**9. Testing Approach (in terms of both functional and non-functional requirements)**

The initial testing approach of the XML Reader was to run it as a stand-alone prototype and print out the return variable and verify it was a list of movies and shooting locations. The following code took care of the printing out of the variable.

foreach (var loc in locations)

{

Console.WriteLine("Film Index " + loc.index);

Console.WriteLine("Film Title " + loc.filmTitle);

foreach (var sLoc in loc.locn)

{

Console.WriteLine("Locn Index " + sLoc.index);

Console.WriteLine("Locn Display Text " + sLoc.locnText);

Console.WriteLine("Locn lat " + sLoc.latCoord);

Console.WriteLine("Locn lng " + sLoc.lngCoord);

Console.WriteLine("Locn Radius " + sLoc.radius);

}

// Console.ReadLine();

}

}

Console.ReadLine();

The main XML Reader class could be slotted in into this prototype for debugging if it stopped working in the main application. The following is an excerpt of the start of the XML output to screen.

Film Index 0

Film Title \*batteries not included

Locn Index 0

Locn Display Text E. 5th St.<br>East Village<br>Manhattan

Locn lat 40.7224452961828

Locn lng -73.9786505699157

Locn Radius 0

Film Index 1

Film Title 12 Angry Men

Locn Index 0

Locn Display Text New York County Courthouse<br>40 Foley Square<br>Lower Manhattan

Locn lat 40.7137

Locn lng -74.0079

Locn Radius 0

Film Index 2

Film Title 13 Going on 30

Locn Index 0

Locn Display Text W. 47th St. and Seventh Ave.<br>Times Square<br>Manhattan

Locn lat 40.7592204876521

Locn lng -73.9846211671829

Locn Radius 0

The following is an excerpt from the end of the XML data.

Film Index 176

Film Title You're a Big Boy Now

Locn Index 0

Locn Display Text Steeplechase Park<br>Coney Island<br>Brooklyn

Locn lat 40.574725662771

Locn lng -73.9803296327591

Locn Radius 0

Locn Index 1

Locn Display Text New York Public Library<br>Fifth Ave. and 41st St.<br>Manhattan

Locn lat 40.7528651331099

Locn lng -73.9815366268158

Locn Radius 0

Film Index 177

Film Title You've Got Mail

Locn Index 0

Locn Display Text Verdi Square<br>W. 73rd St. & Broadway<br>Manhattan

Locn lat 40.7792578185718

Locn lng -73.9815366268158

Locn Radius 0

Locn Index 1

Locn Display Text 91st Street Community Garden<br>Riverside Park<br>Manhattan

Locn lat 40.7928238690811

Locn lng -73.9774167537689

Locn Radius 0

This data provided information for further unit testing of the XML Reader and code which called it. addEntry which takes care of adding a film and one location, has such a unit test. The unit test is as follows:

public void addEntryTest()

{

LocnXMLReader\_Accessor target = new LocnXMLReader\_Accessor();

string filmName = "Die Hard";

double latCoord = 30.5;

double lngCoord = -70.25;

string locnDisplayText = "Manhattan";

target.addEntry(filmName, latCoord, lngCoord, locnDisplayText);

Assert.AreEqual(target.filmLocations.Count, 1);

Assert.AreEqual(target.filmLocations[0].filmTitle, "Die Hard");

Assert.AreEqual(target.filmLocations[0].locations.Count, 1);

Assert.AreEqual(target.filmLocations[0].locations[0].point.x, 30.5);

Assert.AreEqual(target.filmLocations[0].locations[0].point.y, -70.25);

Assert.AreEqual(target.filmLocations[0].locations[0].locnText, "Manhattan");

filmName = "Die Hard";

latCoord = 25.5;

lngCoord = -55.25;

locnDisplayText = "Brooklyn";

target.addEntry(filmName, latCoord, lngCoord, locnDisplayText);

Assert.AreEqual(target.filmLocations.Count, 1);

Assert.AreEqual(target.filmLocations[0].filmTitle, "Die Hard");

Assert.AreEqual(target.filmLocations[0].locations.Count, 2);

Assert.AreEqual(target.filmLocations[0].locations[0].point.x, 30.5);

Assert.AreEqual(target.filmLocations[0].locations[0].point.y, -70.25);

Assert.AreEqual(target.filmLocations[0].locations[0].locnText, "Manhattan");

Assert.AreEqual(target.filmLocations[0].locations[1].point.x, 25.5);

Assert.AreEqual(target.filmLocations[0].locations[1].point.y, -55.25);

Assert.AreEqual(target.filmLocations[0].locations[1].locnText, "Brooklyn");

filmName = "Die Hard 2";

latCoord = 22.75;

lngCoord = -66.66666666667;

locnDisplayText = "Queens";

target.addEntry(filmName, latCoord, lngCoord, locnDisplayText);

Assert.AreEqual(target.filmLocations.Count, 2);

Assert.AreEqual(target.filmLocations[1].filmTitle, "Die Hard 2");

Assert.AreEqual(target.filmLocations[1].locations.Count, 1);

Assert.AreEqual(target.filmLocations[1].locations[0].point.x, 22.75);

Assert.AreEqual(target.filmLocations[1].locations[0].point.y, -66.66666666667);

Assert.AreEqual(target.filmLocations[1].locations[0].locnText, "Queens");

}

This unit test is especially necessary as the function was changed to interface directly with the Data Access Layer and thereby indirectly to the database. The View does not access the XML Reader but accesses the database via the Business Logic and Data Access Layers. This unit test will guard against further changes to the program breaking addEntry.

The LocationFinder has also a straight forward unit test based on the XML data read by the stand-alone prototype. The unit test is as follows:

public void getLocationsForFilmTest()

{

LocationFinder target = new LocationFinder();

String filmName;

LocationListUI expected, actual;

filmName = "12 Angry Men";

expected = new LocationListUI();

expected.filmName = filmName;

expected.locations = new List<String>();

expected.locations.Add ("New York County Courthouse<br>40 Foley Square<br>Lower Manhattan");

actual = target.getLocationsForFilm(filmName);

Assert.AreEqual(expected, actual);

filmName = "15 Minutes";

expected = new LocationListUI();

expected.filmName = filmName;

expected.locations = new List<String>();

expected.locations.Add("E. 60-66th St.and Madison Ave.<br>Upper East Side<br>Manhattan");

actual = target.getLocationsForFilm(filmName);

Assert.AreEqual(expected, actual);

filmName = "25th Hour";

expected = new LocationListUI();

expected.filmName = filmName;

expected.locations = new List<String>();

expected.locations.Add("World Trade Center<br>Lower Manhattan");

expected.locations.Add("Carl Schurz Park<br>Upper East Side<br>Manhattan");

actual = target.getLocationsForFilm(filmName);

Assert.AreEqual(expected, actual);

}

The main View is the HomeController. The unit test for this also makes use of the XML data read by the stand-alone XML prototype. The code for this follows:

public void Index()

{

// Arrange

HomeController controller = new HomeController();

// Act

ViewResult result = controller.Index() as ViewResult;

// Assert

ViewDataDictionary viewData = result.ViewData;

Assert.AreEqual("Welcome to Awesome!", viewData["Message"]);

List<SelectListItem> flicks = (List <SelectListItem> )viewData["movieList"];

Assert.AreEqual(178, flicks.Count);

Assert.AreEqual("\*batteries not included", flicks[0].Text);

Assert.AreEqual("12 Angry Men", flicks[1].Text);

Assert.AreEqual("15 Minutes", flicks[3].Text);

Assert.AreEqual("Die Hard: With a Vengeance", flicks[37].Text);

Assert.AreEqual("You've Got Mail", flicks[177].Text);

List<SelectListItem> locs = (List<SelectListItem>)viewData["locationList"];

Assert.AreEqual(1, locs.Count);

if (locs.Count > 0)

Assert.AreEqual("E. 5th St.<br>East Village<br>Manhattan", locs[0].Text);

}

Unit test are very useful in finding bugs (via the Test Debug option) and also ensuring that if the code breaks due to future changes it usually flags the break.