Banner Management Dashboard API Documentation

Overview

This document provides comprehensive documentation for the Banner Management dashboard API endpoints, including CRUD operations, request/response formats, and all possible status codes.

Base URL

```
/api/dashboard/banners
```

Authentication

Required: Dashboard endpoints require authentication middleware.

- Middleware: `handleLang` (for language handling)
- Additional authentication may be required based on your setup

Headers

```
Content-Type: application/json
Accept: application/json
lang: en|ar (optional, handled by handleLang middleware)
Authorization: Bearer {token} (if authentication is enabled)
```

Endpoints

1. Get Paginated Banners

Endpoint: `GET /api/dashboard/banners`

Description: Retrieve paginated list of banners for dashboard management.

Request:

```
GET /api/dashboard/banners
Accept: application/json
lang: en
```

Query Parameters:

- `page` (optional, integer): Page number (default: 1)
- `position` (optional, string): Filter by position (top, middle, bottom)
- `page_type` (optional, string): Filter by page type (home, about, products, etc.)
- `status` (optional, boolean(0,1)): Filter by status (active, inactive).

Success Response (200 OK)


```
{
    "code": 200,
    "message": "Banners fetched successfully",
    "errors": [],
    "data": {
        "banners": [
            {
                "id": 1,
                "image": "/storage/banners/banner-1.jpg",
                "url": "/shop",
                "position": "top",
                "page": "home",
                "is_active": true,
                "created_at": "2024-01-15T10:30:00.000000Z",
                "updated_at": "2024-01-15T12:45:00.000000Z"
            }
        ],
        "pagination": {
            "current_page": 1,
            "last_page": 3,
            "per_page": 15,
            "total": 35,
            "from": 1,
            "to": 15
        }
    }
}
```


Error Responses

500 Internal Server Error:


```
{
    "code": 500,
    "message": "Failed to fetch banners",
    "errors": [],
    "data": null
}
```

2. Get All Banners (No Pagination)

Endpoint: `GET /api/dashboard/banners/all`

Description: Retrieve all banners without pagination (useful for dropdowns, etc.).

Request:

```
GET /api/dashboard/banners/all
Accept: application/json
lang: en
```

Success Response (200 OK)


```
{
    "code": 200,
    "message": "Banners fetched successfully",
    "errors": [],
    "data": [
        {
            "id": 1,
            "image": "/storage/banners/banner-1.jpg",
            "url": "/shop",
            "position": "top",
            "page": "home",
            "is_active": true,
            "created_at": "2024-01-15T10:30:00.000000Z",
            "updated_at": "2024-01-15T12:45:00.000000Z"
        }
    ]
}
```


3. Get Banner by ID

Endpoint: `GET /api/dashboard/banners/{id}`

Description: Retrieve a specific banner by its ID.

Request:

```
GET /api/dashboard/banners/1
Accept: application/json
lang: en
```

Parameters:

- `id` (required, integer): The banner ID

Success Response (200 OK)


```
{
    "code": 200,
    "message": "Banner fetched successfully",
    "errors": [],
    "data": {
        "id": 1,
        "image": "/storage/banners/banner-1.jpg",
        "url": "/shop",
        "position": "top",
        "page": "home",
        "is_active": true,
        "created_at": "2024-01-15T10:30:00.000000Z",
        "updated_at": "2024-01-15T12:45:00.000000Z"
    }
}
```


Error Responses

404 Not Found:


```
{
    "code": 404,
    "message": "Banner not found",
    "errors": [],
    "data": null
}
```

4. Create Banner

Endpoint: `POST /api/dashboard/banners`

Description: Create a new banner.

Request:

```
POST /api/dashboard/banners
Content-Type: application/json
Accept: application/json
lang: en

{
    "image": "/storage/banners/new-banner.jpg",
    "url": "/sale",
    "position": "top",
    "page": "home",
    "is_active": true
}
```

Required Fields:

- `image` (string, max: 255): Path to banner image
- `position` (string, enum): Banner position (top, middle, bottom)
- `page` (string, enum): Page type (home, about, products, contact, blog)

Optional Fields:

- `url` (string, max: 255): Banner URL
- `is_active` (boolean): Banner active status (default: true)

Success Response (201 Created)


```
{
    "code": 201,
    "message": "Banner created successfully",
    "errors": [],
    "data": {
        "id": 2,
        "image": "/storage/banners/new-banner.jpg",
        "url": "/sale",
        "position": "top",
        "page": "home",
        "is_active": true,
        "created_at": "2024-01-15T14:30:00.000000Z",
        "updated_at": "2024-01-15T14:30:00.000000Z"
    }
}
```


Error Responses

422 Unprocessable Entity (Validation Error):


```
{
    "code": 422,
    "message": "Missing data",
    "errors": [
        {
            "key": "image",
            "value": "The image field is required."
        },
        {
            "key": "position",
            "value": "The position must be one of: top, middle, bottom."
        }
    ],
    "data": null
}
```


500 Internal Server Error:

```
{
    "code": 500,
    "message": "Failed to create banner",
    "errors": [],
    "data": null
}
```

5. Update Banner

Endpoint: `PUT /api/dashboard/banners/{id}`

Description: Update an existing banner.

Request:

```
PUT /api/dashboard/banners/1
Content-Type: application/json
Accept: application/json
lang: en

{
    "position": "middle",
    "is_active": false
}
```

Optional Fields: All fields are optional for updates

- `image` (string, max: 255): Path to banner image
- `url` (string, max: 255): Banner URL
- `position` (string, enum): Banner position (top, middle, bottom)
- `page` (string, enum): Page type (home, about, products, contact, blog)
- `is_active` (boolean): Banner active status

Success Response (200 OK)


```
{
    "code": 200,
    "message": "Banner updated successfully",
    "errors": [],
    "data": {
        "id": 1,
        "image": "/storage/banners/banner-1.jpg",
        "url": "/shop",
        "position": "middle",
        "page": "home",
        "is_active": false,
    }
}
```


Error Responses

404 Not Found:


```
{
    "code": 404,
    "message": "Banner not found",
    "errors": [],
    "data": null
}
```


422 Unprocessable Entity:


```
{
    "code": 422,
    "message": "Invalid data",
    "errors": [
        {
            "key": "position",
            "value": "The position must be one of: top, middle, bottom."
        }
    ],
    "data": null
}
```

6. Delete Banner

Endpoint: `DELETE /api/dashboard/banners/{id}`

Description: Delete a banner permanently.

Request:

```
DELETE /api/dashboard/banners/1
Accept: application/json
lang: en
```

Parameters:

- `id` (required, integer): The banner ID to delete

Success Response (200 OK)

```
{
    "code": 200,
    "message": "Banner deleted successfully",
    "errors": [],
    "data": null
}
```

Error Responses

404 Not Found:

```
{
    "code": 404,
    "message": "Banner not found",
    "errors": [],
    "data": null
}
```

500 Internal Server Error:

```
{
    "code": 500,
    "message": "Failed to delete banner",
    "errors": [],
    "data": null
}
```


HTTP Status Codes Reference

Status Code	Description	When it occurs
200	OK	Successful GET, PUT, PATCH, DELETE requests
201	Created	Successful POST request (resource created)
400	Bad Request	Invalid request format or parameters
401	Unauthorized	Authentication required or failed
403	Forbidden	Access denied to resource
404	Not Found	Banner not found or invalid endpoint
405	Method Not Allowed	HTTP method not supported for endpoint
422	Unprocessable Entity	Validation errors in request data
500	Internal Server Error	Server-side error occurred
503	Service Unavailable	Service temporarily unavailable

Route List Summary

Method	Endpoint	Description
GET	`/api/dashboard/banners`	Get paginated banners
GET	`/api/dashboard/banners/all`	Get all banners (no pagination)

Method	Endpoint	Description
GET	`/api/dashboard/banners/{id}`	Get specific banner
POST	`/api/dashboard/banners`	Create new banner
POST	`/api/dashboard/banners/{id}`	Update banner
DELETE	`/api/dashboard/banners/{id}`	Delete banner

Data Structure Reference

Banner Object Structure

```
{
    "id": integer,           // Unique banner identifier
    "image": string,         // URL/path to banner image
    "url": string,           // Optional URL for banner link
    "position": string,      // Banner position (top, middle, bottom)
    "page": string,         // Page where banner appears (home, about, products, etc.)
    "is_active": boolean,   // Whether banner is active/enabled
}
```

Pagination Object Structure

```
{
    "current_page": integer,          // Current page number
    "last_page": integer,             // Total number of pages
    "per_page": integer,              // Items per page
    "total": integer,                 // Total number of items
    "from": integer,                  // First item number on current page
    "to": integer                     // Last item number on current page
}
```

Validation Rules

Create Banner (POST)

- `image` : required, string, max:255
- `url` : nullable, string, max:255
- `position` : required, string, in:top,middle,bottom
- `page` : required, string, in:home,about,products,contact,blog
- `is_active` : boolean (default: true)

Update Banner (PUT)

- `image` : sometimes, string, max:255
- `url` : sometimes, string, max:255
- `position` : sometimes, string, in:top,middle,bottom
- `page` : sometimes, string, in:home,about,products,contact,blog
- `is_active` : sometimes, boolean

Authentication Notes

If authentication middleware is implemented:

- Include `Authorization: Bearer {token}` header
- Handle 401 responses by redirecting to login
- Refresh tokens when necessary
- Store tokens securely

Enums Reference

Position Values

- `top`
- `middle`
- `bottom`

Page Values

- `home` (default)
- `offers`