**Django**

* A python framework for development of web application
* MVT (Model View Template) architecture

**Front End Technologies**

* HTML
* CSS, Bootstrap
* JavaScript, JQuery

**Front End Technologies**

* Python

**Database**

* Sqlite
* MySQL

**HTML (Hyper Text Markup Language)**

* used to create web pages
* It defines contents of web pages, heading, paragraph, list, image table etc.
* HTML is all about tags and their attributes

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>Welcome</title>  <meta charset="UTF-8" />  </head>    <body>  <h1 align="center">HTML</h1>  </body>  </html> |

|  |  |
| --- | --- |
| Heading Tag | <h1>Text</h1> <h6>Text</h6> |
| Paragraph Tag | <p>Text</p> |
| Lists   1. Unordered List 2. Ordered List 3. Definition List | <ul><li>Item 1</li>><li>Item 2</li></ul>  <ol><li>Item 1</li>><li>Item 2</li></ol>  <dl><dt>Title</dt>><dd>Details</dd></dl> |
| Formatting Tags | b, u, i, sup, sub, pre, em, strike, del, strong |
| Character Entities | &npsp; &copy; &reg; |
| Table | <table><tr><td></td></tr></table> |
| Image Audio Video | <img src="himal.jpg" alt="Himal Image" />  <audio controls>  <source src="audio.mp3" type="audio/mp3" />  <source src="audio.ogg" type="audio/ogg" />  </audio>    <video controls>  <source src="video.mp4" type="video/mp4" />  <source src="video.ogg" type="video/ogg" />  </video> |
| iframe | <iframe src=" " title=" " frameborder="0"></iframe> |
| Hyperlinks  Anchor Tag | <a href="https://www.google.com/" target="\_blank">Google</a>  <a href="second.html">Second Page</a>    <a href="#top">Go to Top</a> |
| Forms | <label>Name</label>  <input type="text/password/radio/checkbox/email/url/range/number/date/file/submit/button/reset" name="Fullname" required  maxlength="5" placeholder="Enter Name" />  <select name=””><option value=””></option></select>  <textarea></textarea>  <button type=”submit/button/reset”></button> |

**CSS (Cascading Style Sheet)**

1. Inline CSS: style as attribute
2. Embedded CSS: style as tag
3. External CSS: style as file

* Text Properties
* Box Properties

height, width, background, border, border-radius, margin, padding

* Floating & Positioning

float, clear, display, position, top, right, bottom, left z-index

* Transform, transition, animation

|  |  |
| --- | --- |
| Universal | \*{} |
| Tag selector | P{} h1{} |
| Id Selector | #id\_name{} |
| Class Selector | .class\_name{} |
| Multiple Element Selector | P, h1, #id\_name, .class\_name{} |
| Descendant Selector | p strong{} #my\_div h1{} |
| Child Selector | #my\_div > h1{} ul > li{} |
| Nth child selector | p:nth-child(n) {} |
| Sibling Selector | p ~ h1{} |
| Adjacent sibling | P + h1{} |
| Pseudo Element | h1::first-letter{} p::first-line{} h1::before{} h1::after{} |
| Pseudo class | a:hover{} a:active{} a:visited{} a:link{} |
| Attribute selector | a[target=”\_blank”]{} input[type="text"]{} |

**Database**

* Used to store data in the form of tables
* SQL databases follows SQL syntax

E.g. MySQL, MSSQL, Oracle

**CRUD (Create Retrieve Update Delete)**

students

name, address, email, mobile, username, password

courses

title, duration, fee, contents, details

* SHOW DATABASES;
* CREATE DATABASE database\_name;
* USE database\_name;
* CREATE TABLE students (id int PRIMARY KEY AUTO\_INCREMENT,

name varchar(50),

address varchar(150), dob date,

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

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

username varchar(10) UNIQUE KEY, password varchar(20),

details longtext, status boolean);

* SHOW TABLES;
* DESC students;
* INSERT INTO students (name, address, email, mobile, dob, gender, username, password, details, status) VALUES ('Ram Kumar', 'Kathmandu', 'ram@email.com', '98123456780', '1998-01-01', 'Male', 'ram123', 'ram@123', 'Details about student', 1);

**OR**

INSERT INTO students VALUES ('', 'Hari Prasad', 'Lalitpur', '1998-01-01', 'Male', 'harry@email.com', '98123456780', 'harry1', 'ram@123', 'Details about student', 1);

* SELECT \* FROM students;
* TRUNCATE TABLE students;
* ALTER ……
* DROP TABLE courses;
* UPDATE table\_name SET col\_name='new value' WHERE id=1;

UPDATE students SET name='Hari Sharma' WHERE id=3;

UPDATE students SET name='Hari Sharma', address='new', email='new', mobile='new', password='new' WHERE id=3;

* DELETE FROM table\_name WHERE id=1;

DELETE FROM students WHERE id=3;

SELECT \* FROM `plants\_plant` WHERE name='snake plant';

SELECT \* FROM `plants\_plant` WHERE name LIKE 'sn%';

SELECT \* FROM `plants\_plant` WHERE price > 500;

SELECT \* FROM `plants\_plant` WHERE price BETWEEN 1400 AND 1500;

SELECT \* FROM `plants\_plant` WHERE price > 1500 and type='Indoor';

SELECT \* FROM `plants\_plant` WHERE price > 1500 or type='Indoor';

SELECT \* FROM `plants\_plant` WHERE price IN (300, 500, 700);

CREATE TABLE payments(id int PRIMARY KEY AUTO\_INCREMENT, amount double, date timestamp, student\_id int,

FOREIGN KEY (student\_id) REFERENCES students(id));

SELECT \* FROM `payments`

JOIN students ON students.id = payments.student\_id

SELECT payments.\*, students.name FROM `payments`

JOIN students ON students.id = payments.student\_id

WHERE students.name=’value’

**Django Project Setup**

* pip install django
* django-admin startproject project\_name .
* python manage.py runserver
* django-admin startapp module\_name
* django-admin makemigrations

OR

python manage.py makemigrations

* python manage.py migrate
* python manage.py createsuperuser

**SETUP**

1. create Django project

django-admin startproject project\_name .

1. create all modules needed for the application

django-admin startapp module\_name

1. create templates, static, media folders
2. add all modules to INSTALLED\_APP in settings.py
3. add templates directory to templates DIR, static URL and directory and media URL and directory in settings.py

**MODELS**

1. define relevant model classes to their modules’ models.py
2. create migration files using:

python manage.py makemigrations

1. Creates tables using:

python manage.py migrate

1. Create a super user to access Django admin panel using:

python manage.py createsuperuser

1. Register your models to admin.py in each modules

**URLS & VIEWS**

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

path('info/', include('information.urls')),

1. Define required URLs for the module in module’s urls.py

|  |
| --- |
| from django.urls import path from information import views   urlpatterns = [  *# http://127.0.0.1:8000/info/about* path('about', views.show\_about),   *# http://127.0.0.1:8000/info/contacts* path('contacts', views.show\_contacts),   *# http://127.0.0.1:8000/info/policy* path('policy', views.show\_policy) ] |

1. Define relevant functions in views.py to render required page

|  |
| --- |
| from django.http import HttpResponse from django.shortcuts import render   def show\_home(request):  return render(request, 'index.html')  def show\_about(request):  pass   def show\_contacts(request):  pass |

1. Create a new module accounts
2. Create User class inheriting AbstractClass in models.py
3. Register accounts in settings.py
4. Change AUTH\_USER\_MODEL to accounts.model in settings.py
5. Use new User class as foreign key in Jobseeker Model
6. Change registration form accordingly

**API**

* SOAP
* Rest

**Methods**

GET POST PUT DELETE

1. pip install djangorestframework
2. Create serializers.py in relevant module as:

from rest\_framework import serializers  
from about.models import About  
  
  
class AboutSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = About  
 fields = ['id', 'title', 'detail', 'status']

1. Create viewset.py in relevant module as:

from rest\_framework import viewsets  
from about.models import About  
from about.serializer import AboutSerializer  
  
  
class AboutViewSet(viewsets.ModelViewSet):  
 queryset = About.objects.all()  
 serializer\_class = AboutSerializer

1. Create routers.py in main project folder to define urls for API as:

from rest\_framework.routers import DefaultRouter  
from about.viewset import AboutViewSet  
  
router = DefaultRouter()  
router.register('about', AboutViewSet)

1. Add path for routers in main project’s urls.py as:

from django.conf.urls.static import static  
from django.contrib import admin  
from django.urls import path, include  
from .routers import router  
from about import views  
from broadway import settings  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path(r'^ckeditor/', include('ckeditor\_uploader.urls')),  
  
 path('', views.show\_home, name='home'),  
 path('about/', include('about.urls')),  
 path('news/', include('news.urls')),  
 path('comment/', include('comment.urls')),  
 path('account/', include('accounts.urls')),  
 path('api/', include(router.urls))  
]

Hence the URL for about API is: http://127.0.0.1:8000/api/about/