**Django**

* A python framework to develop web application
* MVT (Model View Template) architecture

**Set Up**

1. Install djnago

pip install django

1. Create django project

django-admin startproject project\_name .

1. To run Server

python manage.py runserver

1. Define required modules/app for the system

django-admin startapp module\_name

1. Configure all modules in settings.py INSTALLED\_APP[]
2. Create templates, media, static folders and configure them in settings.py

**Migration & Django-Admin**

1. Create models (model classes) for all modules in models.py file
2. The create migration files

django-admin makemigrations

1. Create tables

python manage.py migrate

1. Create a user to access admin panel

python manage.py createsuperuser

1. Register each Models in admin.py file of respective modules

**URL, Views and Templates**

**Creating URL**

1. Creates urls.py in each module
2. Define a suitable URL extension for each modules in project’s urls.py

urlpatterns = [  
 path('admin/', admin.site.urls),   
  
 path('info/', include('information.urls')),  
 path('jobs/', include('jobs.urls')),  
 path('organizations/', include('organizations.urls')),  
 path('jobseekers/', include('jobseekers.urls')),  
 path('applications/', include('applications.urls')),  
]

1. Create a URL extension required for all the pages related to a particular module

from django.urls import path  
from information import views  
  
  
urlpatterns = [  
 *# 127.0.0.1:8000/info/about* path('about', views.show\_about, name='about'),  
  
 *# 127.0.0.1:8000/info/contacts* path('contacts', views.show\_contacts, name='contacts'),  
  
 *# 127.0.0.1:8000/info/policies* path('policies', views.show\_policies, name='policies'),  
]

1. Create views function required to perform each URL actions.

from django.db import connection  
from django.http import HttpResponse  
from django.shortcuts import render  
from jobs.models import Job  
from organizations.models import Category  
  
  
def show\_about(request):  
 return render(request, 'about.html')  
  
  
def show\_contacts(request):  
 return render(request, 'contact.html')  
  
  
def show\_policies(request):  
 return render(request, 'policy.html')

1. Retrieve data required for templates using Django queries (or raw queries) and send it in the form of dictionary.

def show\_home(request):  
 jobs = Job.objects.all()  
  
 cursor = connection.cursor()  
 cursor.execute('SELECT category\_id, count(\*) FROM jobs\_job GROUP BY category\_id')  
 counts = cursor.fetchall()  
   
 return render(request, 'index.html', {'jobs': jobs, 'counts': counts})

**HTML (Hyper Text Markup Language)**

* Used to create web pages
* Defines contents of a web page not their design
* Tag based language, tag has their properties called Attributes

**CSS (Cascading Style Sheet)**

* Used to design layouts of web pages
* Selectors, properties and their values
* Can be applied by using:
  + Inline CSS: style as attribute
  + Embedded CSS: style as tag
  + External CSS: style as file

**Properties**

1. Text/Font properties
2. List properties

* list-style
* list-style-type
* list-style-image
* list-style-position

1. Box properties

* background
* height
* width
* border
* margin
* padding

1. Floating & positioning

* float & clear
* position
* display

1. Styling

* Opacity/visibility
* transition
* animation
* transform

**Selectors**

|  |  |
| --- | --- |
| Universal | \*{} |
| Tag | h1{} p{} |
| Class | .class\_name{} |
| Id | #id\_name{} |
| Multiple element | h1, p, .class\_name{} |
| Descendant | #id\_name ul li{} div p {} |
| Child | #id\_name > ul{} ul > li {} |
| Sibling | p ~ h1{} |
| Adjacent Sibling | p + h1{} |
| Attribute | input[type="submit"]{} a[href="home.html"]{} |
| Nth child | ul li:nth-child(2){} |
| Pseudo Element | h1::before{} h1::after{}  h1::first-letter{} p::first-line{} |
| Pseudo Class | a:hover{} a:active{} a:visited{} a:link{} |

Media Query: for responsive design

Bootstrap: CSS library for responsive design

**JavaScript**

<script>

// JS code goes here….

</script>

<script src=”file.js”></script>

1. Variables & Datatype
2. Operators
3. Conditional Statements
   1. if….else if… else
   2. switch
4. Looping Statements
   1. for(){}
   2. while{}
   3. foreach
5. Array
6. Function
7. JSON

**JQuery – a JavaScript Library**

**Database**

Stores data in the form of tables.

MySQL Server, MSSQL Server, Oracle, SQLITE

SQL: Structured Query Language

No SQL Database: MongoDB, POSTGRESQL

XAMPP-> Apache MariaBD PHP Perl

CRUD

* CREATE DATABASE db\_name;
* USE db\_name;
* CREATE TABLE users(id int PRIMARY KEY AUTO\_INCREMENT,

name varchar(100), dob date,

gender ENUM('Male', 'Female', 'Other'),

mobile varchar(20), email varchar(100),

username varchar(10) UNIQUE KEY, password varchar(50),

status boolean, balance double,

comment mediumtext)

* INSERT INTO users (name, gender, email, mobile, username, password, balance, comment) VALUES ('Pradeep Chapagain', 'Male', 'pradeep@email.com', '9841112233', 'pradeep', 'pradeep123', 100000, 'Comment text goes here.....')

**OR**

INSERT INTO users VALUES ('', 'Bigyan Dhakal', '2000-01-01', 'Male', '9841112233', 'bigyan@email.com', 'pradeep', 'bigyan123', 1, 100000, 'Comment text goes here.....')

* SELECT \* FROM users;
* SELECT name, dob FROM users;
* SELECT name, dob, balance FROM users;
* SELECT \* FROM users WHERE name='Bigyan Dhakal';
* SELECT \* FROM users WHERE dob < '1999-01-01';
* SELECT \* FROM users WHERE dob BETWEEN '1999-01-01' AND '2002-01-01';
* SELECT \* FROM users WHERE name LIKE 'b%';
* SELECT \* FROM users WHERE name LIKE 'p%' AND gender='Male';
* SELECT \* FROM users WHERE name LIKE 'p%' OR gender='Male';
* UPDATE users SET name='Sajana Dahal', gender='Female', mobile='9841122334', email='sajna@email.com', username='sajna', password='sajna123' WHERE id=5;
* UPDATE users SET status=0 WHERE balance=0;
* DELETE FROM users WHERE id=6;
* TRUNCATE table users;
* DROP table users;
* CREATE TABLE orders (id int PRIMARY KEY AUTO\_INCREMENT, order\_date date, product\_id int, user\_id int, qty int, status ENUM('Submitted', 'Dispatched','Delivered','Cancelled'),

FOREIGN KEY(product\_id) REFERENCES products(id),

FOREIGN KEY (user\_id) REFERENCES users(id));

* SELECT \* FROM orders

JOIN products ON products.id = orders.product\_id

JOIN users ON users.id = orders.user\_id;

* SELECT orders.order\_date, products.name, products.price, users.name, users.mobile FROM orders

JOIN products ON products.id = orders.product\_id

JOIN users ON users.id = orders.user\_id;