**Django Architecture – Its Pro's and Cons**

**Introduction:**

In this post we will be taking a look at the advantages and disadvantages of the Django framework. Since Django is implemented using the MVC design pattern, all advantages and disadvantages of the MVC design pattern are incorporated in Django. Alongwith those, there are a few Django specific matters, and we will be taking a look at them.

**Advantages of Django Framework:**

The Django framework has the following advantages:

1) It is easy to use and implement and programmers benefit from it since it saves a lot of development time. Different components may be worked on simultaneously by different groups of professionals – for example, designers can work on the “view” component while developers/programmers can work on the controller part. The DB engineers and admins can use the same time to optimize the entire database, so all three components can be worked on simultaneously. From the Django perspective, it means that the different groups (designers, developers, and DB professionals) can simultaneously work on template files, views.py file and models.py file.

2) Django also supports asynchronous views.py function calls, so responsive web apps and mobile backends are well supported. This saves the end-user's time and provides the user with satisfaction. One can write jquery or xmlHttp calls from a given application screen without disturbing it and get a response while the user is busy with something else.

3) Django allows programmers to create large and extensive web applications and REST APIs with the least amount of effort. It provides the programmer with an ORM (Object Relational Mapping) to manipulate values in database tables through instances of classes defined in the models.py file. The programmer need not write complex SQL statements to handle the data directly. However, Django also allows the flexibility to let the programmer write actual SQL statements if she chooses to do it.

4) Django provides the programmer with 2 excellent classes to handle HTTP(S) protocol – HttpRequest and HttpResponse. The views.py functions take a HttpRequest object as a parameter to the function and it returns an instance of the HttpResponse class. The APIs provided by these above mentioned classes are so well designed that one can manipulate every aspect of the request and response objects.

5) Modification of one component doesn't affect any other component. For example, one may add some fields to a models.py class, but doing that won't affect the views.py functions. The views.py won't be able to get the values of the new fields in the models.py class, but that won't cripple the existing views.py.

**Disadvantages of the Django framework:**

Some of the advantages listed above can also be disadvantages in specific scenarios. Let's take a look at the disadvantages.

1) If the programmer wants to create only a few pages for a website, then Django can be a big headache. The programmer has to do a lot of overhead activities in order to implement a simple 3 or 4 page website. So basically Django is not suitable for small projects. No point firing a cannon to kill a fly.

2) If the web application being created contains a lot of tables in a relational database and each action in the views.py file is a complex enough to retrieve values from more than one table at once, then Django may come in the way of doing it. We mentioned above that it is possible to write SQL statements in Django, but it is not advised to be frequently used. In such a scenario a better solution would be to create the web app without using Django, and write all the complex SQL statements and optimize them in the code.

3) It has to be kept in mind that not every web/mobile app will fall in line with the MVC pattern. Certain web applications (usually ones that are extremely data intensive, like a face recognition system) should be implemented using “Actor” pattern or any other pattern that is appropriate for it. Django isn't of much use in such scenarios.

**Conclusion:**

Thus, as discussed above, when creating a web application or a mobile app backend or a REST API, we need to first identify if Django would serve our purpose. If so, we may use Django, but in case that is not the case, we should not force our app to be implemented using Django. It is a matter of design, and hence, a bad judgement at that stage can spoil the entire app