BACKEND revise

1. Python Fundamentals

- **Data Types**: Strings, Lists, Tuples, Sets, and Dictionaries.
- **Control Structures**: `if`, `else`, `for`, `while` loops, and list comprehensions.
- **Functions**: Defining functions, `lambda` expressions, decorators, and higher-order functions.
- **Error Handling**: `try`, `except`, `finally`, and custom exceptions.
- **File Handling**: Reading from and writing to files.
- **OOP Concepts**: Classes, objects, inheritance, polymorphism, encapsulation, and special methods (e.g., `__init__`, `__str__`).

2. Django Fundamentals

- **Django Project Structure**: Understand the basic files and directories (settings.py, urls.py, models.py, views.py, etc.).
- **MTV Architecture**: Model-Template-View architecture and how Django implements it.
- **Models**: Defining models, fields, model methods, relationships (One-to-One, One-to-Many, Many-to-Many).
- **Migrations**: Creating and applying migrations, understanding migration files.
- **Views**: Function-based views (FBVs) vs. class-based views (CBVs), understanding how to handle requests and responses.
- **Templates**: Template language, extending templates, including static files, and using template tags and filters.
- **Forms**: Django forms, ModelForm, form validation, and form handling in views.
- **Admin Interface**: Customizing the Django admin, registering models, and customizing admin views.
- **Authentication and Authorization**: Built-in user model, login, logout, user creation, permissions, and groups.

- **Middleware**: Purpose, default middlewares, and how to write custom middleware.
- **Signals**: Using signals for decoupled applications (pre-save, post-save, etc.).

3. Django REST Framework (DRF)

- **Serializers**: Understanding serializers, ModelSerializer, fields, validation, and customizing serialization.
- **Views**: APIView, viewsets, and mixins for common actions (ListModelMixin, RetrieveModelMixin, etc.).
- **Routers**: Understanding how routers work, automatic URL routing for viewsets.
- **Authentication**: Token-based authentication, JWT (JSON Web Tokens), session authentication, and custom authentication classes.
- **Permissions**: Object-level permissions, custom permission classes, and built-in permissions (IsAuthenticated, AllowAny, etc.).
- **Throttling and Rate Limiting**: How to set up and customize rate limiting.
- **Pagination**: Using pagination classes, customizing pagination responses.
- **Versioning**: URL versioning, namespace versioning, and how to manage API versions.
- **Filtering, Searching, and Ordering**: How to filter querysets, implement search functionality, and ordering in DRF.

4. Databases and ORM

- **SQL Basics**: Basic CRUD operations, joins, subqueries, indexes, transactions, and locking.
- **Django ORM**: Querysets, filters, field lookups, `annotate()`, `aggregate()`, and raw SQL queries.
- **Database Relationships**: Foreign keys, many-to-many relationships, and database normalization.
- **Performance Optimization**: Indexes, select_related, prefetch_related, and optimizing query performance.

5. Networks, Sockets, and Protocols

- **Basic Networking Concepts**: IP addresses, subnets, DNS, and firewalls.
- **HTTP Protocol**: HTTP methods (GET, POST, PUT, DELETE), status codes, headers, cookies, and sessions.
- **TCP/IP Model**: Layers of the TCP/IP model (Application, Transport, Internet, Network Interface).
- **Sockets**: Basics of socket programming, difference between TCP and UDP, and how sockets are used in web applications.
- **WebSockets**: Understanding WebSockets, real-time communication, and Django Channels.
- **RESTful Principles**: Statelessness, resource-based URLs, proper usage of HTTP methods, and HATEOAS (Hypermedia as the Engine of Application State).

6. APIs and Web Services

- **REST vs. SOAP**: Understanding the differences and use cases.
- **API Documentation**: Tools like Swagger/OpenAPI, writing clear API documentation, and why it's important.
- **JSON and XML**: Common data formats for APIs, their differences, and how they are used.
- **Rate Limiting and Caching**: Why and how to implement rate limiting, basics of caching in Django, and caching strategies.

7. Security Best Practices

- **Django Security Features**: CSRF protection, XSS protection, SQL injection prevention, and other security middleware.
- **HTTPS and SSL**: Importance of HTTPS, how SSL certificates work, and how to configure Django for HTTPS.
- **OAuth2 and OpenID Connect**: Basics of OAuth2, difference between OAuth2 and OpenID Connect, and when to use each.

8. Deployment and Environment Management

- **Deployment Basics**: How to deploy a Django application, understanding WSGI and ASGI, and using Gunicorn and Nginx.
- **Environment Configuration**: Managing different settings for different environments (development, staging, production).
- **Containerization**: Basics of Docker, creating a Dockerfile for a Django application, and running Django in a containerized environment.

9. Testing and Debugging

- **Unit Testing**: Writing tests for views, models, and forms using Django's test framework.
- **Integration Testing**: Testing the entire application stack, including APIs.
- **Debugging Tools**: Django Debug Toolbar, logging, and using `pdb` for debugging.

10. Additional Tools and Libraries

- **Celery**: Basics of task queues, how to use Celery with Django for asynchronous tasks.
- **Redis**: Using Redis as a caching backend or for task queues.
- **Django Channels**: Handling WebSockets, asynchronous views, and real-time features in Django.