**6. Architectural approach**

The application follows a 5-tier structure. On the bottom-most level is the database. This provides the persistence of data, namely Film Locations and their dependent objects, and Restaurants and their dependent objects. The Film Locations objects contain lists of specific Locations, each of which contains a Point object. The Point object is a pair of coordinates. This model is referenced by both the Location object and the Restaurant object.

These models are used to represent existing constructs, as per OOP standards. The Restaurant object, for example contains details that one would expect representative of a restaurant, cuisine type, quality, etc., in addition to physical aspects such as the Address, held in a separate model.

Each model is mapped to a table in the database using EntityFramework, an ORM tool designed for use with C#. The EntityFramework provides Domain Model persistence. It looks after the relational mapping of inter-object relationships and the way that they are stored in the application. Due to the scale of the application and the code-first model with basic relationships, impedance mismatch was not an issue in developing the lower-tier structure.

Object Models are accessed through the second tier, the Data Access Layer (DAL). This layer provides a level of abstraction from the data models, providing a one-stop-shop for access methods including ‘CRUD’ functionality. A design decision was taken here to include the object creation functionality within the DAL. This means that constituent parts of objects, i.e. names, coordinates etc., are passed into the DAL methods, constructed into objects and then persisted. The reasoning behind this decision was to retain a clean space for logic in the Business Logic Layer.

As mentioned, the Point model is separate from both main models but also referenced by both. This is to make data processing more fluent in the Business Logic Layer (BLL). Here ViewModels are received from the top-most two tiers and broken out into their constituent parts. These parts, depending on the required action, are then passed to the DAL for management and persistence or passed into the Location Calculator (through the Restaurant Finder) for processing. This tool uses the Point object of the incoming Location to search through the existing persisted Restaurants matching the Restaurant’s coordinates against those of the Location to find Restaurants within a given radius. This functionality uses the algorithm in the Location Calculator.

Above the BBL lie the Presentation and View/Client-Side layers. These have their own ViewModel constructs that convey subsets of information between the UI and the BLL. Manipulation of these objects is performed in the BLL, calling persisted objects through the DAL when required.

This layered application architecture allows a clearly exposed workflow. Maintenance is made easier by defining subsets of functionality in set levels of the architecture. There is a clear separation of concerns enabling replacement of sets of functionality without excessive side-effects. The application makes extensive use of OOP principles in order to keep datasets in strict cohesion with relative elements while retaining a loosely coupled relationship with other datasets. The general communication style between layers and objects is the call-and-return style. This enables flexibility within the application.