*DJANGO.*

Django is a high web framework that encourages rapid development and a clean design.

We install django because python does not have the capability to develop a web therefore; django helps to provide the capability.

Therefore without it, we can’t create a project in python.

**How to create a django project**

As required, we follow steps in order to come out with the expected goal as mentioned bellow.

* When starting the project we begin by creating a parent folder, example,*” Development*” under it, there must be a project i.e. ”fproject”.
* Configuring the virtual environment
* After this, we go to the terminal and install the virtual environment,”pip install virtualenv-then do virtualenv venv.after this we activate it, *venv\scripts\activate.*
* The venv ,(virtual environment)separates one project from another.
* Installing django
* -we run, Pip install django.
* *Pip install django==3.0.8*.(this is a django version)
* After these commands we start the project
* -*django-admin start project* (project name)
* Then we run ,cd project folder,(cd-change directory into)
* *Run server, python manage.py runserver*

Python manage.py createsuperuser i.e., this allows you to enter username, pass word and email.

* Then run *python manage.py start app* i.e..(myapplication) here
* N.B, you can have more than one application in a project.
* This is because, they interconnect.
* N.B Django was meant for applications however it can be used to create websites.
* **Configuring and creating the templates**
* Go to templates,
* Create folders,eg static,templates by the command mkdir static or templates.
* Go to settings.py add application to the installed apps,
* Create an url.py file in the application folder and include the path, here we also create another url file for the project.
  + The urls call for the views requests .
  + The view is another file under a project.
  + It can also be referred to as a function or method
    - A view is a function (method) that responds to url requests.
    - A server uses views as a response to the response to the web browser ,index page and static specifies the interfaces.note that static folder is under the application, it is a folder where images, audios videos etc. should be applied
    - Models. And Forms.
    - Define Models
    - Open the models.py file within the application.
    - Define Models: Define the models for project folder and Sales. Create classes like Category, Part, Branch, Sale, etc., with appropriate fields such as category name, part name, part number, date of arrival, price, quantity, etc.
    - Set Up Relationships: Define relationships between models using ForeignKey and OneToOneField as needed (e.g., a part belongs to a category, a sale is associated with a part, etc.).
    - Run Migrations: Run *python manage.py makemigrations* and *python manage.py migrate* to apply the database required changes.
    - Forms
    - When creating forms, we apply aunthetication or library.
    - Here, as a view is a method,it pushes the two fields (username and password)into the crispy forms.,{{form | crispy}}
    - Add {% block content %}.This is what we call django templating language.
    - Note. Whenever you come across, csrf, take that it is for security, it prevents unauthorized users from tampering or penetrating into the form contents.
    - Form also helps us to enter data that is not native of the html.
    - That is why we write it as, (csrf\_token)
    - Form more so, verifies that the data being posted to the server is from our own form as well as sending it to the data base.
  + Admin part.
  + Register Models: Open the admin.py file within the project app and register the models using admin.site.register(Model name).
  + Superuser: Create a superuser using python manage.py *createsuperuser to access the Django admin panel.*
  + Customize Admin, Log in to the admin, and start adding categories, parts, branches, etc. This allows the manager to manage stock and other data.
  + (<http://127.0.0.1:8000> )copy this to the urls as said above after running the server.
* Views and Templates
* Create Views: Define views for listing parts, adding new parts, recording sales, etc., in the views.py file within the project app.
* Create Templates: Create HTML templates for these views in the templates directory within the project app.
* Design Interfaces: Use HTML, CSS, and possibly JavaScript to create user-friendly and responsive interfaces for the views

**Implement Receipts**

* Create a new view for generating receipts after a successful sale.
* Design the receipt layout using HTML/CSS. Include details like buyer's name, contact, part name, date, quantity, price, total amount, payment details, etc.
* Connecting the urls.
* Define URLs: Create URL patterns in the urls.py file within the project app, mapping to the views you've created.

Test the Application

* Run the Server: Start the Django server with *python manage.py runserver.*
* Access the Application: Open a web browser and access the provided URL *,(http://127.0.0.1:8000 )* to connect with the application.
* Test all functionalities including *adding parts, recording sales, generating receipts, etc*.
* Remember, this is, an overview, and the implementation details can be extended. You'll need to write code to handle form submissions, *record sales, manage stock levels, generate receipts,* and more. Additionally, you can enhance the application's usability and appearance by utilizing Django's template inheritance, forms, and possibly adding JavaScript for dynamic interactions.
* 