

e-Project Documentation

**DEVELOPED AND DESINGED BY:**

* Waqar Ahmed
* Ali haider
* Ahsan Iqbal

Batch Code: iPRO-201312f

**CONTENTS**

**ACKNOWLEDGMENT……………………………..……..………………...….3**

**Preface** ………………...……………………………………………….……….4

**Certificate of Completion.**……………………….………………….….….…5

**Introduction**………………………………………………………….…...…….6

**Problems Statement** ………..………………………..……….……………...8

**Objectives of the project** …………………………..………………………...9

**REQUIREMENTS SPECIFICATION…………..………………………………14**

**Hardware Requirements**…………………………………………….………...14

**Software Requirements**…………………………...……………….………….15

**Guides……………………………………..………………………….………….13**

Users Guide………………………………………………………………………16

Developers Guide………………………………………………………………..17

**TASK SHEET……………………………..………………………….………….18**

**Source code With Comments**………………………………………………………………...……..19

**Snapshots**………………………………………………………………...……..35

**ACKNOWLEDGEMENT**

We are thankful to almighty ALLAH that by his grace we have been successful. We have tried our best to take all relevant important topics in consideration so that this project may become comprehensive.

This is our **Third Semester** Project. We have tried our best & struggled for a better outlook to come. This Project has been developed in **Visual studio 2013**. We have been assigned to make Application of **Medicine Guide** as a Project. We find easy to do coding in **Visual Studio 2013**

We are hopeful that this project will provide all the necessarily information required by the user to fulfill their inspiration. However suggestions and advices for improvement of this project will be thankfully received & acknowledged.

“THANK YOU”

**PREFACE**

First of all, we would like to thank Allah the almighty who gave us mind to think and heart to feel creativity. Creations depend on thinking.

Many people contributed to this Project and we would like to thank all of them. I guess we shouldn’t be able to complete this Project without the help of our teacher **Sir Ali hyder**.

Very special thanks to all our teachers who gave us suggestions and always encourage us to do the best. In deed class fellows are very important we would like to thank all of them for their kind appreciation, encouragement and moral support.

And finally would like to express our deepest gratitude to our parents whose prayers were always encouraging to us.

**CERTIFICATE OF COMPLETION**

This is to certify that **Waqar ahmed, Ali haider** and **Ahsan iqbal** have successfully completed the development of **Medicine guide** and submitted to Aptech Computer Education,Metro Stargate Shahra-e-Faisal, Karachi.

Yours Truly,

Date: - 1st September, 2015 Waqar ahmed

**INTRODUCTION**

The thirst for learning, upgrading technical skills and applying the concepts in real life environment at a fast pace is what the industry demands from IT professionals today. However busy work schedules, far-flung locations, and unavailability of convenient time-slots pose as major barriers when it comes to applying the concepts into realism. And hence the need to look out for alternative means of implementation in the form of laddered approach.

The above truly pose as constraints especially for our students too! With their busy schedules, it is indeed difficult for our students to keep up with the genuine and constant need for integrated application which can be seen live especially so in the field of IT education where technology can change on the spur of a moment. Well, technology does come to our rescue at such times!!

Keeping the above in mind and in tune with our constant endeavour to use Technology in our training model, we at Aptech have thought of revolutionizing the way our students learn and implement the concepts using tools themselves by providing a live and synchronous IProjects learning environment!

**Problem Statement**

**Write a Windows APP for Medicine Guide.**

The Medicine Guide app is the ready reckoner to understand disease and their cure. Inside app will have the information of various diseases and cure available. It will provide a list of medicines available for curing the disease.

App should have proper listing of disease and medicines in alphabetical order.

Medicine Guide will help to find the understand diseases and their remedies

**Objectives of the project**

You need to complete your project and submit the following by the end-date.

I.      Working Application

a.       Source Code

b.      Compiled Code

II.      eProject Report

n       eProject Report should comprise-

q      Acknowledgements

q      eProject Synopsis

q      eProject Analysis

q      eProject Design

n       DFD’s

n       FlowCharts

n       Process Diagrams

n       Database Design / Structure

q      Screen Shot’s

q      Source Code with Comments

q      User Guide

q      Developer’s Guide

n       Module Descriptions

* User Guide means User Manual, it should have details how a user can operate the application.
* Developer’s Guide should have detail description of the modules, so that any developer can look into the code and get to know about it through the Developer’s Guide.

Some of the benefits of eProject are as under: -

1.    Re-enforcement of Skills happens in the experiential learning process.

2.    A mentor, ensuring that you do not get drifted, constantly guides you.

3.    It gives you a lot of confidence to face an interview as you have worked on a project. You can explain virtually everything on the subject you have learnt.

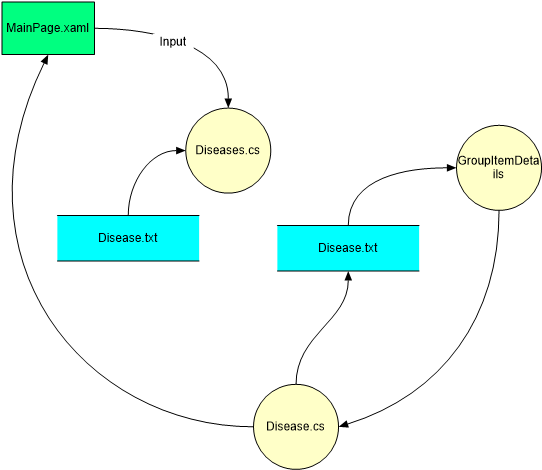
4.    A nice, final, black, hardbound report adds to you credentials and enhances your job prospects.

eProjects is compulsory (Sem-End Project) and is supposed to be done over and above the assignments given in your course books.

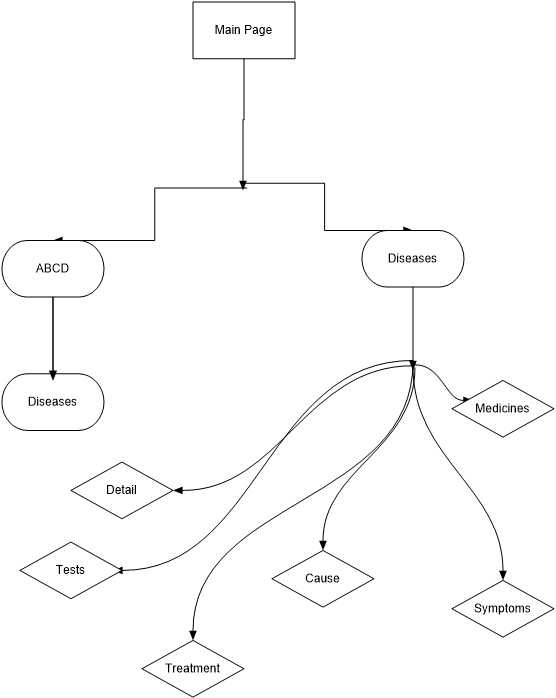
It is advised that if a student follows the projects and submits assignments he would definitely have an edge over other students.

Students need to submit their project report along with working application and source code in the stipulated time.

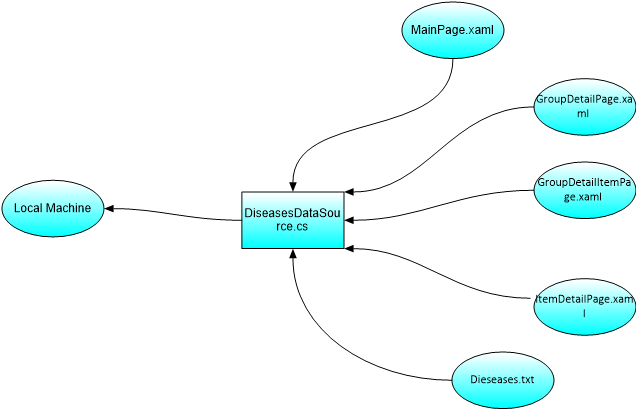
Projects will be evaluated.

DFD  


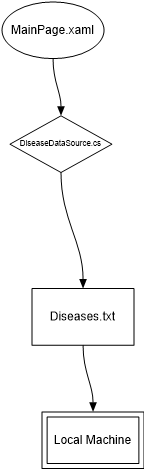
**Flow Chart**



**Process diagram**



**Data Base Design**



**Hardware / Software Requirements**

**Hardware**

* A minimum computer system that will help you access all the tools in the courses is a Pentium 166 or better
* 128 Megabytes of RAM or better
* Windows 2000 Server (or higher if possible)

**Software**

Use software as per your requirement

* Win-8/Visual Studio .net/C#/Notepad/SQL

**USER GUIDE**

|  |  |  |
| --- | --- | --- |
| **No.** | **Items** | **Description** |
| 1 | Operating System | Windows 2000 Server |
| 2 | Main processor | Minimum 166 or better |
| 3 | Ram (Memory) | Minimum 128 or better |
| 4 | Software | Win-8/Visual Studio .net/C#/Notepad/SQL |

System Requirement:

Copy Medicine Guide folder on your hard disk:

**For User:**

**For Visual Studio:**

Open Visual studio 2013 and Open Project file.

**Description:**

* Now go to solution explorer on the top left side of your screen choose main page.xaml
* The main page will apear
* Now go to third row from top center of the page and run it execute the project on local machine
* Then the main gridview of app will apear
* Go to ABCD or any of your desired section of the page
* There are diseases oraganized Alphabatically
* Click on disease according to your problem with the help of image
* Now know about the disease and its Symptoms, Cuases, Tests, Treatment and Medicines

**DEVELOPER GUIDE**

* Main page.xaml is for Gridview and design
* GroupDetailPage.xaml is for design of group details
* GroupDetailItemPage.xaml is for Design of Group item details
* diseaseDataSource.cs is for C# band hand coding
* Diseases.txt is Getting all the data of Diseases

**TASK SHEET**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project No.** | **Project Title** | **Activiy Plan Prepared By:** | **Date of Prepration Activity Plan** | |
| 01 | Medicine Guide | Waqar ahmed |
| **S. No.** | **Tasks** | **Start Date** | **Team Mate Name** | **Status** |
| 1. | Analysis | 16 aug, 15 | All Members | Done |
| 2. | Data Searching | 18 aug, 15 | Ahsan iqbal and Waqar ahmed | Done |
| 4. | Designed By | 20 aug, 15 | Ali haider | Done |
| 5. | Developed By | 22 aug, 15 | Ali haider and Waqar ahmed | Done |
| 6. | Testing By | 30 aug, 15 | Ahsan iqbal | Done |
| 7. | Documentation By | 1st sep, 15 | Waqar ahmed | Done |
| 8. | Presentation | 2nd sep, 15 | All members | Done |
| 9. | Final Testing | 2nd sep, 15 | All Members | Done |
| 10 | Submission | 6 sep, 15 | All Members | Done |

**Source code with comments**

using System;

using System.Linq;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using Windows.ApplicationModel.Resources.Core;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Media.Imaging;

using System.Net.Http;

using Windows.Data.Json;

using Windows.ApplicationModel;

using Windows.Storage.Streams;

using System.Threading.Tasks;

using Windows.Storage;

// The data model defined by this file serves as a representative example of a strongly-typed

// model that supports notification when members are added, removed, or modified. The property

// names chosen coincide with data bindings in the standard item templates.

//

// Applications may use this model as a starting point and build on it, or discard it entirely and

// replace it with something appropriate to their needs.

namespace MedicineGuide.Data

{

/// <summary>

/// Base class for <see cref="diseaseDataItem"/> and <see cref="diseaseDataGroup"/> that

/// defines properties common to both.

/// </summary>

[Windows.Foundation.Metadata.WebHostHidden]

public abstract class diseaseDataCommon : MedicineGuide.Common.BindableBase

{

internal static Uri \_baseUri = new Uri("ms-appx:///");

public diseaseDataCommon(String uniqueId, String title, String shortTitle, String imagePath)

{

this.\_uniqueId = uniqueId;

this.\_title = title;

this.\_shortTitle = shortTitle;

this.\_imagePath = imagePath;

}

private string \_uniqueId = string.Empty;

public string UniqueId

{

get { return this.\_uniqueId; }

set { this.SetProperty(ref this.\_uniqueId, value); }

}

private string \_title = string.Empty;

public string Title

{

get { return this.\_title; }

set { this.SetProperty(ref this.\_title, value); }

}

private string \_shortTitle = string.Empty;

public string ShortTitle

{

get { return this.\_shortTitle; }

set { this.SetProperty(ref this.\_shortTitle, value); }

}

private ImageSource \_image = null;

private String \_imagePath = null;

public Uri ImagePath

{

get

{

return new Uri(diseaseDataCommon.\_baseUri, this.\_imagePath);

}

}

public ImageSource Image

{

get

{

if (this.\_image == null && this.\_imagePath != null)

{

this.\_image = new BitmapImage(new Uri(diseaseDataCommon.\_baseUri, this.\_imagePath));

}

return this.\_image;

}

set

{

this.\_imagePath = null;

this.SetProperty(ref this.\_image, value);

}

}

public void SetImage(String path)

{

this.\_image = null;

this.\_imagePath = path;

this.OnPropertyChanged("Image");

}

public string GetImageUri()

{

return \_imagePath;

}

}

/// <summary>

/// disease item data model.

/// </summary>

public class diseaseDataItem : diseaseDataCommon

{

public diseaseDataItem()

: base(String.Empty, String.Empty, String.Empty, String.Empty)

{

}

public diseaseDataItem(String uniqueId, String title, String shortTitle, String imagePath, String treatments, ObservableCollection<string> medicines, diseaseDataGroup group)

: base(uniqueId, title, shortTitle, imagePath)

{

this.\_treatments = treatments;

this.\_medicines = medicines;

this.\_group = group;

}

private string \_treatments = string.Empty;

public string treatments

{

get { return this.\_treatments; }

set { this.SetProperty(ref this.\_treatments, value); }

}

private ObservableCollection<string> \_medicines;

public ObservableCollection<string> medicines

{

get { return this.\_medicines; }

set { this.SetProperty(ref this.\_medicines, value); }

}

private diseaseDataGroup \_group;

public diseaseDataGroup Group

{

get { return this.\_group; }

set { this.SetProperty(ref this.\_group, value); }

}

private ImageSource \_tileImage;

private string \_tileImagePath;

public Uri TileImagePath

{

get

{

return new Uri(diseaseDataCommon.\_baseUri, this.\_tileImagePath);

}

}

public ImageSource TileImage

{

get

{

if (this.\_tileImage == null && this.\_tileImagePath != null)

{

this.\_tileImage = new BitmapImage(new Uri(diseaseDataCommon.\_baseUri, this.\_tileImagePath));

}

return this.\_tileImage;

}

set

{

this.\_tileImagePath = null;

this.SetProperty(ref this.\_tileImage, value);

}

}

public void SetTileImage(String path)

{

this.\_tileImage = null;

this.\_tileImagePath = path;

this.OnPropertyChanged("TileImage");

}

}

/// <summary>

/// disease group data model.

/// </summary>

public class diseaseDataGroup : diseaseDataCommon

{

public diseaseDataGroup()

: base(String.Empty, String.Empty, String.Empty, String.Empty)

{

}

public diseaseDataGroup(String uniqueId, String title, String shortTitle, String imagePath, String description)

: base(uniqueId, title, shortTitle, imagePath)

{

}

private ObservableCollection<diseaseDataItem> \_items = new ObservableCollection<diseaseDataItem>();

public ObservableCollection<diseaseDataItem> Items

{

get { return this.\_items; }

}

public IEnumerable<diseaseDataItem> TopItems

{

// Provides a subset of the full items collection to bind to from a GroupedItemsPage

// for two reasons: GridView will not virtualize large items collections, and it

// improves the user experience when browsing through groups with large numbers of

// items.

//

// A maximum of 12 items are displayed because it results in filled grid columns

// whether there are 1, 2, 3, 4, or 6 rows displayed

get { return this.\_items.Take(12); }

}

private string \_description = string.Empty;

public string Description

{

get { return this.\_description; }

set { this.SetProperty(ref this.\_description, value); }

}

private ImageSource \_groupImage;

private string \_groupImagePath;

public ImageSource GroupImage

{

get

{

if (this.\_groupImage == null && this.\_groupImagePath != null)

{

this.\_groupImage = new BitmapImage(new Uri(diseaseDataCommon.\_baseUri, this.\_groupImagePath));

}

return this.\_groupImage;

}

set

{

this.\_groupImagePath = null;

this.SetProperty(ref this.\_groupImage, value);

}

}

public int diseasesCount

{

get

{

return this.Items.Count;

}

}

public void SetGroupImage(String path)

{

this.\_groupImage = null;

this.\_groupImagePath = path;

this.OnPropertyChanged("GroupImage");

}

}

/// <summary>

/// Creates a collection of groups and items.

/// </summary>

public sealed class diseaseDataSource

{

//public event EventHandler diseasesLoaded;

private static diseaseDataSource \_diseaseDataSource = new diseaseDataSource();

private ObservableCollection<diseaseDataGroup> \_allGroups = new ObservableCollection<diseaseDataGroup>();

public ObservableCollection<diseaseDataGroup> AllGroups

{

get { return this.\_allGroups; }

}

public static IEnumerable<diseaseDataGroup> GetGroups(string uniqueId)

{

if (!uniqueId.Equals("AllGroups")) throw new ArgumentException("Only 'AllGroups' is supported as a collection of groups");

return \_diseaseDataSource.AllGroups;

}

public static diseaseDataGroup GetGroup(string uniqueId)

{

// Simple linear search is acceptable for small data sets

var matches = \_diseaseDataSource.AllGroups.Where((group) => group.UniqueId.Equals(uniqueId));

if (matches.Count() == 1) return matches.First();

return null;

}

public static diseaseDataItem GetItem(string uniqueId)

{

// Simple linear search is acceptable for small data sets

var matches = \_diseaseDataSource.AllGroups.SelectMany(group => group.Items).Where((item) => item.UniqueId.Equals(uniqueId));

if (matches.Count() == 1) return matches.First();

return null;

}

public static async Task LoadRemoteDataAsync()

{

// Retrieve disease data from Azure

var client = new HttpClient();

client.MaxResponseContentBufferSize = 1024 \* 1024; // Read up to 1 MB of data

var response = await client.GetAsync(new Uri("http://contosodiseases8.blob.core.windows.net/AzurediseasesRP"));

var result = await response.Content.ReadAsStringAsync();

// Parse the JSON disease data

var diseases = JsonArray.Parse(result);

// Convert the JSON objects into diseaseDataItems and diseaseDataGroups

CreatediseasesAnddiseaseGroups(diseases);

}

public static async Task LoadLocalDataAsync()

{

// Retrieve disease data from diseases.txt

var file = await Package.Current.InstalledLocation.GetFileAsync("Data\\diseases.txt");

var result = await FileIO.ReadTextAsync(file);

// Parse the JSON disease data

var diseases = JsonArray.Parse(result);

// Convert the JSON objects into diseaseDataItems and diseaseDataGroups

CreatediseasesAnddiseaseGroups(diseases);

}

private static void CreatediseasesAnddiseaseGroups(JsonArray array)

{

foreach (var item in array)

{

var obj = item.GetObject();

diseaseDataItem disease = new diseaseDataItem();

diseaseDataGroup group = null;

foreach (var key in obj.Keys)

{

IJsonValue val;

if (!obj.TryGetValue(key, out val))

continue;

switch (key)

{

case "key":

disease.UniqueId = val.GetNumber().ToString();

break;

case "title":

disease.Title = val.GetString();

break;

case "shortTitle":

disease.ShortTitle = val.GetString();

break;

case "treatments":

disease.treatments = val.GetString();

break;

case "medicines":

var medicines = val.GetArray();

var list = (from i in medicines select i.GetString()).ToList();

disease.medicines = new ObservableCollection<string>(list);

break;

case "backgroundImage":

disease.SetImage(val.GetString());

break;

case "tileImage":

disease.SetTileImage(val.GetString());

break;

case "group":

var diseaseGroup = val.GetObject();

IJsonValue groupKey;

if (!diseaseGroup.TryGetValue("key", out groupKey))

continue;

group = \_diseaseDataSource.AllGroups.FirstOrDefault(c => c.UniqueId.Equals(groupKey.GetString()));

if (group == null)

group = CreatediseaseGroup(diseaseGroup);

disease.Group = group;

break;

}

}

if (group != null)

group.Items.Add(disease);

}

}

private static diseaseDataGroup CreatediseaseGroup(JsonObject obj)

{

diseaseDataGroup group = new diseaseDataGroup();

foreach (var key in obj.Keys)

{

IJsonValue val;

if (!obj.TryGetValue(key, out val))

continue;

switch (key)

{

case "key":

group.UniqueId = val.GetString();

break;

case "title":

group.Title = val.GetString();

break;

case "shortTitle":

group.ShortTitle = val.GetString();

break;

case "description":

group.Description = val.GetString();

break;

case "backgroundImage":

group.SetImage(val.GetString());

break;

case "groupImage" :

group.SetGroupImage(val.GetString());

break;

}

}

\_diseaseDataSource.AllGroups.Add(group);

return group;

}

}

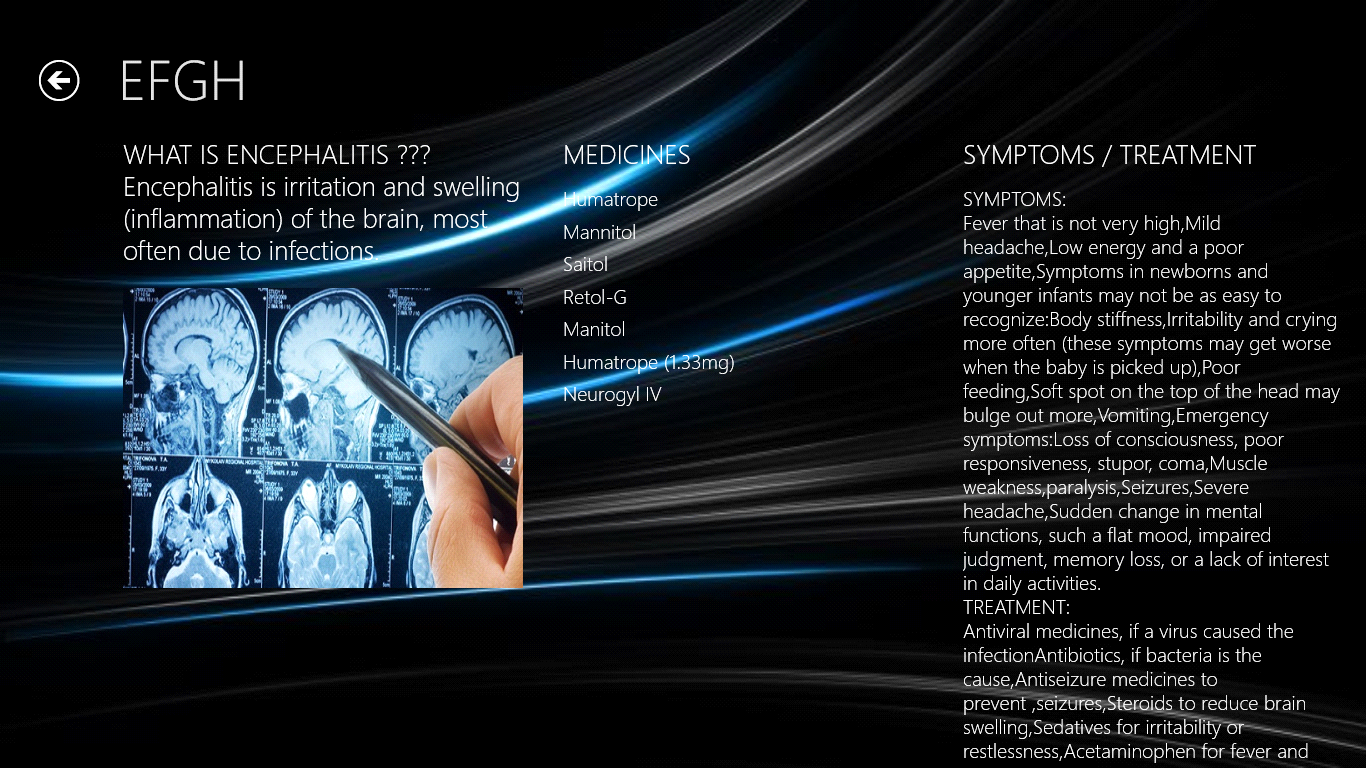
}

**Snaps Shorts  
MainPage**

 **DetailPage**



**MedicinePage**



Best Regards from Our Team, Thank you

………………………………………………………………………