API Documentation Configuration

Overview

This document outlines the comprehensive API documentation setup for the Academic Paper Generator application using Django REST Framework's drf-spectacular package. The documentation provides complete Swagger UI integration with detailed endpoint descriptions, field documentation, and interactive API exploration capabilities.

Swagger Configuration

The application uses drf-spectacular for automatic OpenAPI schema generation and Swagger UI integration. The configuration is defined in the Django settings and provides comprehensive API documentation for all endpoints.

Settings Configuration

The Swagger documentation is configured in `config/settings.py` with the following specifications:

```
SPECTACULAR_SETTINGS = {
    'TITLE': 'Academic Paper Generator API',
    'DESCRIPTION': 'Comprehensive API for generating academic papers using AI technology',
    'VERSION': '1.0.0',
    'SERVE_INCLUDE_SCHEMA': False,
    'COMPONENT_SPLIT_REQUEST': True,
    'SCHEMA_PATH_PREFIX': '/api/v1/',
    'SWAGGER_UI_SETTINGS': {
        'deepLinking': True,
        'persistAuthorization': True,
        'displayOperationId': True,
        'filter': True,
        'tryItOutEnabled': True,
    },
    'REDOC_UI_SETTINGS': {
        'hideDownloadButton': False,
        'expandResponses': 'all',
        'pathInMiddlePanel': True,
    },
    'PREPROCESSING_HOOKS': [
        'apps.core.schema_hooks.custom_preprocessing_hook',
```

```
    ],
    'POSTPROCESSING_HOOKS': [
        'apps.core.schema_hooks.custom_postprocessing_hook',
    ],
}
```

URL Configuration

The API documentation is accessible through multiple endpoints:

- `/api/schema/` - OpenAPI schema endpoint
- `/api/docs/` - Swagger UI interface
- `/api/redoc/` - ReDoc interface

API Documentation Structure

Authentication Endpoints

The authentication system provides comprehensive user management capabilities with detailed documentation for each endpoint.

User Registration

- **Endpoint**: `POST /api/v1/auth/register/`
- **Description**: Create a new user account with email verification
- **Request Body**: User registration data including username, email, password
- **Response**: User object with verification status
- **Error Codes**: 400 (validation errors), 409 (email already exists)

User Login

- **Endpoint**: `POST /api/v1/auth/login/`
- **Description**: Authenticate user and return JWT tokens with security features
- **Request Body**: Email and password credentials
- **Response**: Access token, refresh token, and user information
- **Security Features**: Account lockout after failed attempts, IP tracking
- **Error Codes**: 401 (invalid credentials), 423 (account locked)

Token Refresh

- **Endpoint**: `POST /api/v1/auth/token/refresh/`
- **Description**: Refresh access token using refresh token
- **Request Body**: Refresh token

- **Response**: New access token
- **Error Codes**: 401 (invalid refresh token)

Email Verification

- **Endpoint**: `POST /api/v1/auth/verify-email/`
- **Description**: Verify user email address with verification code
- **Request Body**: 6-digit verification code
- **Response**: Verification success confirmation
- **Error Codes**: 400 (invalid or expired code)

Password Management

- **Reset Request**: `POST /api/v1/auth/reset-password/`
- **Reset Confirm**: `POST /api/v1/auth/reset-password/confirm/`
- **Change Password**: `POST /api/v1/auth/change-password/`

Content Management Endpoints

The content management system provides dynamic control over landing page content, testimonials, features, and localization.

Landing Page Data

- **Endpoint**: `GET /api/v1/content/landing-page/`
- **Description**: Retrieve complete landing page content for specified language
- **Parameters**:
- `language` (optional): Language code (default: 'en')
- **Response**: Organized sections including hero, features, testimonials, CTA
- **Caching**: Content is cached for performance optimization

Testimonials

- **Endpoint**: `GET /api/v1/content/testimonials/`
- **Description**: Retrieve user testimonials with filtering options
- **Parameters**:
- `language` (optional): Language code
- `featured` (optional): Show only featured testimonials
- **Response**: Array of testimonial objects with user information and ratings

Features

- **Endpoint**: `GET /api/v1/content/features/`
- **Description**: Retrieve product features by language

- **Parameters**: `language` (optional)
- **Response**: Array of feature objects with descriptions and icons

Localization

- **Endpoint**: `GET /api/v1/content/localization/`
- **Description**: Retrieve localization texts organized by category
- **Parameters**:
- `language` (required): Language code
- `category` (optional): Filter by text category
- **Response**: Nested object with categories and translation keys

Paper Management Endpoints

The paper management system provides comprehensive functionality for academic paper generation, template management, and format handling.

Paper Formats

- **Endpoint**: `GET /api/v1/papers/formats/`
- **Description**: Retrieve all available academic paper formats
- **Response**: Array of format objects with structure definitions and style guidelines
- **Formats Included**: APA, MLA, Chicago, IEEE, and custom formats

Paper Templates

- **List**: `GET /api/v1/papers/templates/`
- **Detail**: `GET /api/v1/papers/templates/{id}/`
- **Search**: `GET /api/v1/papers/templates/search/`
- **Featured**: `GET /api/v1/papers/templates/featured/`
- **Recommended**: `GET /api/v1/papers/templates/recommended/`

Template Search Parameters

- `paper_type` : Filter by paper type (research, essay, thesis, etc.)
- `language` : Language code for templates
- `format_id` : Filter by specific format
- `is_premium` : Filter premium templates
- `q` : Search query for template names and descriptions

Paper Generation

- **Endpoint**: `POST /api/v1/papers/generate/`

- **Description**: Generate academic paper using AI with specified template
- **Request Body**: `json { "template_id": 1, "title": "Optional paper title", "user_inputs": { "topic": "Research topic", "length": 1500, "academic_level": "Undergraduate", "subject_area": "Psychology" } }`
- **Response**: Generated paper object with content and metadata
- **Credit System**: Deducts credits based on template requirements

Streaming Generation

- **Endpoint**: `POST /api/v1/papers/generate/stream/`
- **Description**: Generate paper with real-time streaming response
- **Response Type**: Server-Sent Events (SSE)
- **Event Types**: status, content, completed, error

Paper Validation

- **Endpoint**: `POST /api/v1/papers/validate/`
- **Description**: Validate generated paper content against requirements
- **Validation Checks**:
- Structure validation (required sections)
- Word count validation
- Citation validation
- Readability metrics

Paper Export

- **Endpoint**: `POST /api/v1/papers/export/`
- **Description**: Export paper to different formats
- **Supported Formats**: PDF, DOCX, LaTeX
- **Request Body**: Paper ID and desired format
- **Response**: Download URL for exported file

Paper History

- **Endpoint**: `GET /api/v1/papers/history/`
- **Description**: Retrieve user's generated paper history
- **Response**: Paginated list of user's papers with metadata

Billing and Payment Endpoints

The billing system provides comprehensive payment processing, credit management, and subscription handling.

Credit Packages

- **Endpoint**: `GET /api/v1/billing/packages/`
- **Description**: Retrieve available credit packages for purchase
- **Response**: Array of package objects with pricing and features
- **Package Information**: Credits, price, features, popularity indicators

Package Purchase

- **Endpoint**: `POST /api/v1/billing/purchase/`
- **Description**: Purchase credit package using payment method
- **Request Body**: `json { "package_id": 1, "payment_method_id": 2 }`
- **Response**: Transaction confirmation with updated credit balance

Transaction History

- **Endpoint**: `GET /api/v1/billing/transactions/`
- **Description**: Retrieve user's credit transaction history
- **Response**: Paginated transaction list with details

Payment Methods

- **List**: `GET /api/v1/billing/payment-methods/`
- **Create**: `POST /api/v1/billing/payment-methods/`
- **Update**: `PUT /api/v1/billing/payment-methods/{id}/`
- **Delete**: `DELETE /api/v1/billing/payment-methods/{id}/`

Billing Information

- **Endpoint**: `GET /api/v1/billing/info/`
- **Description**: Comprehensive billing information for user
- **Response**: Current credits, spending history, active subscriptions, payment methods

Pricing Information

- **Endpoint**: `GET /api/v1/billing/pricing/`
- **Description**: Public pricing information for pricing page
- **Response**: Package details and subscription plans

User Management Endpoints

User Profile

- **Get**: `GET /api/v1/auth/profile/`
- **Update**: `PATCH /api/v1/auth/profile/`
- **Description**: Manage user profile information and preferences

User Statistics

- **Endpoint**: `GET /api/v1/auth/stats/`
- **Description**: Retrieve user usage statistics
- **Response**: Papers generated, credits used, account metrics

User Preferences

- **Get**: `GET /api/v1/auth/preferences/`
- **Update**: `POST /api/v1/auth/preferences/update/`
- **Preferences**: Language, email notifications, marketing emails

Field Documentation

User Model Fields

Field	Type	Description	Validation
email	EmailField	User's email address (unique)	Valid email format, unique
username	CharField	User's username	3-150 characters, unique
credits	PositiveIntegerField	Available credits for paper generation	Non-negative integer
is_verified	BooleanField	Email verification status	Boolean
language	CharField	User's preferred language	Language code (en, zh-CN, etc.)

Field	Type	Description	Validation
total_papers_generated	PositiveIntegerField	Total papers created by user	Read-only
total_credits_used	PositiveIntegerField	Total credits consumed	Read-only

Paper Template Fields

Field	Type	Description	Validation
name	CharField	Template name	1-100 characters
paper_type	CharField	Type of paper	Choice field (research, essay, thesis, etc.)
format	ForeignKey	Associated paper format	Valid PaperFormat ID
language	CharField	Template language	Language code
required_fields	JSONField	Required input fields from user	Valid JSON array
estimated_credits	PositiveIntegerField	Credits required for generation	1-100
is_premium	BooleanField	Premium template flag	Boolean

Generated Paper Fields

Field	Type	Description	Validation
title	CharField	Paper title	1-200 characters
content	TextField	Generated paper content	Required
word_count	PositiveIntegerField	Number of words in paper	Auto-calculated

Field	Type	Description	Validation
status	CharField	Generation status	Choice field (pending, generating, completed, failed)
credits_used	PositiveIntegerField	Credits consumed for generation	Non-negative
generation_time	DurationField	Time taken to generate	Auto-calculated

Package Fields

Field	Type	Description	Validation
name	CharField	Package name	1-100 characters
credits	PositiveIntegerField	Number of credits in package	Positive integer
price	DecimalField	Package price	Positive decimal, 2 decimal places
currency	CharField	Price currency	3-character currency code
features	JSONField	Package features list	Valid JSON array
is_popular	BooleanField	Popular package indicator	Boolean

Error Handling Documentation

Standard Error Responses

All API endpoints follow consistent error response formatting:

```
{
 "error": "Error message description",
 "details": {
   "field_name": ["Field-specific error messages"]
 },
```

```
  "code": "ERROR_CODE"
}
```

Common Error Codes

Code	Status	Description
VALIDATION_ERROR	400	Request data validation failed
AUTHENTICATION_REQUIRED	401	Authentication credentials required
PERMISSION_DENIED	403	Insufficient permissions
NOT_FOUND	404	Requested resource not found
INSUFFICIENT_CREDITS	402	Not enough credits for operation
ACCOUNT_LOCKED	423	Account temporarily locked
RATE_LIMITED	429	Too many requests
INTERNAL_ERROR	500	Server internal error

Field Validation Errors

Each field type has specific validation rules and error messages:

Email Fields

- Invalid format: "Enter a valid email address"
- Already exists: "User with this email already exists"

Password Fields

- Too short: "Password must be at least 8 characters"
- Too common: "This password is too common"
- Similar to user info: "Password too similar to personal information"

Credit Fields

- Insufficient credits: "Not enough credits for this operation"
- Invalid amount: "Credit amount must be positive"

Authentication Documentation

JWT Token Authentication

The API uses JSON Web Tokens (JWT) for authentication with the following characteristics:

Token Structure

- **Access Token**: Short-lived (60 minutes) for API requests
- **Refresh Token**: Long-lived (7 days) for token renewal
- **Token Rotation**: Refresh tokens are rotated on use
- **Blacklisting**: Used tokens are blacklisted for security

Authentication Header

```
Authorization: Bearer <access_token>
```

Token Refresh Process

1. Send refresh token to `/api/v1/auth/token/refresh/`
2. Receive new access token and refresh token
3. Old refresh token is automatically blacklisted

Security Features

Account Protection

- **Failed Login Tracking**: Monitors failed login attempts
- **Account Lockout**: Temporary lockout after 5 failed attempts
- **IP Tracking**: Records login IP addresses
- **Session Management**: Secure session handling

Password Security

- **Strength Requirements**: Minimum 8 characters with complexity rules
- **Hash Storage**: Passwords stored using Django's PBKDF2 algorithm
- **Reset Security**: Secure password reset with time-limited codes

Rate Limiting Documentation

Rate Limiting Rules

Endpoint Category	Rate Limit	Window
Authentication	5 requests	1 minute
Paper Generation	10 requests	1 hour
Content Retrieval	100 requests	1 hour
Billing Operations	20 requests	1 hour
General API	1000 requests	1 hour

Rate Limit Headers

Response headers include rate limiting information:

```
 X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1640995200
```

Pagination Documentation

Pagination Parameters

Parameter	Type	Description	Default
page	Integer	Page number	1
page_size	Integer	Items per page	20
limit	Integer	Maximum items (alternative)	20
offset	Integer	Items to skip (alternative)	0

Pagination Response Format


```
{
 "count": 150,
 "next": "http://api.example.com/endpoint/?page=3",
```

```
  "previous": "http://api.example.com/endpoint/?page=1",
  "results": [...]
}
```

API Versioning

Version Strategy

- **Current Version**: v1
- **URL Versioning**: `/api/v1/`
- **Backward Compatibility**: Maintained for major versions
- **Deprecation Policy**: 6-month notice for breaking changes

Version Headers

```
API-Version: 1.0
Accept-Version: 1.0
```

Testing Documentation

API Testing Guidelines

Test Categories

1. **Unit Tests**: Individual endpoint functionality
2. **Integration Tests**: Cross-service interactions
3. **Performance Tests**: Load and stress testing
4. **Security Tests**: Authentication and authorization

Test Data

- **Test Users**: Predefined test accounts
- **Sample Templates**: Test paper templates
- **Mock Payments**: Simulated payment processing

Testing Tools

- **Django Test Framework**: Built-in testing capabilities
- **Factory Boy**: Test data generation
- **pytest**: Advanced testing features
- **Coverage.py**: Code coverage analysis

Performance Optimization

Caching Strategy

- **Redis Cache**: Session and temporary data
- **Database Query Optimization**: Select_related and prefetch_related
- **API Response Caching**: Cached responses for static content
- **CDN Integration**: Static asset delivery

Database Optimization

- **Indexing**: Optimized database indexes
- **Query Optimization**: Efficient database queries
- **Connection Pooling**: Database connection management
- **Read Replicas**: Scaled read operations

Monitoring and Logging

API Monitoring

- **Request Logging**: Comprehensive request/response logging
- **Performance Metrics**: Response time and throughput monitoring
- **Error Tracking**: Automated error detection and alerting
- **Usage Analytics**: API usage patterns and statistics

Health Checks

- **System Health**: `/health/` endpoint for system status
- **Database Health**: Database connectivity checks
- **External Service Health**: Third-party service monitoring
- **Performance Metrics**: Real-time performance indicators

This comprehensive API documentation provides complete coverage of all endpoints, field specifications, error handling, and integration guidelines for the Academic Paper Generator application. The documentation is designed to support both frontend developers and third-party integrators with detailed examples and clear specifications for all API interactions.