshots:

Fig. No.1 Home Page

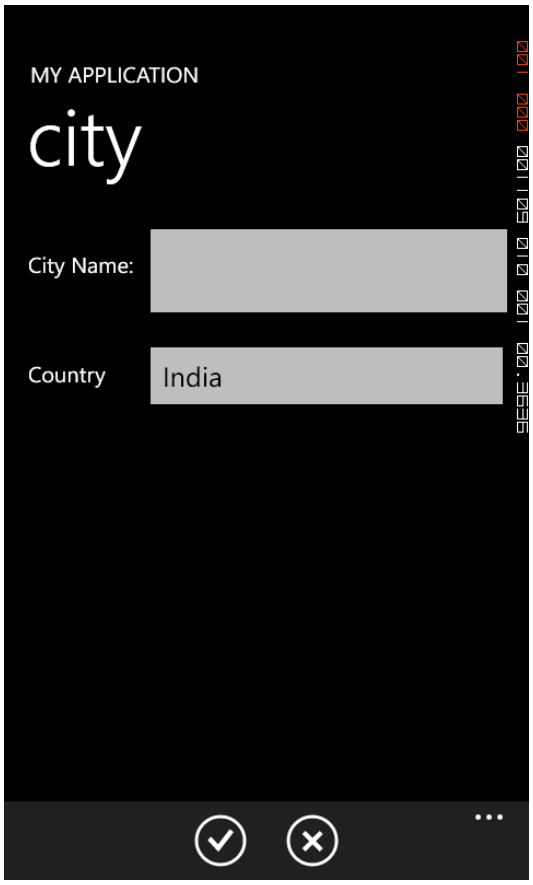

Fig. No.2 Add City Name

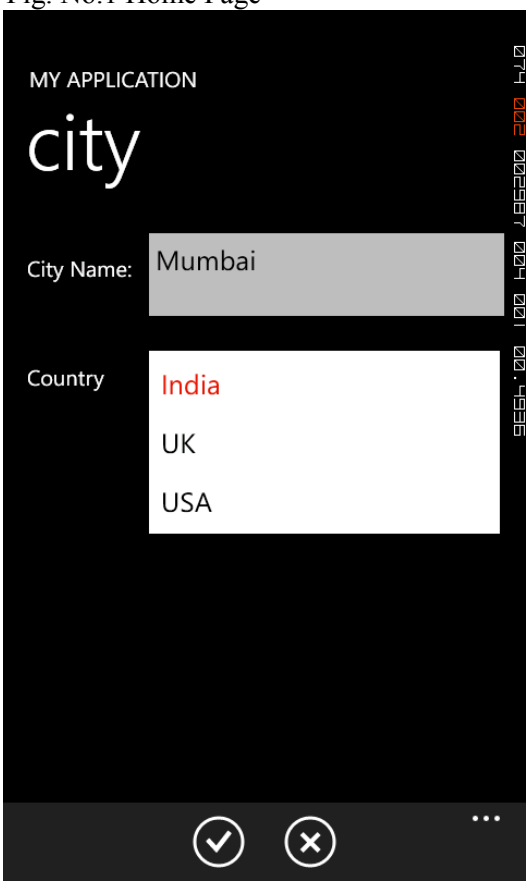

Fig. No.3 Shows the Country Dropdown List

Fig. No.4 Cities Added to corresponding country

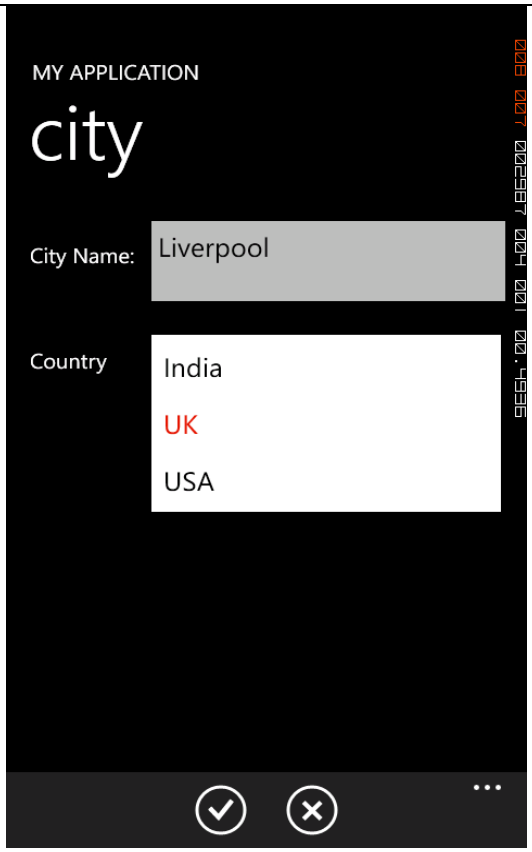

Fig. No.5 Adding new City under UK

Fig. No.6 Liverpool is added with corresponding country

Fig. No.7 Adding new city under USA

Fig. No.8 New York is added with corresponding country

Fig. No.9 All cities are displaying

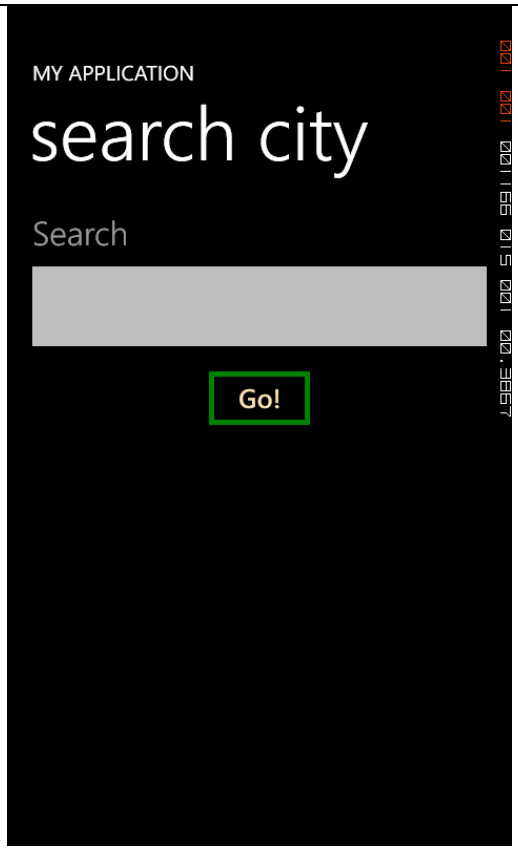

Fig. No.10 Search City

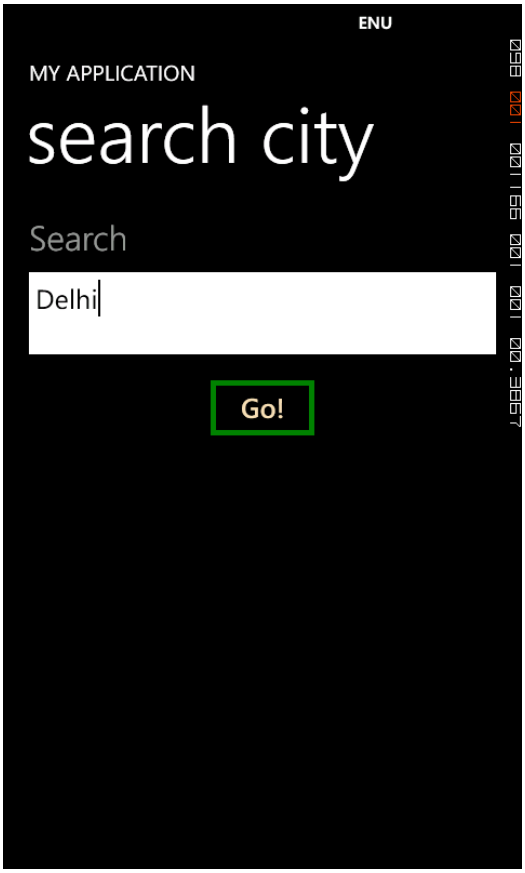

Fig. No.11 Type name of city to be search

Fig. No.12 Message box showing the response

Fig. no 13 Message box showing the response of the query

Observations: It is observed by developer, that MVVM architecture is very flexible to use in Windows Phone during database management. It enhances the response for querying to the database in many folds.