Django Static:

What it Accomplishes?
This django library allows users to have static files. This can be used to display files that will not change like javascript or css files. This library is imported within class_project/urls.py

How does this technology accomplish what it does?
The django official documentation here explains the intricacies of using the static library correctly during development, testing and deployment. Please refer to this to see how it should be used.

The source code for this library here shows us that the static()method does the heavy lifting. It takes (`settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)` as it's parameters. It escapes characters, and also removes any '/' characters. It then returns this escaped path along with (view) and any other additional arguments that were passed.

This way, it allows the developer to display static files.

ASGI:

What it Accomplishes?
ASGI (Asynchronous Server Gateway Interface) provides a standard interface between async-capable Python web servers, frameworks, and applications. We use this within our class_project/asgi.py

How does this technology accomplish what it does?
The official documentation of the django usage of asgi can be found here.

The method being called to create an asgi object is here. This method returns an ASGIHandler object which can be found here. The object being returned will be used within our class_project/asgi.py.

HttpRequest:

What it Accomplishes?
This request object handles receiving a HTTP request and parses the data within it. It then serves this data using various data structures within this HttpRequest class.

How does this technology accomplish what it does?
The django official documentation describes how to use the various data elements(user, method, session etc). Please refer to this to see the details present within this class.

This class mentioned above has its source code here. This class will be instantiated and is used within most of our views.py files. As one can see this class contains methods and variables that contain the details about a HTTP request, which we use.

In addition, this class contains a headers() method, which returns an object of another class called HttpHeaders. This HttpHeaders class contains a function called parse_header_name() which parses our header details and returns it.

HttpResponse:

What it Accomplishes?

In contrast to the HttpRequest, this class must be populated by the developer and can be used to serve up data. This uses a class called HttpResponse, and although it does not accomplish too many things, I feel like it is worth mentioning in the report.

How does this technology accomplish what it does?

The django official documentation describes how to use the various data elements(user, method, session etc). Please refer to this to see the details present within this class.

This is used by first using the HttpResponseBase. The class object that we receive this is then passed to HttpResponse. Once passed, this HttpResponse object is then used to populate various different data elements within this class object.

Path:

What it Accomplishes?

Returns an element for inclusion in urlpatterns within urls.py.

How does this technology accomplish what it does?

The usage of Render is present via the django official documentation listed here.

This library lives within the django source code, hows init.py is here. We see the path function being used here. In this line, the _path function is called and this either:

a) Returns a URLResolver class object
b) Returns a URLPattern class object
c) Raises a typeError

Include:

What it Accomplishes?

A function that takes a full Python import path to another URLconf module that should be “included" in this place. This is present via the urlpatterns within urls.py.

How does this technology accomplish what it does?

The usage of Include is present via the django official documentation listed here. This function is present here. This function takes arguments, and uses these parameters to create a tuple with three values. These three values are (urlconf_module, app_name, namespace). This tuple is returned.

What license(s) or terms of service apply to this technology?

The entire application runs on the django framework. The owner of the license of all django libraries is Django Software Foundation as mentioned here. They give the right to use django under many circumstances mentioned in the link provided.

It is mentioned that you can use the libraries within the following conditions of :

1) Service Identification
2) Django-related software project
3) Groups and Events
4) Merchandise
5) Products and services serving the community
6) Other Commercial activity
7) Domain Names
8) Uses outside of this license
9) Community standards
10) Interpretation