Table of Contents

1. Background research and investigations
2. Project Plan
3. Software development methodology employed
4. Requirements analysis
5. Use cases
6. Architecture/Design approach
7. Models (Class Models / Data Models etc.)
8. Data
9. Development Phase
10. Implementation of particular OOP constructs
11. Design patterns and architectural patterns implemented in the application
12. How cross-cutting concerns have been handled
13. Security of the application
14. Configuration of the application
15. Scalability of the application
16. Testing Approach (in terms of both functional and non-functional requirements)

o Other relevant features of the application, e.g.,

use of client-side processing,

use of Ajax,

use of web services,

use of a workflow engine etc.

use of an ORM tool

use of dependency injection / IoC containers

1. **Background research and investigations**

The ‘Rate my Amenities’ enterprise application is intended to provide users with a place to search for and view amenities in their local area, rate the amenity and view other users ratings and comments.

Users will be able to search for amenities using a range of criteria including location, type of amenity and rating. Alternatively users can search within a selected radius. This may be from a selected location or from a user’s current location, if GPS is available on the users device.

**2. Project Plan**

Due care should be taken to accurately record details of which team member was assigned responsibility for each activity

**HOW DO WE DIVIDE UP THIS?**

3. **Software development methodology employed**

This application was developed using Microsoft Visual Studio 2010, ASP.NET and C#

**4. Requirements analysis**

* **Functional Requirements**

1. Call data using an APIfrom Fingal Open Data <http://data.fingal.ie/> and display it.
2. Using search filters display appropriate data to the user.
3. Using a Google Maps API, display the search requirements.
4. Provide an Administrator Login to be stored in the Membership Database
5. Provide a user Login to be stored in the Membership Database
6. Provide a Twitter feed from Fingal Council.

7. Allow logged in users to insert comments.

* **Non Functional Requirements**
  1. Provide a basic user interface.

**5. Use cases**

**Actors**

**1. Non Registered User**

**2. Registered User**

**3. Administrator**

**4, System**

**Non Registered User**

**1. User browses index page.**

**2. User searches for amenity**

**Registered User**

**1. User browses index page.**

**2. User searches for amenity**

**3. User adds ratings**

**4. User adds amenity.**

**Administrator**

**1. User browses index page.**

**2. User searches for amenity**

**3. User adds ratings**

**4. User adds amenity.**

**5. User edits and deletes ratings**

**6. User edits and deletes amenities**

**System**

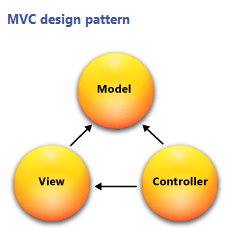
**1. System interacts with database**

**2. System interacts with Google Maps by means of an API**

(need to insert diagrams)

**6. Architecture/Design approach**

MVC Book Store was designed using ASP.Net MVC3 framework. It is developed using the C# programing language.



The database used in the development of this application was SQL Server Compact Edition. This is a free, embedded, file based database which does not require any installation or configuration.

We also used the Entity Framework version 4 to query and update the database. This is a flexible object relational mapping data API (ORM) which queries and updates the data stored in the database in an object oriented manner.

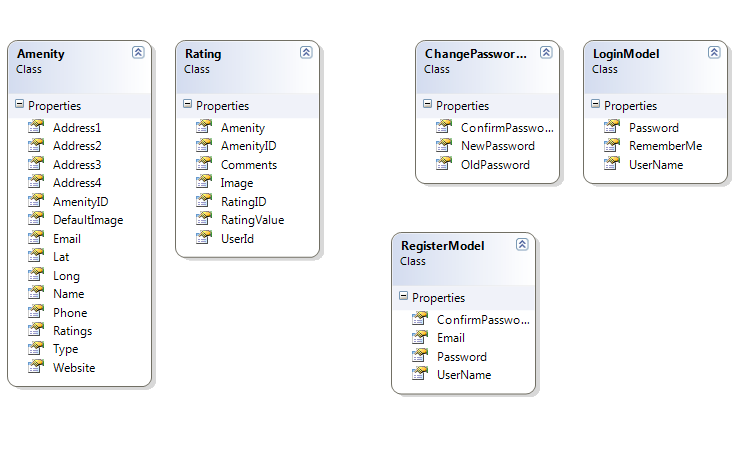
**7. Models (Class Models / Data Models etc.)**

We created a number of Data Models and classes based on the data in our datasets. We then created relationships between the models.

Models

* Account Model
* Amenity Model
* DataBaseDB
* DBInitialiser
* Ratings

Classes



# 8. Data

Data for the amenities used in this project are obtained from the public datasets provided by Fingal County Council ([http://data.fingal.ie/).](http://data.fingal.ie/).%20%20)  As the data is not subject to regular changes the data will be checked for updates periodically and imported to an internal database. For this project the datasets have been limited to a small range of amenities only and have not included services i.e. Garda Stations. The amenities used are and their corresponding URL is as follows.

* + Art Centres - http://data.fingal.ie/datasets/csv/Arts\_Centres.csv
  + Cinemas - http://data.fingal.ie/datasets/csv/Cinemas.csv
  + Heritage Venues - http://data.fingal.ie/datasets/csv/Heritage\_Venues.csv
  + Parks - http://data.fingal.ie/datasets/csv/Parks.csv
  + Play Areas - http://data.fingal.ie/datasets/csv/Play\_Areas.csv

**File Format and Data Set**

For most of the amenities the data is available in CSV, XML and KML formats. However for this project the CSV file type was used.

While a number of the amenities have an extensive range of data available only a subset of the data was used for this project. The fields used were common across all of the datasets and these were Name, Address1, Address2, Address3, Address4, Phone, Email, Website, LAT and LONG.

## Data Import

As outlined in section 2 the data for the amenities is imported from public datasets provided by Fingal Co. Co. To import the data the following code was implemented.

* Each of the dataset URL’s and a definition of its ‘type’ were added to a multidimensional array called amenities in the model/DBInitialiser.cs.
* A For loop is used to navigate through each of the datasets and complete the following.
  + Create a new instance of the CSVParser
  + Fetch the data from the URL using a HTTPWebRequest
  + Stream the data through the parser
  + Pass in the amenity ‘type’.

Add and save the data to the database.

**9. Development Phase**

## To setup construct for import CSV data file

DataImport Code:

IDataParser.cs : located in /DataImport, created to insure that any class created for that interface includes three methods.

CSVParser.cs: located in /DataImport , creates a new instance of IDataParser.

Unit Tests:

Units Tests located in /CSVPaserTest.cs

Unit test were carried out to test the parser to:

* + test for csv files
  + test for non csv files
  + test for null file (csv parser file insert value == null)
  + set up test array for expected amenities
  + test for object within CSV data fields

## Database Migrations

To create the initial migration we ran the Package Manager Console

* Tools => Library Package Manager =>Package Control Console
* Enable – migrations ‘creates initial migration class’ InitialCreate.cs
* Add migration
  + Name new image column
* This create a new migration class called newimagecolumn.cs’

**10. Implementation of particular OOP constructs**

**What ones did we use**

**11. Design patterns and architectural patterns implemented in the application**

**What ones did we use**

**12. How cross-cutting concerns have been handled**

Though there was a large geographical spread between the team members we compensated for this by making use of the frequent opportunities to meet presented by the course schedule.

Most of the discussion and planning was done on these occasions with the team working together at these times in developing code and researching problems that arose during these sessions.

We also used these times to plan for the days when we would not be together were we would research particular functionality and implement it.

**13. Security of the application**

The application uses ASP.Net’s Membership Framework to manage user accounts.

ASP.Net manages security in a number of ways.

1. By authenticating the user.
2. By authorizing the user
3. Setting up User Accounts
4. Setting up Roles.
5. Authentication is the process of ascertaining the users identity.
6. Authorization is the process of determining whether the user has the authority to access a specific resource.
7. A user account is a store for persisting information about a particular user.
8. A role is a label assigned to a user and provides a means for defining authorization rules and page level functionality.
9. The application defines Administrator and User Roles with the Administrator having full access to the applications functions.

**14. Configuration of the application**

**15. Scalability of the application**

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

We approached testing by developing tests as we progressed with the development of the application.

ASP.Net provides tools for the development of Unit Tests.

**Definition: Unit Testing**

Unit Testing is a method by which units of source code are tested to see if they are fit for use. Unit Tests can be written quickly and are fully automated.

**o Other relevant features of the application, e.g.,**

use of client-side processing,

use of Ajax,

use of web services,

use of a workflow engine etc.

use of an ORM tool

use of dependency injection / IoC containers

etc.