Medium Blog Website

(Project Brief & Backend API Documentation/Explanation)

Overview — Medium Blog API

The **Medium Blog API** is a backend-only blogging platform built using **Python Django REST Framework (DRF)**.

It allows users to create, read, update, and delete (CRUD) blog articles, manage user authentication, and perform content categorization — similar to Medium's core functionalities but in API form.

This project demonstrates complete REST API design following industry best practices including authentication, permission handling, and data serialization.

It can be used as a backend for a full-stack blog application or integrated with a frontend built using React, Vue, or any other framework.

Key Features

- **User Authentication:**
  JWT-based login and signup system using Django's built-in authentication with custom User model.

- **Article Management:**
  CRUD operations for creating, updating, deleting, and viewing articles with slug-based unique URLs.

- **Category & Tagging System:**
  Each article can belong to one or multiple categories and tags for better content organization.

- **Author Relations:**
  Each article is linked with its author through a foreign key relationship for ownership tracking.

- **Image Upload Support:**
  Media file handling for article cover images using Django's ImageField.

- **Search & Filter:**
  APIs support searching and filtering articles by title, category, author, and publication date.

- **Pagination & Ordering:**
  Optimized paginated responses and customizable ordering for listing APIs.

- **Admin Management:**
  Superusers can moderate content and manage users via the Django Admin Panel.

Tech Stack

- **Backend:** Python, Django REST Framework

- **Database:** MySQL (configurable to PostgreSQL/SQLite)

- **Authentication:** JWT (Simple JSON Web Token)

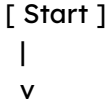
- **Tools:** Postman (API testing), Django Admin

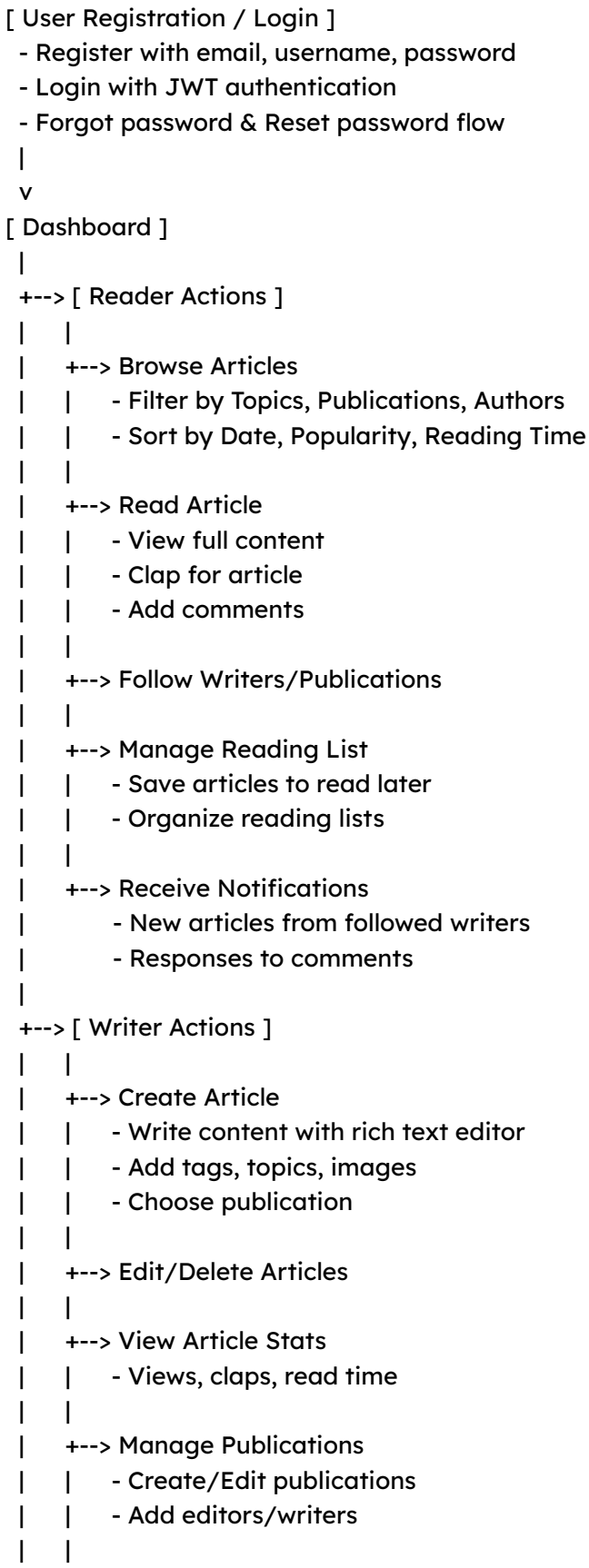
File Structure of Medium Blog Website (Backend API)

```
MEDIUM-BLOGS-RESTAPI/
├── logs/                          # Application log files
├── media/                         # User-uploaded media storage
│   ├── article_images/            # Images for articles/blogs
│   └── profiles/                  # User profile pictures
│
├── medium_blog_api_app/           # Main Django application
│   │
│   ├── articles_blogs/            # Articles & Blogs management
│   │   ├── articles_view.py       # Article CRUD operations (create, update,
delete, search)
│   │   ├── clap_and_comments.py   # Claps & comments functionality
│   │   ├── publications_and_topics.py  # Publications & topics management
│   │   └── readinglist.py         # Reading list operations
│   │
│   ├── authentication/            # Authentication system
│   │   └── custom_jwt_auth.py     # Custom JWT authentication implementation
```

```
│   │
│   ├── 📁 middleware/                  # Custom middleware
│   │   └── 📄 logging_middleware.py     # Request/response logging middleware
│   │
│   ├── 📁 migrations/                  # Database migration files
│   │
│   └── 📁 user/                       # User management app
│       ├── 📄 user_view.py             # User profile & authentication views
│       ├── 📄 __init__.py              # Package initialization
│       ├── 📄 admin.py                 # Django admin configuration
│       ├── 📄 apps.py                  # App configuration
│       ├── 📄 models.py                # User models & database schema
│       ├── 📄 serializers.py           # Data serialization for APIs
│       ├── 📄 tests.py                 # Test cases
│       ├── 📄 urls.py                  # URL routing for user endpoints
│       ├── 📄 utils.py                 # Utility functions & helpers
│       └── 📄 views.py                 # Business logic for user operations
│
├── 📁 medium_blog_api_project/        # Django project configuration
│   ├── 📄 __init__.py                  # Package initialization
│   ├── 📄 asgi.py                      # ASGI configuration for async support
│   ├── 📄 logging_config.py            # Logging configuration
│   ├── 📄 settings.py                  # Project settings & configuration
│   ├── 📄 urls.py                      # Main URL routing
│   └── 📄 wsgi.py                     # WSGI configuration for deployment
│
├── 📁 myvenv/                         # Python virtual environment
├── 📄 .gitignore                      # Git ignore rules
├── 📄 LICENSE                          # Project license
├── 📄 manage.py                        # Django management script
├── 📄 README.md                        # Project documentation
├── 📄 requirements.txt                 # Python dependencies
└── 📄 test.py                          # Test scripts
```

Medium Blog API Flow Diagram:




[User Registration / Login]
- Register with email, username, password
- Login with JWT authentication
- Forgot password & Reset password flow
[Dashboard]
[Reader Actions]
Browse Articles
- Filter by Topics, Publications, Authors
- Sort by Date, Popularity, Reading Time
Read Article
- View full content
- Clap for article
- Add comments
Follow Writers/Publications
Manage Reading List
- Save articles to read later
- Organize reading lists
Receive Notifications
- New articles from followed writers
- Responses to comments
[Writer Actions]
Create Article
- Write content with rich text editor
- Add tags, topics, images
- Choose publication
Edit/Delete Articles
View Article Stats
- Views, claps, read time
Manage Publications
- Create/Edit publications
- Add editors/writers

```
|     +--> Respond to Comments
|     |
|     +--> Notifications
|           - New claps and comments
|           - Publication updates
|
+--> [ Admin Actions ]
      |
      +--> Manage Users
      |     - Activate/Deactivate accounts
      |     - Manage roles
      |
      +--> Manage Content
      |     - Review reported articles
      |     - Manage topics/categories
      |
      +--> Analytics Dashboard
      |     - User engagement stats
      |     - Popular articles
      |     - Writer performance
```

Database Models:

- User
- Publication
- Article
- ReadingList
- Topics
- StaffPics
- ArticlePublicationTopic
- Comment
- TokenBlacklistLogout

Permissions classes:

IsAuthenticatedCustom → Basic logged-in users only
IsAdminCustom → Admin Only
IsMemberUser → Paid members, is_member = True Only

Authentication & User Management Functions:

1. **register_user()**

   """

   Register a new user with username, email, and password.

   This will create a user account in the system.

   """

   Method: POST

   Endpoint: http://127.0.0.1:8000/api/register/

   Authentication: ❌ Not Required

   Request Body Example:


```
{
  "username": "john_doe",
  "email": "john@example.com",
  "password": "StrongPass@123",
  "confirm_password": "StrongPass@123",
  "full_name": "John Doe",
  "contact_number": "1234567890",
  "bio": "Software developer and writer",
  "gender": "male",
  "profile_pic": "profile.jpg"
}
```


   Response:


```
{
    "status": "success",
    "message": "User registered successfully",
    "data": {
        "user_id": 19,
        "username": "john_doe",
        "email": "john@example.com",
        "full_name": "John Doe",
        "contact_number": "1234567890",
        "bio": "Software developer and writer",
        "profile_pic": "http://127.0.0.1:8000/media/profiles/profile.jpg",
        "followers_count": 0,
```

```
        "gender": "male"
    }
}
```


2. **login_user()**

"""
Login user using username, email or contact_number and password.
"""

Method: POST
Endpoint: http://127.0.0.1:8000/api/login/
Authentication: ❌ Not Required

Request Body Example:


```
{
  "username/email/phone_number": "john_doe",
  "password": "StrongPass@123"
}
```


Response:


```
{
    "status": "success",
    "message": "Login successful",
    "data": {
        "User": {
            "username": "john_doe",
            "email": "john@example.com",
            "full_name": "John Doe"
        },
        "access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzYyMzIwNzU5LCJpYXQiOjE3NjIyMzQzNTksImp0aSI6IjY2NTZiYWMyYjVjNzQxOT..."
    }
}
```


3. **logout_user()**

"""

Logout user from the system and add the access token to the blacklist.

"""

Method: POST

Endpoint: http://127.0.0.1:8000/api/logout/

Authentication: ✅ Required (JWT Token)

Response:

```
{
   "status": "success",
   "message": "Logout successful"
}
```

4. **edit_profile()**

"""

Edit user profile information.

"""

Method: PUT

Endpoint: http://127.0.0.1:8000/api/edit-profile/

Authentication: ✅ Required (JWT Token)

Request Body Example:

```
{
   "username": "john_doe_updated",
   "email": "john.updated@example.com",
   "full_name": "John Doe Updated",
   "contact_number": "9876543210",
   "bio": "Updated bio information",
   "gender": "male",
   "profile_pic": "new_profile.jpg",
   "remove_profile_pic": false
}
```

Response:

```
{
```

```
  "status": "success",
  "message": "Profile updated successfully",
  "data": {
    "user_id": 19,
    "username": "john_doe_updated",
    "email": "john.updated@example.com",
    "full_name": "John Doe Updated",
    "contact_number": "9876543210",
    "profile_pic": "http://127.0.0.1:8000/media/profiles/new_profile.jpg",
    "bio": "Updated bio information",
    "gender": "male"
  }
}
```


5. **delete_profile()**

   """

   Permanently delete user profiles from the system.

   """

   Method: DELETE

   Endpoint: http://127.0.0.1:8000/api/delete-profile/

   Authentication: ✅ Required (JWT Token)

   Request Body:


```
{
  "user_id": 19
}
```


   Response:


```
{
  "status": "success",
  "message": "Profile deleted successfully"
}
```


6. **deactivate_profile()**

   """

Deactivate user account (soft delete).
"""

Method: POST
Endpoint: http://127.0.0.1:8000/api/deactivate-profile/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "deactivate": true
}
```

Response:

```
{
  "status": "success",
  "message": "Profile deactivated successfully"
}
```

7. **change_password()**

"""
Change user password while logged in.
"""

Method: POST
Endpoint: http://127.0.0.1:8000/api/change-password/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "old_password": "OldPass@123",
  "new_password": "NewStrongPass@123",
  "confirm_password": "NewStrongPass@123"
}
```

Response:

```
{
  "status": "success",
  "message": "Password changed successfully"
```

```
}
```

8. **forgot_password()**

   """
   Send OTP to email for password reset.
   """

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/forgot-password/
   Authentication: ❌ Not Required

   Request Body:

```
{
  "email": "john@example.com"
}
```

   Response:

```
{
  "status": "success",
  "message": "OTP sent to email id: john@example.com"
}
```

9. **reset_password()**

   """
   Reset password using OTP received via email.
   """

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/reset-password/
   Authentication: ❌ Not Required

   Request Body:

```
{
  "email": "john@example.com",
  "otp": "123456",
  "new_password": "NewStrongPass@123",
  "confirm_new_password": "NewStrongPass@123"
}
```

Response:

```
{
  "status": "success",
  "message": "Password changed successfully"
}
```

10. **follow_user()**

    """
    Follow another user in the system.
    """

    Method: POST
    Endpoint: http://127.0.0.1:8000/api/follow-user/
    Authentication: ✅ Required (JWT Token)

    Request Body:

```
{
  "user_id": 20,
  "mark_as_following": true
}
```

    Response:

```
{
  "status": "success",
  "message": "You are now following jane_doe"
}
```

11. **unfollow_user()**

    """
    Unfollow a user you are currently following.
    """

    Method: POST
    Endpoint: http://127.0.0.1:8000/api/unfollow-user/
    Authentication: ✅ Required (JWT Token)

    Request Body:

```
{
  "user_id": 20,
  "mark_as_unfollow": true
}
```

Response:

```
{
  "status": "success",
  "message": "You unfollowed jane_doe"
}
```

12. **view_my_profile()**

"""
View current logged-in user's profile details.
"""

Method: GET
Endpoint: http://127.0.0.1:8000/api/my-profile/
Authentication: ✅ Required (JWT Token)

Response:

```
{
  "status": "success",
  "message": "Profile fetched successfully",
  "data": {
    "user_id": 19,
    "username": "john_doe",
    "email": "john@example.com",
    "full_name": "John Doe",
    "contact_number": "1234567890",
    "bio": "Software developer and writer",
    "profile_pic": "http://127.0.0.1:8000/media/profiles/profile.jpg",
    "followers_count": 15
  }
}
```

13. **view_other_user_profile()**

"""

View another user's public profile details.

"""

Method: GET

Endpoint: http://127.0.0.1:8000/api/user-profile/

Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "user_id": 20
}
```

Response:

```
{
  "status": "success",
  "message": "Profile fetched successfully",
  "data": {
    "user_id": 20,
    "username": "jane_doe",
    "email": "jane@example.com",
    "full_name": "Jane Doe",
    "contact_number": "0987654321",
    "bio": "Content writer and blogger",
    "profile_pic": "http://127.0.0.1:8000/media/profiles/jane_profile.jpg",
    "followers_count": 25
  }
}
```

14. **view_my_following_list()**

"""

View list of users and publications the current user is following.

"""

Method: GET

Endpoint: http://127.0.0.1:8000/api/following-list/

Authentication: ✅ Required (JWT Token)

Response:


```
{
  "status": "success",
  "message": "Following list fetched",
  "results": {
    "users": [
      {
        "user_id": 20,
        "username": "jane_doe",
        "full_name": "Jane Doe",
        "profile_pic": "http://127.0.0.1:8000/media/profiles/jane_profile.jpg"
      }
    ],
    "publications": [
      {
        "publication_id": 5,
        "publication_title": "Tech Insights",
        "logo_image":
"http://127.0.0.1:8000/media/publications/tech_logo.jpg",
        "owner": "tech_writer"
      }
    ]
  }
}
```


Articles:

1. **create_article()**

   """

   Create an article.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/create/
   Authentication: ✅ Required (JWT Token)

   Request Body:

```
{
  "article_title": "Introduction to Python Programming",
  "article_content": "Python is a high-level programming language...",
  "article_subtitle": "Learn Python from scratch",
  "article_category": "Programming",
  "publication_id": 5,
  "url": "https://example.com/python-intro",
  "image": "file",
  "video": "https://youtube.com/embed/xyz",
  "code_block": "print('Hello World')",
  "is_member_only": false,
  "topics": ["python", "programming", "tutorial"],
  "allow_to_share_article": true
}
```


Response:


```
{
  "status": "success",
  "message": "Article created successfully",
  "data": {
    "article_id": 25,
    "article_title": "Introduction to Python Programming",
    "article_subtitle": "Learn Python from scratch",
    "article_category": "Programming",
    "publication": 5,
    "url": "https://example.com/python-intro",
    "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
    "video": "https://youtube.com/embed/xyz",
    "code_block": "print('Hello World')",
    "read_time": 5,
    "is_member_only": false,
    "allow_to_share_article": true,
    "published_by": 19,
    "updated_by": 19
  }
}
```

2. **update_article()**

"""

Update article details.

"""

Method: PUT

Endpoint: http://127.0.0.1:8000/api/update/

Authentication: ✅ Required (JWT Token)

Request Body:


```
{
  "article_id": 25,
  "article_title": "Updated Python Programming Guide",
  "article_subtitle": "Complete Python tutorial for beginners",
  "article_content": "Updated content with more examples...",
  "article_category": "Programming Tutorials",
  "url": "https://example.com/updated-python",
  "video": "https://youtube.com/embed/abc",
  "code_block": "def hello():\n    print('Hello World')",
  "is_member_only": true,
  "topics": [1, 2, 3],
  "image": "file",
  "allow_to_share_article": false
}
```


Response:


```
{
  "status": "success",
  "message": "Article updated successfully",
  "data": {
    "article_id": 25,
    "article_title": "Updated Python Programming Guide",
    "article_subtitle": "Complete Python tutorial for beginners",
    "article_category": "Programming Tutorials",
    "publication": 5,
    "url": "https://example.com/updated-python",
```

```
        "image":
"http://127.0.0.1:8000/media/article_images/updated_python.jpg",
        "video": "https://youtube.com/embed/abc",
        "code_block": "def hello():\n    print('Hello World')",
        "read_time": 8,
        "is_member_only": true,
        "allow_to_share_article": false,
        "published_by": 19,
        "updated_by": 19
    }
}
```


3. **delete_article()**

   """

   Delete article (Only author / publication owner / admin).

   """

   Method: DELETE

   Endpoint: http://127.0.0.1:8000/api/delete/

   Authentication: ✅ Required (JWT Token)

   Request Body:


```
{
   "article_id": 25
}
```


   Response:


```
{
   "status": "success",
   "message": "Article deleted"
}
```


4. **search_articles()**

   """

   Search articles by title or content.

   """

   Method: POST

Endpoint: http://127.0.0.1:8000/api/search/
Authentication: ❌ Not Required

Request Body:


```
{
  "search_by_title": "Python programming"
}
```


Response:


```
{
  "status": "success",
  "message": "Articles found",
  "results": [
    {
      "article_id": 25,
      "article_title": "Introduction to Python Programming",
      "article_subtitle": "Learn Python from scratch",
      "article_content": "Python is a high-level programming language...",
      "article_category": "Programming",
      "url": "https://example.com/python-intro",
      "video": "https://youtube.com/embed/xyz",
      "code_block": "print('Hello World')",
      "is_member_only": false,
      "allow_to_share_article": true,
      "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "created_by": "john_doe",
      "updated_by": "john_doe",
      "published_at": "2024-01-15T10:30:00Z",
      "published_by": "john_doe",
      "clap_count": 15
    }
  ]
}
```

5. **get_my_articles()**

   """

   Show all articles published by the user.

   """

   Method: GET

   Endpoint: http://127.0.0.1:8000/api/my-articles/

   Authentication: ✅ Required (JWT Token)

   Response:


```
{
  "status": "success",
  "message": "Articles found",
  "results": [
    {
      "article_id": 25,
      "article_title": "Introduction to Python Programming",
      "article_subtitle": "Learn Python from scratch",
      "article_content": "Python is a high-level programming language...",
      "article_category": "Programming",
      "url": "https://example.com/python-intro",
      "video": "https://youtube.com/embed/xyz",
      "code_block": "print('Hello World')",
      "is_member_only": false,
      "allow_to_share_article": true,
      "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "created_by": "john_doe",
      "updated_by": "john_doe",
      "published_at": "2024-01-15T10:30:00Z",
      "published_by": "john_doe",
      "clap_count": 15
    }
  ]
}
```

6. **get_all_articles()**

   """

   Get all articles from all users for the dashboard.

   """

   Method: GET

   Endpoint: http://127.0.0.1:8000/api/all/

   Authentication: ❌ Not Required

   Response:


```
{
   "status": "success",
   "message": "Articles fetched",
   "results": [
      {
         "article_id": 25,
         "title": "Introduction to Python Programming",
         "subtitle": "Learn Python from scratch",
         "author": "john_doe",
         "publication": "Tech Insights",
         "is_member_only": false,
         "message": null,
         "article_content": "Python is a high-level programming language...",
         "published_at": "2024-01-15T10:30:00Z",
         "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
         "url": "https://example.com/python-intro",
         "read_time": 5,
         "clap_count": 15,
         "comment_count": 3,
         "is_shared": false,
         "original_article_id": null,
         "total_shared": 2
      }
   ]
}
```


7. **report_article()**

"""

Report an article (is_reported = True).

"""

Method: POST

Endpoint: http://127.0.0.1:8000/api/report/

Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "article_id": 25,
  "report_article": true
}
```

Response:

```
{
  "status": "success",
  "message": "Article reported"
}
```

8. **get_reported_articles()**

"""

Get all reported articles (Admin only).

"""

Method: GET

Endpoint: http://127.0.0.1:8000/api/reported/

Authentication: ✅ Required (Admin JWT Token)

Response:

```
{
  "status": "success",
  "message": "Articles fetched",
  "data": [
    {
      "article_id": 25,
      "title": "Introduction to Python Programming",
      "subtitle": "Learn Python from scratch",
```

```
            "author": "john_doe",
            "publication": "Tech Insights",
            "is_member_only": false,
            "message": null,
            "article_content": "Python is a high-level programming language...",
            "published_at": "2024-01-15T10:30:00Z",
            "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
            "url": "https://example.com/python-intro",
            "read_time": 5,
            "clap_count": 15,
            "comment_count": 3,
            "is_shared": false,
            "original_article_id": null,
            "total_shared": 2
        }
    ]
}
```


9. **mute_author()**

    """
    Mute/Unmute an author.
    """

    Method: POST
    Endpoint: http://127.0.0.1:8000/apimute-author/
    Authentication: ✅ Required (JWT Token)

    Request Body:


```
{
    "author_id": 20,
    "mute_author": true
}
```


    Response:


```
{
    "status": "success",
    "message": "Author: jane_doe has been muted."
```

}

10. **mute_publication()**

"""

Mute/Unmute a publication.

"""

Method: POST

Endpoint: http://127.0.0.1:8000/api/mute-publication/

Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "publication_id": 5,
  "mute_publication": true
}
```

Response:

```
{
  "status": "success",
  "message": "Publication muted"
}
```

11. **show_less_like_this_func()**

"""

Mark an article as 'show less like this' or unmark it.

"""

Method: POST

Endpoint: http://127.0.0.1:8000/api/show-less/

Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "article_id": 25,
  "show_less_like_this": true
}
```

Response:


```
{
  "status": "success",
  "message": "Show less like this turned on"
}
```


12. share_article()

"""
Share an article.
"""

Method: POST
Endpoint: http://127.0.0.1:8000/apishare/
Authentication: ✅ Required (JWT Token)

Request Body:


```
{
  "article_id": 25
}
```


Response:


```
{
  "status": "success",
  "message": "Article shared successfully",
  "shared_article": {
    "article_id": 26,
    "article_title": "Introduction to Python Programming",
    "article_subtitle": "Learn Python from scratch",
    "article_content": "Python is a high-level programming language...",
    "article_category": "Programming",
    "url": "https://example.com/python-intro",
    "video": "https://youtube.com/embed/xyz",
    "code_block": "print('Hello World')",
    "is_member_only": false,
    "published_by": 19,
    "shared_from": 25,
    "shared_by": 19,
```

```
    "allow_to_share_article": true,
    "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
    "created_at": "2024-01-15T11:30:00Z",
    "updated_at": "2024-01-15T11:30:00Z",
    "published_at": "2024-01-15T11:30:00Z",
    "read_time": 5,
    "clap_count": 0
  }
}
```


13. **undo_reshare()**

    """
    Undo reshare an article.
    """

    Method: POST
    Endpoint: http://127.0.0.1:8000/api/undo-reshare/
    Authentication: ✅ Required (JWT Token)

    Request Body:


```
{
  "article_id": 26,
  "undo_reshare": true
}
```


    Response:


```
{
  "status": "success",
  "message": "Undo reshare turned on"
}
```


14. **get_shared_articles()**

    """
    Get all shared articles.
    """

    Method: GET
    Endpoint: http://127.0.0.1:8000/api/shared/

Authentication: ✅ Required (JWT Token)

Response:


```
{
  "status": "success",
  "message": "Articles found",
  "results": [
    {
      "article_id": 26,
      "article_title": "Introduction to Python Programming",
      "article_subtitle": "Learn Python from scratch",
      "article_content": "Python is a high-level programming language...",
      "author": "john_doe",
      "publication": "Tech Insights",
      "read_time": "5 min",
      "clap_count": "0",
      "comment_count": 0,
      "published_at": "2024-01-15T11:30:00Z",
      "is_member_only": false,
      "shared_from": 25,
      "shared_by": 19,
      "allow_to_share_article": true,
      "image": "http://127.0.0.1:8000/media/article_images/python.jpg",
      "created_at": "2024-01-15T11:30:00Z",
      "updated_at": "2024-01-15T11:30:00Z",
      "published_at": "2024-01-15T11:30:00Z"
    }
  ]
}
```


Clap, Comments & Reading lists :

1. **give_clap()**

   """

   Give a clap to an article.

Method: POST
Endpoint: http://127.0.0.1:8000/api/clap/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "article_id": 25
}
```

Response:

```
{
  "status": "success",
  "message": "Clapped"
}
```

2. **remove_clap()**
   """
   Remove a clap from an article.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/remove-clap/
   Authentication: ✅ Required (JWT Token)

   Request Body:

   ```
   {
     "article_id": 25
   }
   ```

   Response:

   ```
   {
     "status": "success",
     "message": "Clap removed"
   }
   ```

3. **add_comment()**
   """
   Add a comment to an article.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/add-comment/

Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "article_id": 25,
  "content": "This is a great article! Very informative."
}
```

Response:

```
{
  "status": "success",
  "message": "Comment added",
  "data": {
    "comment_id": 12,
    "article": 25,
    "user": 19,
    "comment_content": "This is a great article! Very informative.",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T10:30:00Z"
  }
}
```

4. **edit_comment()**
   """
   Edit a comment.

   Method: PUT
   Endpoint: http://127.0.0.1:8000/api/edit-comment/
   Authentication: ✅ Required (JWT Token)

   Request Body:

   ```
   {
     "comment_id": 12,
     "content": "Updated comment: This article is excellent!"
   }
   ```

   Response:

   ```
   {
     "status": "success",
     "message": "Comment updated",
     "data": {
   ```

```
    "comment_id": 12,
    "article": 25,
    "user": 19,
    "comment_content": "Updated comment: This article is excellent!",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T11:30:00Z"
  }
}
```


5. **remove_comment()**
   """
   Remove a comment from an article.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/remove-comment/
   Authentication: ✅ Required (JWT Token)

   Request Body:

```
{
  "comment_id": 12
}
```


   Response:

```
{
  "status": "success",
  "message": "Comment removed"
}
```


6. **create_readinglist()**
   """
   Create a reading list - Add article to reading list.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/create/
   Authentication: ✅ Required (JWT Token)

   Request Body:

```
{
  "article_id": 25,
  "visibility": "public"
}
```

Response:


```
{
  "status": "success",
  "message": "Readinglist created",
  "data": {
    "reading_list_id": 8,
    "article_id": 25,
    "article_title": "Introduction to Python Programming",
    "article_category": "Programming",
    "visibility": "public",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T10:30:00Z",
    "created_by": 19,
    "updated_by": 19
  }
}
```


7. **get_readinglist()**

"""

Get the user's reading list.

Method: GET
Endpoint: http://127.0.0.1:8000/api/my-list/
Authentication: ✅ Required (JWT Token)

Response:


```
{
  "status": "success",
  "message": "Readinglist found",
  "data": [
    {
      "reading_list_id": 8,
      "article_id": 25,
      "article_title": "Introduction to Python Programming",
      "article_category": "Programming",
      "visibility": "public",
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "created_by": 19,
      "updated_by": 19,
```

```
            "article_details": {
                "author": "john_doe",
                "read_time": 5,
                "clap_count": 15,
                "image": "http://127.0.0.1:8000/media/article_images/python.jpg"
            }
        },
        {
            "reading_list_id": 9,
            "article_id": 30,
            "article_title": "Advanced JavaScript Concepts",
            "article_category": "Web Development",
            "visibility": "private",
            "created_at": "2024-01-14T15:20:00Z",
            "updated_at": "2024-01-14T15:20:00Z",
            "created_by": 19,
            "updated_by": 19,
            "article_details": {
                "author": "jane_doe",
                "read_time": 8,
                "clap_count": 25,
                "image": "http://127.0.0.1:8000/media/article_images/javascript.jpg"
            }
        }
    ]
}
```


8. **edit_readinglist()**

"""

Edit reading list visibility.

Method: PUT
Endpoint: http://127.0.0.1:8000/api/edit/
Authentication: ✅ Required (JWT Token)

Request Body:


```
{
    "readinglist_id": 8,
    "visibility": "private"
}
```

Response:


```
{
  "status": "success",
  "message": "Readinglist updated",
  "data": {
    "reading_list_id": 8,
    "article_id": 25,
    "article_title": "Introduction to Python Programming",
    "article_category": "Programming",
    "visibility": "private",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T11:30:00Z",
    "created_by": 19,
    "updated_by": 19
  }
}
```


9. **delete_readinglist()**

"""

Remove articles from the reading list.

Method: DELETE
Endpoint: http://127.0.0.1:8000/api/delete/
Authentication: ✅ Required (JWT Token)
Request Body:


```
{
  "readinglist_id": 8
}
```


Response:


```
{
  "status": "success",
  "message": "Readinglist deleted"
}
```


10. **add_multiple_to_readinglist()**

"""

Add multiple articles to the reading list at once.

Method: POST
Endpoint: http://127.0.0.1:8000/api/readinglist/add-multiple/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "article_ids": [25, 30, 35],
  "visibility": "public"
}
```

Response (Success):

```
{
  "status": "success",
  "message": "3 articles added to reading list",
  "data": {
    "added_count": 3,
    "skipped_count": 0,
    "total_articles_in_list": 15
  }
}
```

11. **clear_readinglist()**

"""

Clear the entire reading list.

Method: DELETE
Endpoint: http://127.0.0.1:8000/api/readinglist/clear-all/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "readinglist_id": 8
}
```

Response (Success):

```
{
  "status": "success",
  "message": "Reading list cleared successfully",
  "data": {
    "deleted_count": 12,
    "total_articles_removed": 12
  }
}
```

12. search_readinglist()

"""

Search articles in the reading list by title, content, or category.

Method: POST
Endpoint: http://127.0.0.1:8000/api/readinglist/search/
Authentication: ✅ Required (JWT Token)

Request Body:


```
{
  "search_text": "python"
}
```


Response:


```
{
  "status": "success",
  "message": "Search results found",
  "data": [
    {
      "reading_list_id": 8,
      "article_id": 25,
      "article_title": "Introduction to Python Programming",
      "article_category": "Programming",
      "visibility": "public",
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "match_reason": "Title contains search text, Content contains search text"
    },
    {
      "reading_list_id": 12,
      "article_id": 30,
      "article_title": "Python Data Analysis",
      "article_category": "Data Science",
      "visibility": "private",
      "created_at": "2024-01-14T15:20:00Z",
      "updated_at": "2024-01-14T15:20:00Z",
      "match_reason": "Title contains search text, Category contains search
text"
    }
  ]
}
```

13. **get_readinglist_stats()**

"""

Get reading list statistics and analytics.

Method: GET
Endpoint: http://127.0.0.1:8000/api/readinglist/stats/
Authentication: ✅ Required (JWT Token)

Response (Success):


```
{
  "status": "success",
  "message": "Reading list stats fetched",
  "data": {
    "total_articles": 15,
    "public_articles": 10,
    "private_articles": 5,
    "total_read_time": 125,
    "categories": {
      "Programming": 6,
      "Web Development": 4,
      "Data Science": 3,
      "Design": 2
    },
    "last_added": "2024-01-15T10:30:00Z"
  }
}
```


Response (Empty reading list):


```
{
  "status": "success",
  "message": "Reading list stats fetched",
  "data": {
    "total_articles": 0,
    "public_articles": 0,
    "private_articles": 0,
    "total_read_time": 0,
    "categories": {},
    "last_added": null
  }
}
```

Topics :

1. **create_topic()**
   """
   Create topic (Admin only).

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/create-topic/
   Authentication: ✅ Required (Admin JWT Token)

   Request Body:
   ```
   {
     "topic_header_1": "Technology",
     "topic_header_2": "Programming",
     "topic_header_3": "Web Development",
     "topic_description": "All about technology and programming"
   }
   ```

   Response:
   ```
   {
     "status": "success",
     "message": "Topic created",
     "data": {
       "topic_id": 8,
       "topic_header_1": "Technology",
       "topic_header_2": "Programming",
       "topic_header_3": "Web Development",
       "topic_description": "All about technology and programming",
       "total_stories": 0,
       "total_followers": 0,
       "created_at": "2024-01-15T10:30:00Z",
       "updated_at": "2024-01-15T10:30:00Z",
       "created_by": 1,
       "updated_by": 1
     }
   }
   ```

2. **edit_topic()**

"""
Edit topic (Admin only).

Method: PUT
Endpoint: http://127.0.0.1:8000/api/edit-topic/
Authentication: ✅ Required (Admin JWT Token)

Request Body:

```
{
  "topic_id": 8,
  "topic_header_1": "Tech & Innovation",
  "topic_header_2": "Software Development",
  "topic_header_3": "Full Stack Development",
  "topic_description": "Latest in technology and software development trends"
}
```


Response:

```
{
  "status": "success",
  "message": "Topic updated",
  "data": {
    "topic_id": 8,
    "topic_header_1": "Tech & Innovation",
    "topic_header_2": "Software Development",
    "topic_header_3": "Full Stack Development",
    "topic_description": "Latest in technology and software development
trends",
    "total_stories": 15,
    "total_followers": 120,
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T11:30:00Z",
    "created_by": 1,
    "updated_by": 1
  }
}
```


3. **delete_topic()**
"""
Delete topic (Admin only).

Method: DELETE

Endpoint: http://127.0.0.1:8000/api/delete-topic/
Authentication: ✅ Required (Admin JWT Token)

Request Body:

```
{
  "topic_id": 8
}
```

Response:

```
{
  "status": "success",
  "message": "Topic deleted"
}
```

4. **view_all_topics()**
   """
   View all topics.

   Method: GET
   Endpoint: http://127.0.0.1:8000/api/topics/
   Authentication: ✅ Required (JWT Token)

   Response:

```
{
  "status": "success",
  "message": "Topics fetched successfully",
  "data": [
    {
      "topic_id": 1,
      "topic_header_1": "Technology",
      "topic_header_2": "Programming",
      "topic_header_3": "Web Development",
      "topic_description": "All about technology and programming",
      "total_stories": 45,
      "total_followers": 230,
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "created_by": 1,
      "updated_by": 1
    }
  ]
```

```
}
```

5. **search_topics()**
   """
   Search topics by headers or description.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/search-topics/
   Authentication: ✅ Required (JWT Token)

   Request Body:
   ```
   {
     "enter_text": "technology"
   }
   ```

   Response:
   ```
   {
     "status": "success",
     "message": "Topics fetched successfully",
     "data": [
       {
         "topic_id": 1,
         "topic_header_1": "Technology",
         "topic_header_2": "Programming",
         "topic_header_3": "Web Development",
         "topic_description": "All about technology and programming",
         "total_stories": 45,
         "total_followers": 230,
         "created_at": "2024-01-15T10:30:00Z",
         "updated_at": "2024-01-15T10:30:00Z",
         "created_by": 1,
         "updated_by": 1
       }
     ]
   }
   ```

6. **view_specific_topic()**
   """
   View Specific Topic.

   Method: POST

Endpoint: http://127.0.0.1:8000/api/view-specific-topic/
Authentication: ✅ Required (Admin JWT Token)

Request Body:

```
{
  "topic_id": 8
}
```

Response:

```
{
  "status": "success",
  "message": "Topic fetched successfully",
  "data": [
    {
      "topic_id": 1,
      "topic_header_1": "Technology",
      "topic_header_2": "Programming",
      "topic_header_3": "Web Development",
      "topic_description": "All about technology and programming",
      "total_stories": 45,
      "total_followers": 230,
      "created_at": "2024-01-15T10:30:00Z",
      "updated_at": "2024-01-15T10:30:00Z",
      "created_by": 1,
      "updated_by": 1
    }
  ]
}
```

Publications :

1. **create_publication()**
   """
   Create publications.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/create/
   Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "publication_title": "Tech Insights",
  "topic_id": 1,
  "short_note": "Latest technology trends and insights",
  "logo_image": "file",
  "topics_of_publications": "1,2,3",
  "is_public": true,
  "default_article_visibility": "public"
}
```

Response:

```
{
  "status": "success",
  "message": "Publication created",
  "data": {
    "publication_id": 5,
    "publication_title": "Tech Insights",
    "topic": 1,
    "owner": 19,
    "short_note": "Latest technology trends and insights",
    "logo_image": "http://127.0.0.1:8000/media/publications/tech_logo.jpg",
    "topics_of_publications": "1,2,3",
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T10:30:00Z",
    "created_by": 19,
    "updated_by": 19
  }
}
```

2. **edit_publication()**

"""

Edit publication (Owner/Editor/Writer only).

Method: PUT
Endpoint: http://127.0.0.1:8000/api/edit/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
  "publication_id": 5,
```

```
    "publication_title": "Tech Insights Updated",
    "topic_id": 2,
    "short_note": "Updated description about technology",
    "logo_image": "file",
    "topics_of_publications": "1,2,3,4",
    "default_article_visibility": "members"
}
```

Response:

```
{
    "status": "success",
    "message": "Publication updated",
    "data": {
        "publication_id": 5,
        "publication_title": "Tech Insights Updated",
        "topic": 2,
        "owner": 19,
        "short_note": "Updated description about technology",
        "logo_image":
"http://127.0.0.1:8000/media/publications/updated_logo.jpg",
        "topics_of_publications": "1,2,3,4",
        "created_at": "2024-01-15T10:30:00Z",
        "updated_at": "2024-01-15T11:30:00Z",
        "created_by": 19,
        "updated_by": 19
    }
}
```

3. **delete_publication()**

"""

Delete publication (Owner only).

Method: DELETE
Endpoint: http://127.0.0.1:8000/apidelete/
Authentication: ✅ Required (JWT Token)

Request Body:

```
{
    "publication_id": 5
}
```

Response:

```
{
   "status": "success",
   "message": "Publication deleted"
}
```

4. **view_publications()**

   """
   View all publications.

   Method: GET
   Endpoint: http://127.0.0.1:8000/api/all/
   Authentication: ✅ Required (JWT Token)

   Response:

```
{
   "status": "success",
   "message": "Publications found",
   "data": [
      {
         "publication_id": 5,
         "publication_title": "Tech Insights",
         "owner": "john_doe",
         "created_at": "2024-01-15T10:30:00Z",
         "updated_at": "2024-01-15T10:30:00Z"
      }
   ]
}
```

5. **follow_publication()**

   """
   Follow a publication.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/follow/
   Authentication: ✅ Required (JWT Token)

   Request Body:

```
{
   "publication_id": 5,
```

```
    "mark_publications_followed": "true"
}
```

Response:

```
{
    "status": "success",
    "message": "You are now following 'Tech Insights'"
}
```

6. **unfollow_publication()**
   """
   Unfollow a publication.

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/unfollow/
   Authentication: ✅ Required (JWT Token)

   Request Body:

```
{
    "publication_id": 5,
    "mark_publications_unfollow": "true"
}
```

   Response:

```
{
    "status": "success",
    "message": "You unfollowed 'Tech Insights'"
}
```

7. **add_staff_picks()**
   """
   Add an article to staff picks (Admin only).

   Method: POST
   Endpoint: http://127.0.0.1:8000/api/add-staff-picks/
   Authentication: ✅ Required (Admin JWT Token)

   Request Body:

```
{
    "publication_id": 5,
    "article_id": 25,
```

```
  "topic_id": 1
}
```


Response:


```
{
  "status": "success",
  "message": "Article added to staff picks"
}
```


8. **edit_staff_picks()**

"""

Edit staff picks (Admin only).

Method: PUT
Endpoint: http://127.0.0.1:8000/api/edit-staff-picks/
Authentication: ✅ Required (Admin JWT Token)

Request Body:


```
{
  "publication_id": 5,
  "article_id": 25,
  "topic_id": 2
}
```


Response:


```
{
  "status": "success",
  "message": "Staff pick updated successfully",
  "updated_data": {
    "staff_pic_id": 3,
    "article_id": 25,
    "article_title": "Introduction to Python Programming",
    "topic_id": 2,
    "topic_name": "Programming",
    "publication_id": 5,
    "publication_name": "Tech Insights",
    "total_stories": 45.0,
    "total_saves": 120.0,
    "created_at": "2024-01-15T10:30:00Z",
    "updated_at": "2024-01-15T11:30:00Z",
    "created_by": 1,
```

```
        "updated_by": 1
    }
}
```

9. **remove_staff_pick_field()**
   """
   Remove fields from staff picks (Admin only).

   Method: PUT
   Endpoint: http://127.0.0.1:8000/api/remove-staff-pick/
   Authentication: ✅ Required (Admin JWT Token)

   Request Body:
   ```
   {
      "staff_pic_id": 3,
      "remove_article": true,
      "remove_topic": false,
      "remove_publication": false
   }
   ```

   Response:
   ```
   {
      "status": "success",
      "message": "Staff pick field removed successfully"
   }
   ```

10. **remove_staff_pick_field()**
    """
    Remove fields from staff picks (Admin only).

    Method: PUT
    Endpoint: http://127.0.0.1:8000/api/remove-staff-pick/
    Authentication: ✅ Required (Admin JWT Token)

    Request Body:
    ```
    {
       "staff_pic_id": 3,
       "remove_article": true,
       "remove_topic": false,
       "remove_publication": false
    }
    ```

Response:


```
{
  "status": "success",
  "message": "Staff pick field removed successfully"
}
```


11. **view_all_staff_picks()**

    """

    View all staff picks articles.

    Method: GET

    Endpoint: http://127.0.0.1:8000/api/staff-picks/

    Authentication: ✅ Required (JWT Token)

    Request Body:


```
{
  "publication_id": 5,
  "article_id": 25,
  "topic_id": 1
}
```


    Response:


```
{
  "status": "success",
  "message": "staff_picks fetched successfully",
  "data": [
    {
      "staff_pic_id": 3,
      "article_id": 25,
      "article_title": "Introduction to Python Programming",
      "topic_id": 1,
      "topic_name": "Technology",
      "publication_id": 5,
      "publication_name": "Tech Insights",
```

```
        "total_stories": 45.0,
        "total_saves": 120.0,
        "created_at": "2024-01-15T10:30:00Z",
        "updated_at": "2024-01-15T10:30:00Z",
        "created_by": "admin_user",
        "updated_by": "admin_user"
      }
    ]
}
```