Django Registration System – Practical & Interview-Ready Notes

1. What is Validation in Real-World Apps?

Validation is not just “checking inputs”. It protects the system from invalid, harmful, duplicate, or spam data. Modern apps like Instagram, Facebook, and Swiggy validate at the backend to maintain clean and secure data.

2. Why Use Middleware for Signup Validation?

Middleware acts like a security guard before the request reaches the main logic. It centralizes checks, keeps views clean, and is reusable across the project.

Benefits:

- Centralised validation
- Reusable rules
- Cleaner views
- Scalable architecture
- Prevents repeated code
- Industry-style engineering

3. Real-World Signup Flow

Request → UsernameMiddleware → EmailMiddleware → PasswordMiddleware → View → DB → Response

4. Username Validation – Practical Rules (Instagram Style)

Why apps enforce username rules:

- Avoid impersonation
- Prevent ugly/broken URLs

• Stop bots

• Maintain uniform format

Rules:

• 3–30 characters

• Only letters, numbers, dot, underscore

• Cannot start/end with dot or underscore

• Cannot contain ".." or "__"

• No spaces

• Must be unique

• Clean and URL-friendly

5. Email Validation – Real Meaning

Importance:

• Avoid fake accounts

• Reduce bounce emails

• Ensure account recovery

• Avoid duplicate users

• Prevent spambots

Rules:

• Must contain "@" and valid domain

• No spaces

• Unique (case-insensitive)

• Basic pattern validation

• Must be trustworthy format

6. Password Validation – Practical Rules

Industry password goals:

• Security

• Not guessable

• Cannot match username/email

• Protect user accounts

Rules:

• Minimum 6 characters

• Must contain lowercase

• Must contain digits

• Should not be same as username or email

• Should not be too weak

7. Why NOT Validate Inside Views?

Issues:

• Unclean and messy views

• Hard to maintain

• Code repetition

• Not scalable

Views should focus ONLY on DB operations and business logic. Validation should happen earlier → in middleware.

8. Professional Signup Architecture

Client → Middleware (validation) → View (create user) → Model → Database → Response

Each layer has a job:

Middleware = protections

View = only business logic

Model = database rules

9. Interview-Ready Answers

Q1: Why use middleware for validation?

A: To centralize and reuse validation logic. It ensures every request follows the same rules before reaching the view.

Q2: What should happen in validation middleware?

A: Format checks, missing field checks, pattern checks, duplicate checks, input filtering, low-cost security checks.

Q3: What should NOT happen in middleware?

A: Heavy logic, DB writes, sending emails, business workflows.

Q4: Why separate middlewares for username, email, password?

A: Each follows Single Responsibility Principle. Easier testing, switching, debugging.

Q5: How does middleware chaining work?

A: Django passes the request through middlewares in settings.py order. Each middleware must call get_response(request) to pass to the next one.

10. Summary

- Middleware validation makes backend clean and scalable
- Username, email, password rules ensure safety and quality
- Instagram-style layered validation is industry standard
- View handles only DB insertion, middleware handles checks