**Admin.py:**

The admin.py file used to display your models in the Django admin panel. You can also customize your admin panel. Admin class is the representation of a model in the admin interface. Usually these are stored in a file named admin.py in your application.

Django admin application can use your models to automatically build a site area that you can use to create, view, update, and delete records(CRUD).

**Key Functions:**

1.Registering Models:

Make models visible in the Django admin interface.

2.Customizing Admin Interface:

Modify field sets, add custom admin actions and more.

3.Defining Admin Classes:

Create custom admin classes for models.

**Key Concepts:**

1.admin.site.register():

Register models or custom admin classes.

2.Model Admin:

Base class for custom admin classes.

3.Search\_fields:

Specify fields to search in the admin list view.

Activation of Django admin file:

When you put ‘django. contrib. admin’ in your INSTALLED\_APPS setting, Django automatically looks for an admin module in each application and imports it.

**Settings.py:**

A Django file settings file contains all the configuration of your Django project. In this article, the important points of the settings.py file of Django will be discussed. A settings file is just a python module with module-level variables.

The settings.py file is typically used to store configuration information in Django. It may contains the site URL, paths to various directories, and other values that the executable code may use. By changing and supplementing this file, the developers configure Django projects.

It is used for running the development server. Doing database migration.

**Using of settings.py:**

By using an environment variable, DJANGO\_ SETTINGS\_ MODULE .The value of DJANGO\_SETTINGS\_MODULE should be in python path syntax, e.g. mysite. settings.

**Primary Functions:**

**Project Configuration:**

Define project\_ wide settings, such as database connections, installed apps, and middleware classes.

**Environment Variables:**

Stores environment-specific settings, like secret keys, API keys, and database credentials.

**Static Files Configuration:**

Configuration static file serving, including directories and URL prefixes.

**Template Configuration:**

Define template engines, directories, and settings.

**Urls.py:**

The urls.py file provides a way to map a URL to a view. A view is a python callable that a request and return an HTTP response. To map a URL to a view, we can create a url patterns list in the urls.py file. Each element of the list is a path() or re\_ path() function call that maps a URL pattern to a view.

**Use of urls.py:**

A request in Django first comes to urls.py and then goes to the matching function in views.py . Python function in views.py and take the web request from urls.py and give the web response to templates. It may go to the data access layer in modules.py as per the queryset.

**Primary Functions:**

URL Routing:

Maps URLs to views, templates, or other URLs.

URL pattern Definition:

Define URL patterns using regular expressions.

View mapping:

Maps URL pattern to specific views.

Namespace Definition:

Define namespaces for URL patterns.