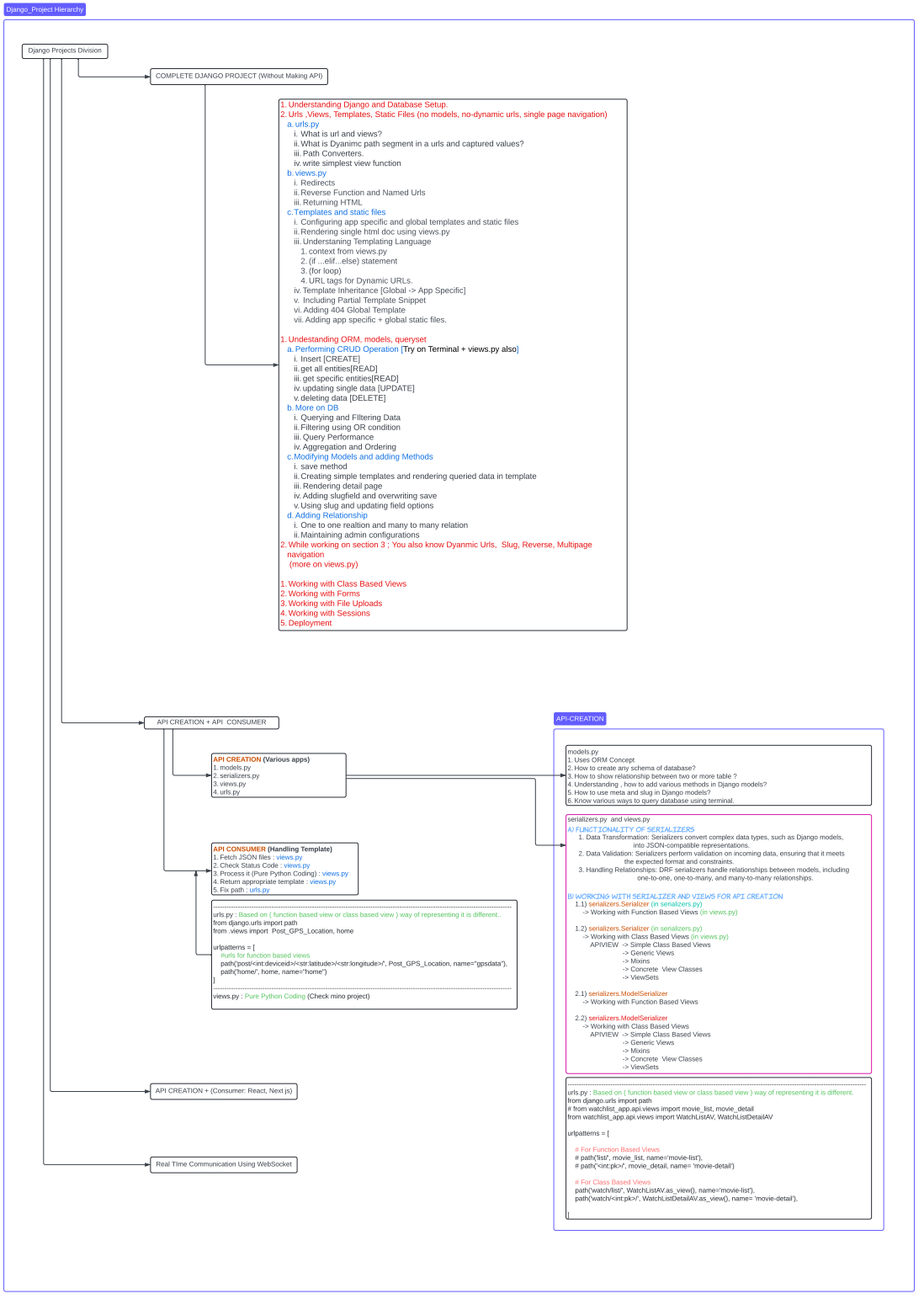
Django_Project Hierarchy
Django Projects Division
COMPLETE DJANGO PROJECT (Without Making API)
1. Understanding Django and Database Setup.
2. Urls ,Views, Templates, Static Files (no models, no-dynamic urls, single page navigation)
a. urls.py
i. What is url and views?
ii. What is Dyanimc path segment in a urls and captured values?
iii. Path Converters.
iv. write simplest view function
b. views.py
i. Redirects
ii. Reverse Function and Named Urls
iii. Returning HTML
c. Templates and static files
i. Configuring app specific and global templates and static files
ii. Rendering single html doc using views.py
iii. Understaning Templating Language
1. context from views.py
2. (if ...elif...else) statement
3. (for loop)
4. URL tags for Dynamic URLs.
iv. Template Inheritance [Global -> App Specific]
v. Including Partial Template Snippet
vi. Adding 404 Global Template
vii. Adding app specific + global static files.
1. Undestanding ORM, models, queryset
a. Performing CRUD Operation [Try on Terminal + views.py also]
i. Insert [CREATE]
ii. get all entities[READ]
iii. get specific entities[READ]
iv. updating single data [UPDATE]
v. deleting data [DELETE]
b. More on DB
i. Querying and FIltering Data
ii. Filtering using OR condition
iii. Query Performance
iv. Aggregation and Ordering
c. Modifying Models and adding Methods
i. save method
ii. Creating simple templates and rendering queried data in template
iii. Rendering detail page
iv. Adding slugfield and overwriting save
v. Using slug and updating field options
d. Adding Relationship
i. One to one realtion and many to many relation
ii. Maintaining admin configurations
2. While working on section 3 ; You also know Dyanmic Urls, Slug, Reverse, Multipage navigation
(more on views.py)
1. Working with Class Based Views
2. Working with Forms
3. Working with File Uploads
4. Working with Sessions
5. Deployment
API CREATION + API CONSUMER
API CREATION (Various apps)
1. models.py
2. serializers.py
3. views.py
4. urls.py
API CONSUMER (Handling Template)
1. Fetch JSON files : views.py
2. Check Status Code : views.py
3. Process it (Pure Python Coding) : views.py
4. Return appropriate template : views.py
5. Fix path : urls.py
urls.py : Based on (function based view or class based view) way of representing it is different..
from django.urls import path
from .views import Post_GPS_Location, home
urlpatterns = [
#urls for function based views
path('post/<int:deviceid>/<str:latitude>/<str:longitude>/', Post_GPS_Location, name="gpsdata"),
path('home/', home, name="home")
]
views.py : Pure Python Coding (Check mino project)
API CREATION + (Consumer: React, Next.js)
Real TIme Communication Using WebSocket
API-CREATION
models.py
1. Uses ORM Concept
2. How to create any schema of database?
3. How to show relationship between two or more table ?
4. Understanding , how to add various methods in Django models?
5. How to use meta and slug in Django models?
6. Know various ways to query database using terminal.
serializers.py and views.py
A) FUNCTIONALITY OF SERIALIZERS
1. Data Transformation: Serializers convert complex data types, such as Django models, into JSON-compatible representations.
2. Data Validation: Serializers perform validation on incoming data, ensuring that it meets the expected format and constraints.
3. Handling Relationships: DRF serializers handle relationships between models, including one-to-one, one-to-many, and many-to-many relationships.
B) WORKING WITH SERIALIZER AND VIEWS FOR API CREATION
1.1) serializers.Serializer (in serializers.py)
-> Working with Function Based Views (in views.py)
1.2) serializers.Serializer (in serializers.py)
-> Working with Class Based Views (in views.py)
APIVIEW -> Simple Class Based Views
-> Generic Views
-> Mixins
-> Concrete View Classes
-> ViewSets
2.1) serializers.ModelSerializer
-> Working with Function Based Views
2.2) serializers.ModelSerializer
-> Working with Class Based Views
APIVIEW -> Simple Class Based Views
-> Generic Views
-> Mixins
-> Concrete View Classes
-> ViewSets
urls.py : Based on (function based view or class based view) way of representing it is different.
from django.urls import path
# from watchlist_app.api.views import movie_list, movie_detail
from watchlist_app.api.views import WatchListAV, WatchListDetailAV
urlpatterns = [
# For Function Based Views
# path('list/', movie_list, name='movie-list'),
# path('<int:pk>/', movie_detail, name= 'movie-detail')
# For Class Based Views
path('watch/list/', WatchListAV.as_view(), name='movie-list'),
path('watch/<int:pk>/', WatchListDetailAV.as_view(), name= 'movie-detail'),
]