Comprehensive Blog API with DRF

You are tasked with building a sophisticated blogging API with the following requirements:

1. **User Authentication, Roles, and Permissions:**
   - Implement JWT (JSON Web Token) authentication for users.
   - Create three user roles:
     - Admin (Full access)
     - Author (Can create, update, and delete their own posts)
     - Reader (Can only read posts)
   - Implement fine-grained permissions to control access to various API endpoints.

2. **Enhanced Blog Post Model:**
   - Augment the blog post model with the following advanced features:
     - **Title:** The title of the blog post.
     - **Content:** The main textual content of the blog post.
     - **Publication Date:** Track and display the publication date of each blog post.
     - **Categories:** Introduce support for categorizing blog posts.
     - **Attachments:** Enable the inclusion of attachments (e.g., PDFs, documents) associated with a blog post.
     - **Images:** Implement support for including images linked to a blog post, enhancing visual content.

3. **API Endpoints (Class based view):**
     - Create a flexible endpoint that allows CRUD operations for blog posts:
     - **List Blog Posts:** Retrieve a list of all blog posts.
     - **Retrieve Blog Post:** Retrieve details of a specific blog post.
     - **Create Blog Post:** Allow users to create a new blog post.
     - **Update Blog Post:** Enable users to update the content, title, and other details of an existing blog post.
     - **Delete Blog Post:** Provide functionality to delete a blog post.

- Implement filtering options for the list endpoint:
    - Filter by author(s)
    - Filter by publication date range
    - Filter by categories or tags
- Implement search functionality:
    - Allow users to search for blog posts based on keywords in the title and content.

4. **Comments and Likes:**
    - Add a model for comments on blog posts.
    - Users should be able to add comments to a blog post.
    - Implement a like/dislike feature for blog posts.

5. **Error Message Translation (Arabic):**
    - Implement a system for translating error messages specifically into Arabic.
    - Ensure that error messages are provided in Arabic for a more localized user experience.

6. **Documentation:**
    - Create comprehensive API documentation using tools like Swagger or DRF's built-in documentation.

7. **Testing:**
    - Write thorough unit tests for one of the implemented functionalities.
    - Include tests for edge cases and error handling.

Bonus:

- Deploy the project on PythonAnywhere or a similar platform.
- Implement a notification system:
    - Users should receive notifications when a new comment is added to their post.
    - Readers can subscribe to receive notifications for new posts or updates to specific categories.