Functional vs Non-Functional Requirements

Functional Requirements

These describe what the system must do.

1. User Management & Authentication

- The system must support **user registration, login, and authentication**.
- The system must use a **custom user model**.
- Each user must be assigned **one role**: Reader, Editor, or Journalist.
- Users must be automatically added to a **group based on their role**.
- Each group must have **role-specific permissions**.

2. Roles & Permissions

- **Reader**
    - Can view approved articles.
    - Can view newsletters.
- **Editor**
    - Can view, update, and delete articles.
    - Can review and **approve or reject articles**.
- **Journalist**
    - Can create, view, update, and delete articles.
    - Can create, view, update, and delete newsletters.
    - Can publish articles independently of publishers.

3. Publishers & Articles

- The system must allow the creation of **publishers**.
- A publisher can have **multiple editors and journalists**.
- Articles must:

    - Be linked to either a **publisher or an independent journalist**.
    - Have an **approval status** (approved / not approved).
- Only **approved articles** may be visible to readers.

4. Subscriptions

- Readers must be able to:
    - Subscribe to **publishers**.
    - Subscribe to **journalists**.
- A subscription must apply to **either a publisher or a journalist**.
- Readers must receive content **only from subscribed entities**.

5. Article Approval & Notifications

- Editors must be able to **review and approve articles** via the UI.
- When an article is approved:
    - An **email notification** must be sent to all relevant subscribers.
    - The article must be **posted to an X (Twitter) account**.
- This logic must be implemented using:
    - **Django signals, or**
    - **Explicit logic in the approval view**.

6. RESTful API

- The system must expose a **RESTful API**.
- The API must allow third-party clients to:
    - Retrieve articles from **subscribed publishers**.
    - Retrieve articles from **subscribed journalists**.
- The API must:
    - Use **serializers** to convert data to JSON/XML.
    - Define **API views** to handle requests.
    - Provide **URL endpoints** for access.

7. Testing

- The system must include **unit tests** for the RESTful API.

- Tests must verify:
    - Correct retrieval of articles.
    - Subscription-based filtering.
- API endpoints must be testable using **Postman**.

8. Database

- The database must:
    - Be **normalized**.
    - Be migrated from SQLite to **MariaDB**.

Non-Functional Requirements

These describe **how well the system must work**.

1. Code Quality & Standards

- The code must follow the **PEP 8 style guide**.
- The code must be:
    - Free of syntax, runtime, and logical errors.
    - Readable and well-documented.
    - Modular, using functions and reusable components.

2. Performance & Efficiency

- The system must be implemented **efficiently**.
- Database queries should avoid unnecessary duplication.

3. Security

- Access control must be enforced:
    - Users can only access features permitted by their role.
- Defensive coding must be used to:
    - Validate user input.

    - Handle exceptions gracefully.

4. Usability (UI/UX)

- The application must provide a **clear and user-friendly interface**.
- Editors must have an intuitive workflow for **reviewing and approving articles**.
- Readers must easily:
    - Browse articles.
    - Manage subscriptions.

5. Maintainability

- The project structure must follow **Django best practices**.
- Code should be easy to extend and maintain.

6. Portfolio & Submission

- The project must be:
    - Comprehensive and professional.
    - Suitable for upload to **GitHub**.
    - Presentable to **potential employers**.