Middleware – Complete Notes

What is Middleware?

Middleware is a software layer that sits between the client and the server to process requests and responses.

Where Middleware Works

1. User Request → Middleware → View → Middleware → User Response
2. Works before the view processes the request
3. Works after the view sends the response

Why Middleware is Needed

1. Avoid repeating the same code in every view
2. Handle authentication globally
3. Add security checks
4. Log requests and responses

Main Responsibilities of Middleware

1. Authentication checking
2. Security protections (CSRF, headers)
3. Logging user activity
4. Session management
5. Error handling
6. Modifying request or response

Django Middleware

In Django, middleware is written as a Python class and listed inside the MIDDLEWARE setting. It processes every request and response.

Django Middleware Execution Flow

1. Request comes in
2. Middleware runs from top to bottom
3. View executes
4. Response returns
5. Middleware runs from bottom to top

Built-in Django Middleware Examples

1. SecurityMiddleware – Adds security enhancements
2. SessionMiddleware – Manages user sessions
3. AuthenticationMiddleware – Adds user info to request
4. CsrfViewMiddleware – Protects against CSRF attacks
5. CommonMiddleware – Handles URL and response processing
6. MessageMiddleware – Supports one-time messages

Middleware vs View

1. Middleware works globally on all requests
2. Views handle specific business logic
3. Middleware runs before and after views

Advantages of Middleware

1. Code reuse
2. Centralized security
3. Cleaner project structure
4. Global request/response control

Disadvantages of Middleware

1. Bugs can affect the whole site
2. Debugging can be harder
3. Too many middleware can reduce performance

One-Line Definition:

Middleware is a processing layer between request and response that handles authentication, security, logging, and other common tasks.