**MyNewMedia.com – Rough Architecture**

**Justification –**

The MyNewMedia site has a very clearly defined architecture thanks to the Django framework. Django was developed as a slight modifaction to the standard Model-View-Controller (MVC) style architecture. The framework refers to the Controller layer as the ‘View’ and the View layer as the ‘Template’. So the name becomes the Model-View-Template (MVT) architecture. The purpose behind using the MVT architecture is to decouple all layers of the application and allow all developers to work on different programming aspects – from the database to the front end design – at the same time. This enables teams to work rapidly and effectively on any application

There are two keys to a successful Django web application, and any application that uses MVC or MVT architecture: code reuseability and separation of concerns. The code for any Django project is organized into “applications” that can be bundled and reused anywhere in the application, or exported and used in different projects altogether. Because so many web applications require very similar features, the ability to reuse code becomes absolutely essential. The seperation of each of the layers not only increases the productivity of developers by allowing work to occur simulataneously; they also enable the protabliliyt of the code. By making sure that the template layer doesn’t rely on a certain database schema, a team could easily make changes to the database without affecting the other layers. It makes the application much more maintainable.

**Model –**

The model layer essentially acts as a representation of the database and all related APIs. Within the model are defined entities (Python functions in Django’s case) that correspond to tables with in the database. Django will write the SQL calls needed to create the database tables so ensure they are identical to the model definition. In addition to building the schema, Django will provide an API that allows our application the hooks it needs to communicate with the other layers. In other cases this might not be the case. Many MVC architectures use a passive model, where the view is responsible for updating the model. Ours will use the active version, where the model contains the update calls and the view simply accesses them

**View –**

This is one area where our architecture will differ from the typical MVC architecture. In most MVC architectures, the view layer is what is shown to the user; the HTML or CSS. Django’s view layer is much more akin to the controller of the traditional MVC. Most of the logic is defined in the view layer. It acts as the middleware between the model and the template (as shown in Figure 1). As the user manipulates the data presented in the view the controller modifies the model. Then, it notifies the template layer that the model has changed and updates the template with the requested data.

**Template –**

The template is everything the user sees. This includes all HTML or CSS files, any images, any audio, text, more or less anything that can be queried. The Django framework refers to this as the template, instead of the view because of how extensable it is. Using HTML tags, we can dynamically generate our pages based on the active view.