API Documentation

GET /api/ping

Simple ping endpoint to check if the server is responsive.

Headers:

N/A

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "status": "pong",
  "message": "Server is running"
}
```

Error (500):

JSON

```
{
  "status": "error",
  "message": "Server error"
}
```

POST /api/users/register

Create a new user. Requires admin privileges.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
first_name	string	Required	User's first name.
last_name	string	Required	User's last name.
email	string	Required	User's email address (must be unique).
contact_number	string	Required	User's contact phone number.
username	string	Required	Username (must be unique, min 4 chars).
password	string	Required	User's password (min 8 chars).
role_id	integer	Required	ID of the role to assign.
environment_id	integer	Required	ID of the environment to assign.

Example:

JSON

```
{
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.doe@example.com",
  "contact_number": "1234567890",
  "username": "johndoe",
  "password": "securePassword123",
  "role_id": 2,
  "environment_id": 1
}
```

Responses:

Success (201):

JSON

```
{
  "message": "User created successfully",
  "user": {
    "id": 1,
    "username": "johndoe",
    "email": "john.doe@example.com",
    "first_name": "John",
    "last_name": "Doe",
    "contact_number": "1234567890",
    "role": {
      "id": 2,
      "name": "Technician",
      "description": "Technical staff",
      "is_super_user": false
    },
    "environment": {
      "id": 1,
      "name": "Default",
      "description": "Default environment"
    },
    "is_deleted": false,
    "created_at": "2025-04-27T20:05:29Z",
    "updated_at": "2025-04-27T20:05:29Z"
  }
}
```

Error (400):

JSON

```
{
  "error": "Missing required fields | Username already exists | Email already exists | Password must be at least 8 characters long | ..."
}
```

POST /api/users/login

Authenticate a user and receive a JWT.

Headers:

Name	Purpose	Status	Example
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
username	string	Required	Registered username.
password	string	Required	User's password.

Example:

JSON

```
{
  "username": "johndoe",
  "password": "securePassword123"
}
```

Responses:

Success (200):

JSON

```
{
  "access_token": "<JWT_TOKEN>"
}
```

Error (401):

JSON

```
{
  "error": "Invalid credentials"
}
```

POST /api/users/logout

Logs out the user. Invalidates the token if possible (depends on implementation, often relies on client-side token removal).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token (optional, aids logging)	Optional	Bearer <JWT_TOKEN>

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
```

```
 "message": "Successfully logged out",
 "status": "success"
}
```

Error: N/A (Typically designed to succeed even with invalid tokens)

GET /api/users

Get a list of users. Filters based on user role (Admin sees all, others see users in their environment).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include soft-deleted users (Admin only). Default: false.

Request Body:

N/A

Responses:

Success (200):

JSON

```
[
 {
  "id": 1,
  "username": "johndoe",
  "email": "john.doe@example.com",
  "first_name": "John",
  "last_name": "Doe",
  "contact_number": "1234567890",
  "role": {
   "id": 2,
   "name": "Technician",
   "description": "Technical staff",
   "is_super_user": false
  },
  "environment": {
   "id": 1,
   "name": "Default",
   "description": "Default environment"
  },
  "is_deleted": false,
  "created_at": "2025-04-27T20:05:29Z",
  "updated_at": "2025-04-27T20:05:29Z"
 },
 { ... }
]
```

Error (500):

JSON

```
{
  "error": "Internal server error | Database error: <details>"
}
```

GET /api/users/batch

Get a paginated list of users in a compact format.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
page	integer	Optional	Page number. Default: 1.
per_page	integer	Optional	Items per page. Default: 50.
include_deleted	boolean	Optional	Include soft-deleted users (Admin only). Default: false.
role_id	integer	Optional	Filter by role ID.
environment_id	integer	Optional	Filter by environment ID (Admin only, otherwise uses user's environment).

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "metadata": {
    "total_items": 5,
    "total_pages": 1,
    "current_page": 1,
    "per_page": 50
  },
  "items": [
    {
      "id": 1,
      "username": "johndoe",
      "first_name": "John",
      "last_name": "Doe",
      "full_name": "John Doe",
      "email": "john.doe@example.com",
      "contact_number": "1234567890",
      "role": {
        "id": 2,
        "name": "Technician",
        "description": "Technical staff",
```

```
      "is_super_user": false
    },
    "environment": {
     "id": 1,
     "name": "Default",
     "description": "Default environment"
    }
   },
   { ... }
 ]
}
```


Error (500):


```
JSON
{
 "error": "Internal server error",
 "details": "<error_details>"
}
```

GET /api/users/compact-list

Get a list of all users with essential details (no permissions).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include soft-deleted users (Admin only). Default: false.

Request Body:

N/A

Responses:

Success (200):


```
JSON
[
 {
  "id": 1,
  "username": "johndoe",
  "first_name": "John",
  "last_name": "Doe",
  "full_name": "John Doe",
  "email": "john.doe@example.com",
  "contact_number": "1234567890",
  "role": {
   "id": 2,
   "name": "Technician",
   "description": "Technical staff",
```

```
    "is_super_user": false
   },
   "environment": {
    "id": 1,
    "name": "Default",
    "description": "Default environment"
   }
  },
  { ... }
]
```

Error (500):

```
JSON
{
  "error": "Internal server error | Database error: <details>"
}
```

GET /api/users/byRole/{role_id}

Get users belonging to a specific role.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
role_id	integer	Required	ID of the role.

Request Body:

N/A

Responses:

Success (200):

```
JSON
[
  {
   "id": 1,
   "username": "johndoe",
   "email": "john.doe@example.com",
   "first_name": "John",
   "last_name": "Doe",
   "contact_number": "1234567890",
   "role": { ... },
   "environment": { ... }
  },
  { ... }
]
```

Error (500):

```
JSON
```

```
{
 "error": "Internal server error"
}
```

GET /api/users/byEnvironment/{environment_id}

Get users belonging to a specific environment.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
environment_id	integer	Required	ID of the environment.

Request Body:

N/A

Responses:

Success (200):

JSON

```
[
 {
  "id": 1,
  "username": "johndoe",
  "email": "john.doe@example.com",
   ...
 },
 { ... }
]
```

Error (403):

JSON

```
{
 "error": "Unauthorized access to environment"
}
```

GET /api/users/search

Search for users based on criteria (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
id	integer	Optional	Search by user ID.
username	string	Optional	Search by username (partial match).

role_id	integer	Optional	Search by role ID.
environment_id	integer	Optional	Search by environment ID.

Request Body:

N/A

Responses:

Success (200):

JSON

```
[
 {
   "id": 1,
   "username": "johndoe",
    ... (full user details)
 },
 { ... }
]
```

Error (403):

JSON

```
{
 "error": "Unauthorized"
}
```

GET /api/users/{user_id}

Get details for a specific user.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
user_id	integer	Required	ID of the user.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
 "id": 1,
 "username": "johndoe",
 "email": "john.doe@example.com",
 "first_name": "John",
 "last_name": "Doe",
 "contact_number": "1234567890",
 "role": { ... },
 "environment": { ... },
```

```
 "permissions": [ "perm1", "perm2", ...],
 "is_deleted": false,
 "created_at": "...",
 "updated_at": "..."
}
```

Error (404):

```
{
 "error": "User not found"
}
```

PUT /api/users/{user_id}

Update details for a specific user.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
user_id	integer	Required	ID of the user to update.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
first_name	string	Optional	User's first name.
last_name	string	Optional	User's last name.
email	string	Optional	User's email address (must be unique).
contact_number	string	Optional	User's contact phone number.
password	string	Optional	New password (min 8 chars).
username	string	Optional	Username (Admin only, must be unique, min 4 chars).
role_id	integer	Optional	ID of the role to assign (Admin only).
environment_id	integer	Optional	ID of the environment to assign (Admin only).

Example:

```
{
 "first_name": "Jonathan",
 "email": "jonathan.doe@example.com",
 "password": "newSecurePassword456"
}
```

Responses:

Success (200):

```
{
  "message": "User updated successfully",
  "user": { ... (updated user details) }
}
```

Error (400):

JSON

```
{
  "error": "Username already exists | Email already exists | Invalid role_id | ..."
}
```

DELETE /api/users/{user_id}

Soft delete a user and associated data.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
user_id	integer	Required	ID of the user to delete.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "message": "User and all associated data deleted successfully",
  "deleted_items": {
    "user": 1,
    "forms_created": 0,
    "form_submissions": 5,
    "answers_submitted": 25,
    "attachments": 3
  }
}
```

Error (404):

JSON

```
{
  "error": "User not found"
}
```

GET /api/users/current

Get details of the currently authenticated user.

Headers:

Name	Purpose	Status	Example

Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "id": 1,
  "username": "johndoe",
  "email": "john.doe@example.com",
   ... (full user details including permissions)
}
```

Error (404):

JSON

```
{
  "error": "User not found"
}
```

POST /api/roles

Create a new role (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
name	string	Required	Name of the role (must be unique).
description	string	Optional	Description of the role.
is_super_user	boolean	Optional	Whether this is a super user role (default: false).

Example:

JSON

```
{
  "name": "Editor",
  "description": "Can edit content",
  "is_super_user": false
}
```

Responses:

Success (201):

JSON

```
{
  "message": "Role created successfully",
  "role": {
    "id": 3,
    "name": "Editor",
    "description": "Can edit content",
    "is_super_user": false,
    "is_deleted": false,
    "created_at": "...",
    "updated_at": "..."
  }
}
```

Error (400):

JSON

```
{
  "error": "Name is required | Role name already exists"
}
```

GET /api/roles

Get a list of roles. Non-admins cannot see super user roles.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
[
  {
    "id": 1,
    "name": "Admin",
    "description": "Full system administrator",
    "is_super_user": true,
    ...
  },
  {
    "id": 2,
    "name": "Technician",
    "description": "Technical staff",
    "is_super_user": false,
    ...
  }
```

]

Error (500):

JSON

```
{
  "error": "Internal server error"
}
```

GET /api/roles/batch

Get a paginated list of roles.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
page	integer	Optional	Page number. Default: 1.
per_page	integer	Optional	Items per page. Default: 50.
include_deleted	boolean	Optional	Include soft-deleted roles (Admin only). Default: false.
is_super_user	boolean	Optional	Filter by super user status.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "metadata": { ... },
  "items": [
    {
      "id": 1,
      "name": "Admin",
      ...
    },
    { ... }
  ]
}
```

Error (500):

JSON

```
{
  "error": "Internal server error",
  "details": "<error_details>"
}
```

GET /api/roles/{role_id}

Get details for a specific role.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
role_id	integer	Required	ID of the role.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "id": 2,
  "name": "Technician",
  "description": "Technical staff",
  "is_super_user": false,
   ...
}
```

Error (404):

JSON

```
{
  "error": "Role not found"
}
```

PUT /api/roles/{role_id}

Update a role (Admin only). Cannot modify the main Admin role.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
role_id	integer	Required	ID of the role to update.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
name	string	Optional	New name for the role (must be unique).
description	string	Optional	New description for the role.
is_super_user	boolean	Optional	Set super user status.

Example:

JSON

```
{
  "description": "Staff performing technical duties"
}
```

Responses:

Success (200):

JSON

```
{
  "message": "Role updated successfully",
  "role": { ... (updated role details) }
}
```

Error (403):

JSON

```
{
  "error": "Cannot modify the main administrator role"
}
```

DELETE /api/roles/{role_id}

Soft delete a role (Admin only). Cannot delete the main Admin role or roles with active users.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
role_id	integer	Required	ID of the role to delete.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "message": "Role and associated permissions deleted successfully"
}
```

Error (400):

JSON

```
{
  "error": "Cannot delete role with active users",
  "role": { ... },
  "active_users": {
    "count": 2,
    "users": [ { ... }, { ... } ]
  },
  "suggestion": "Please reassign or deactivate these users before deleting this role"
}
```

DELETE /api/roles/{role_id}/permissions/{permission_id}

Remove a specific permission from a role (Admin only). Cannot modify the main Admin role.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
role_id	integer	Required	ID of the role.
permission_id	integer	Required	ID of the permission to remove.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "message": "Permission removed from role successfully"
}
```

Error (400):

JSON

```
{
  "error": "Failed to remove permission from role"
}
```

POST /api/forms

Create a new form.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
title	string	Required	Title of the form.
description	string	Optional	Description of the form.
is_public	boolean	Optional	Whether the form is public (default: false).
user_id	integer	Optional	ID of the user to assign as creator (Admin only). Defaults to current user.

Example:

JSON

```
{
  "title": "Daily Safety Check",
  "description": "Checklist for daily equipment safety",
  "is_public": false
}
```

Responses:

Success (201):

JSON

```
{
  "message": "Form created successfully",
  "form": {
    "id": 1,
    "title": "Daily Safety Check",
    "description": "Checklist for daily equipment safety",
    "is_public": false,
    "created_by": { ... },
    "questions": [],
    "is_deleted": false,
    "created_at": "...",
    "updated_at": "..."
  },
  "form_creator": "current_username"
}
```

Error (400):

JSON

```
{
  "error": "Missing required fields | Form title already exists | ..."
}
```

GET /api/forms

Get a list of forms accessible to the user.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
[
  {
    "id": 1,
```

```
  "title": "Daily Safety Check",
  "description": "...",
  "is_public": false,
  "created_by": { ... },
  "questions": [ { ... }, ... ],
   ...
 },
 { ... }
]
```

Error (500):

```
{
 "error": "Internal server error"
}
```

GET /api/forms/batch

Get a paginated list of forms with filtering options.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
page	integer	Optional	Page number. Default: 1.
per_page	integer	Optional	Items per page. Default: 50.
include_deleted	boolean	Optional	Include soft-deleted forms (Admin only). Default: false.
is_public	boolean	Optional	Filter by public status.
user_id	integer	Optional	Filter by creator user ID.
environment_id	integer	Optional	Filter by environment ID (Admin only).
only_editable	boolean	Optional	Return only forms editable by the user. Default: false.

Request Body:

N/A

Responses:

Success (200):

```
{
 "metadata": { ... },
 "items": [
  {
   "id": 1,
   "title": "Daily Safety Check",
    ... (compact form details)
  },
```

{ ... }
]
}
Error (500):
JSON
{
"error": "Internal server error",
"details": "<error_details>"
}

GET /api/forms/{form_id}
Get details for a specific form.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
Path Parameters:

Name	Type	Status	Description
form_id	integer	Required	ID of the form.

Request Body:
N/A
Responses:
Success (200):
JSON
{
"id": 1,
"title": "Daily Safety Check",
"description": "...",
"is_public": false,
"created_by": { ... },
"questions": [
{
"id": 1,
"form_question_id": 101,
"text": "Is equipment powered off?",
"type": "checkbox",
"order_number": 1,
"remarks": null,
"possible_answers": [{ "id": 1, "form_answer_id": 201, "value": "Yes" }, { "id": 2, "form_answer_id": 202, "value": "No" }]
},
{ ... }
],
"is_deleted": false,
"created_at": "...",

```
 "updated_at": "..."
}
```

Error (404):

```
JSON
{
 "error": "Form not found"
}
```

POST /api/forms/{form_id}/questions

Add multiple questions to an existing form.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
form_id	integer	Required	ID of the form to add questions to.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
questions	array	Required	List of question objects to add.

Example:

```
JSON
{
 "questions": [
  { "question_id": 15, "order_number": 3 },
  { "question_id": 22, "order_number": 4 }
 ]
}
```

Responses:

Success (200):

```
JSON
{
 "message": "Questions added successfully",
 "form": { ... (updated form details including new questions) }
}
```

Error (400):

```
JSON
{
 "error": "Questions are required | Invalid question ID | ..."
}
```

POST /api/form-submissions

Submit answers for a specific form.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json or multipart/form-data if signatures are included

URL Parameters:

N/A

Request Body:

Format: JSON (potentially multipart/form-data) **Fields:**

Name	Type	Status	Description
form_id	integer	Required	ID of the form being submitted.
submitted_at	string	Optional	Timestamp of submission in ISO format (YYYY-MM-DDTHH:MM:SS). Defaults to server time.
answers	array	Required	List of answers submitted.

Example:

JSON

```
{
 "form_id": 1,
 "answers": [
  {
   "question_id": 5,
   "answer_text": "Yes",
   "question_type_text": "checkbox"
  },
  {
   "question_id": 6,
   "answer_text": "Checked all safety guards.",
   "question_type_text": "text"
  },
  {
   "question_id": 7,
   "answer_text": "Signature data or reference",
   "question_type_text": "signature",
   "is_signature": true
  }
 ]
}
```

Responses:

Success (201):

JSON

```
{
 "message": "Form submitted successfully",
```

```
 "submission": {
  "id": 101,
  "form_id": 1,
  "submitted_by": "current_username",
  "submitted_at": "...",
  "answers_submitted": [ ... ],
  "attachments": [ ... ],
   ...
 }
}
```

Error (400):

```
JSON
{
 "error": "form_id is required | Invalid answer data | ..."
}
```

POST /api/answers-submitted

Create a single submitted answer record. Typically used internally by form submission.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
form_submission_id	integer	Required	ID of the parent form submission.
question_text	string	Required	Text of the question at the time of submission.
question_type_text	string	Required	Type of the question (e.g., 'text', 'table').
answer_text	string	Required	Answer provided by the user.
column	integer	Conditional	Column index (required if question_type_text is 'table').
row	integer	Conditional	Row index (required if question_type_text is 'table').
cell_content	string	Optional	Content for table cells.

Example:

```
JSON
{
 "form_submission_id": 101,
 "question_text": "Is equipment powered off?",
 "question_type_text": "checkbox",
 "answer_text": "Yes"
}
```

Responses:

Success (201):

JSON

```
{
  "message": "Answer submitted successfully",
  "answer_submitted": {
    "id": 501,
    "form_submission_id": 101,
    "question": "Is equipment powered off?",
    "question_type": "checkbox",
    "answer": "Yes",
    "column": null,
    "row": null,
    "cell_content": null,
     ...
  }
}
```

Error (400):

JSON

```
{
  "error": "Missing required fields | Column and row are required for table-type questions | ..."
}
```

POST /api/answers-submitted/bulk

Create multiple submitted answer records for a single form submission. Typically used internally.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
form_submission_id	integer	Required	ID of the parent form submission.
submissions	array	Required	List of submitted answers.

Example:

JSON

```
{
  "form_submission_id": 101,
  "submissions": [
    { "question_text": "Q1?", "question_type_text": "text", "answer_text": "Ans1" },
    { "question_text": "Q2?", "question_type_text": "table", "answer_text": "CellVal", "column": 0, "row": 1,
"cell_content": "Cell Value" }
  ]
```

}

Responses:

Success (201):

JSON

```
{
  "message": "Answers submitted successfully",
  "submissions": [ { ... }, { ... } ]
}
```

Error (400):

JSON

```
{
  "error": "Missing required fields: form_submission_id and submissions | Each submission must contain required fields | ..."
}
```

POST /api/attachments

Upload an attachment (file or signature) linked to a form submission.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	multipart/form-data

URL Parameters:

N/A

Request Body:

Format: Form Data **Fields:**

Name	Type	Status	Description
form_submission_id	integer	Required	ID of the form submission to link to.
file	file	Required	The file to upload.
is_signature	boolean	Optional	Set to 'true' if the file is a signature image. Default: false.
answer_submitted_id	integer	Conditional	ID of the AnswerSubmitted record this signature relates to (if is_signature=true).
signature_position	string	Optional	Position/identifier for the signature (e.g., 'Completed By').
signature_author	string	Optional	Name of the person who signed.

Example: N/A (Form data)

Responses:

Success (201):

JSON

```
{
  "message": "Attachment created successfully",
  "attachment": {
    "id": 301,
    "form_submission_id": 101,
    "filename": "signature_john_doe.png",
```

```
  "file_path": "uploads/submission_101/signature_john_doe_timestamp.png",
  "file_type": "image/png",
  "is_signature": true,
  "answer_submitted_id": 505,
  "signature_position": "Completed By",
  "signature_author": "John Doe",
   ...
 }
}
```


Error (400):


```
JSON
{
 "error": "form_submission_id is required | No file provided | Invalid file type | ..."
}
```

POST /api/attachments/bulk

Upload multiple attachments for a single form submission.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	multipart/form-data

URL Parameters:

N/A

Request Body:

Format: Form Data **Fields:**

Name	Type	Status	Description
form_submission_id	integer	Required	ID of the form submission.
file<N>	file	Required	File upload field (e.g., file1, file2).
is_signature<N>	boolean	Optional	Indicates if file<N> is a signature.
answer_submitted_id<N>	integer	Conditional	Answer ID for signature<N>.
signature_position<N>	string	Optional	Position for signature<N>.
signature_author<N>	string	Optional	Author for signature<N>.

Example: N/A (Form data with multiple file fields like file1, file2, etc.)

Responses:

Success (201):


```
JSON
{
 "message": "Attachments created successfully",
 "attachments": [ { ... }, { ... } ]
}
```


Error (400):


```
JSON
{
```

"error": "form_submission_id is required | No files provided | ..."
}

GET /api/attachments/{attachment_id}

Download a specific attachment file.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
attachment_id	integer	Required	ID of the attachment to download.

Request Body:

N/A

Responses:

Success (200): File content (e.g., image/png, application/pdf) **Error (404):**

JSON

{
"error": "File not found"
}

GET /api/export/form/{form_id}

Export a form structure (questions, possible answers) to PDF or DOCX.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
form_id	integer	Required	ID of the form to export.

Query Parameters:

Name	Type	Status	Description
format	string	Optional	Export format ('PDF' or 'DOCX'). Default: PDF.
page_size	string	Optional	Page size ('A4', 'LETTER', 'LEGAL'). Default: LETTER.
margin_top	float	Optional	Top margin in inches. Default: 1.0.
margin_bottom	float	Optional	Bottom margin in inches. Default: 1.0.
margin_left	float	Optional	Left margin in inches. Default: 1.0.
margin_right	float	Optional	Right margin in inches. Default: 1.0.
line_spacing	float	Optional	Line spacing multiplier. Default: 1.15.
font_size	integer	Optional	Base font size in points. Default: 12.
logo_path	string	Optional	Server path to a logo image (PNG/JPG).

signature_count	integer	Optional	Number of signature lines (1-5). Default: 2.
signature<N>_title	string	Optional	Title for signature line N (e.g., 'signature1_title').
signature<N>_name	string	Optional	Pre-filled name for signature line N.
signature<N>_date	boolean	Optional	Include date field for signature line N. Default: true.
signature_space_before	float	Optional	Space before signature section (points). Default: 20.
signature_space_between	float	Optional	Space between signatures (points). Default: 10.
signature_space_date	float	Optional	Space after date field (points). Default: 5.
signature_space_after	float	Optional	Space after signature section (points). Default: 20.

Request Body:
N/A
Responses:
Success (200): File content (application/pdf or application/vnd.openxmlformats-officedocument.wordprocessingml.document) **Error (400):**
JSON
```
{
  "error": "Invalid format | Logo file not found | Error generating export: <details>"
}
```

GET /api/export/formats
Get the list of supported export formats.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
N/A
Request Body:
N/A
Responses:
Success (200):
JSON
```
{
  "formats": ["PDF", "DOCX"],
  "default": "PDF"
}
```
Error: N/A

GET /api/export/parameters
Get details about available export formatting parameters.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
N/A

Request Body:
N/A
Responses:
Success (200):
JSON
```
{
  "parameters": {
    "page_size": { "description": "...", "values": ["A4", "LETTER", "LEGAL"], "default": "LETTER" },
    "margin_top": { "description": "...", "range": [0.1, 3.0], "default": 1.0 },
     ... (all parameters described)
  }
}
```
Error: N/A

GET /api/export/form/{form_id}/preview-params
Get formatting parameters and example URLs for exporting a specific form.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
Path Parameters:

Name	Type	Status	Description
form_id	integer	Required	ID of the form.

Query Parameters: Accepts same query parameters as /api/export/form/{form_id} to show current values
Request Body:
N/A
Responses:
Success (200):
JSON
```
{
  "form_info": { "id": 1, "title": "...", "questions_count": 5 },
  "current_parameters": { "format": "PDF", "page_size": "LETTER", ... },
  "available_parameters": { ... (details of all params) },
  "example_urls": { "minimal": "...", "with_basic_formatting": "...", "complete": "..." },
  "notes": [ ... ]
}
```
Error (404):
JSON
```
{
  "error": "Form not found"
}
```

GET /api/health/ping
Simple check if the API is responsive.

Headers:
N/A
URL Parameters:
N/A
Request Body:
N/A
Responses:
Success (200):
JSON
```
{
  "status": "pong",
  "message": "Server is responsive"
}
```
Error (500):
JSON
```
{
  "status": "error",
  "message": "Server error"
}
```

GET /api/health/status
Get detailed health status including database connectivity.
Headers:
N/A
URL Parameters:
N/A
Request Body:
N/A
Responses:
Success (200): # or 503 if unhealthy
JSON
```
{
  "server_status": "ok",
  "database_status": "ok",
  "timestamp": "...",
  "health_status": "healthy"
}
```
Error (500):
JSON
```
{
  "status": "error",
  "message": "Server error"
}
```

POST /api/cmms-configs
Create a new JSON configuration file in the 'configs' subfolder (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
filename	string	Required	Filename ending in .json.
content	object	Required	JSON content for the file.

Example:

JSON

```
{
  "filename": "settings.json",
  "content": { "theme": "dark", "notifications": true }
}
```

Responses:

Success (201):

JSON

```
{
  "message": "Configuration file created successfully",
  "config": {
    "filename": "settings.json",
    "path": "configs/settings.json",
    "content": { "theme": "dark", "notifications": true },
    "created_at": "...",
    "updated_at": "..."
  }
}
```

Error (400):

JSON

```
{
  "error": "Missing required fields | Only JSON files are supported | Filename already exists | ..."
}
```

POST /api/cmms-configs/upload

Upload a configuration file (places it in the 'configs' subfolder, Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	multipart/form-data

URL Parameters:

N/A

Request Body:

Format: Form Data **Fields:**

Name	Type	Status	Description
file	file	Required	The configuration file to upload.

Example: N/A (Form data)

Responses:

Success (201):

JSON

```
{
  "message": "Configuration file uploaded successfully",
  "config": { ... (details of the uploaded config file) }
}
```

Error (400):

JSON

```
{
  "error": "No file provided | No selected file | File type not allowed | ..."
}
```

GET /api/cmms-configs/files

List all files within the main 'uploads' directory and its subdirectories (e.g., configs, submission attachments).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

N/A

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "message": "Files retrieved successfully",
  "files": [
    {
      "filename": "settings.json",
      "path": "configs/settings.json",
      "full_path": "/path/to/app/uploads/configs/settings.json",
      "size": 123,
      "size_formatted": "123 B",
      "mime_type": "application/json",
      "created_at": "...",
      "modified_at": "...",
      "directory": "configs"
```

```
  },
  {
   "filename": "report.pdf",
   "path": "submission_101/report.pdf",
   "full_path": "/path/to/app/uploads/submission_101/report.pdf",
   "size": 102400,
   "size_formatted": "100.0 KB",
   "mime_type": "application/pdf",
   "created_at": "...",
   "modified_at": "...",
   "directory": "submission_101"
  },
  { ... }
 ],
 "total_files": 2
}
```

Error (500):

```
{
 "error": "Internal server error"
}
```

GET /api/cmms-configs/file/{filepath}

Download a specific file from the 'uploads' directory structure using its relative path.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
filepath	string	Required	Relative path to the file within the uploads directory (e.g., 'configs/settings.json' or 'submission_101/report.pdf').

Request Body:

N/A

Responses:

Success (200): File content **Error (404):**

```
{
 "error": "File not found | Invalid path"
}
```

GET /api/cmms-configs/{filename}

Load the content of a configuration file from the 'configs' subfolder.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
filename	string	Required	Name of the config file in the 'configs' directory.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
  "filename": "settings.json",
  "path": "configs/settings.json",
  "file_type": "application/json",
  "content": { "theme": "dark", "notifications": true },
  "created_at": "...",
  "updated_at": "..."
}
```

Error (404):

JSON

```
{
  "error": "Configuration file not found"
}
```

PUT /api/cmms-configs/configs/{filename}

Update the content of a JSON configuration file in the 'configs' subfolder (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
filename	string	Required	Name of the config file to update.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
content	object	Required	New JSON content for the file.

Example:

JSON

```
{
  "content": { "theme": "light", "notifications": false, "new_setting": 123 }
```

}

Responses:

Success (200):

JSON

```
{
  "message": "Configuration file updated successfully",
  "config": { ... (updated config details) }
}
```

Error (400):

JSON

```
{
  "error": "Content is required | File is not a JSON file | Error updating file | ..."
}
```

PUT /api/cmms-configs/{filename}/rename

Rename a configuration file within the 'configs' subfolder (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
filename	string	Required	Current name of the config file.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
new_filename	string	Required	New filename (must end in .json).

Example:

JSON

```
{
  "new_filename": "app_settings.json"
}
```

Responses:

Success (200):

JSON

```
{
  "message": "Configuration file renamed successfully",
  "config": { ... (details with new filename) }
}
```

Error (400):

JSON

```
{
```

"error": "New filename is required | New filename must end with .json | File not found | New filename already exists | ..."
}

DELETE /api/cmms-configs/{filename}
Delete a configuration file from the 'configs' subfolder (Admin only).
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
Path Parameters:

Name	Type	Status	Description
filename	string	Required	Name of the config file to delete.

Request Body:
N/A
Responses:
Success (200):
JSON
{
"message": "Configuration file deleted successfully"
}
Error (404):
JSON
{
"error": "Configuration file not found"
}

GET /api/cmms-configs/check
Check if the main configuration file exists in the 'configs' subfolder.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
N/A
Request Body:
N/A
Responses:
Success (200):
JSON
{
"exists": true,
"metadata": {
"filename": "cmms_config.json",
"path": "configs/cmms_config.json",

```
  "size": 512,
  "modified_at": "..."
 },
 "message": "Configuration file found in configs folder"
}
```

or exists: false

Error (500):

JSON

```
{
 "error": "Internal server error",
 "details": "<error_details>"
}
```

GET /api/export_submissions/{submission_id}/pdf

Export a specific form submission as a PDF document.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
submission_id	integer	Required	ID of the form submission to export.

Request Body:

N/A

Responses:

Success (200): File content (application/pdf) **Error (400):**

JSON

```
{
 "error": "Submission not found | Error generating PDF | Unauthorized access"
}
```

POST /api/export_submissions/{submission_id}/pdf/logo

Export a form submission as PDF with custom header image and signature formatting.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	multipart/form-data

URL Parameters:

Path Parameters:

Name	Type	Status	Description
submission_id	integer	Required	ID of the form submission.

Request Body:

Format: Form Data **Fields:**

Name	Type	Status	Description
header_image	file	Optional	Image file (PNG/JPG/JPEG) for the header.
header_opacity	float	Optional	Opacity of the header image (0-100). Default: 100.
header_size	float	Optional	Size percentage for header image (maintains aspect ratio). Default: original size (100%).
header_width	float	Optional	Explicit width in pixels for header image (overrides header_size).
header_height	float	Optional	Explicit height in pixels for header image (overrides header_size).
header_alignment	string	Optional	Alignment of header image ('left', 'center', 'right'). Default: center.
signatures_size	float	Optional	Size percentage for signature images. Default: 100.
signatures_alignment	string	Optional	Layout for signatures ('vertical', 'horizontal'). Default: vertical.

Example: N/A (Form data)

Responses:

Success (200): File content (application/pdf) **Error (400):**

JSON

```
{
 "error": "Submission not found | Header image must be PNG or JPEG | Header opacity must be a number | ..."
}
```

GET /api/counts

Get counts for all entity types accessible to the user.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include counts of soft-deleted items (Admin only). Default: false.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
 "users": 15,
 "roles": 4,
 "permissions": 50,
 "environments": 3,
 "question_types": 8,
 "questions": 120,
 "answers": 35,
```

```
 "forms": 25,
 "form_submissions": 500,
 "answers_submitted": 2500,
 "attachments": 150,
 "role_permissions": 100,
 "form_questions": 150,
 "form_answers": 60
}
```


Error (500):

JSON


```
{
 "error": "Internal server error"
}
```

GET /api/counts/{entity}

Get the count for a specific entity type (e.g., users, forms, roles).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
entity	string	Required	The entity type plural name (e.g., 'users', 'forms').

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include count of soft-deleted items (Admin only). Default: false.

Request Body:

N/A

Responses:

Success (200):

JSON


```
{
 "entity": "users",
 "count": 15,
 "include_deleted": false
}
```


Error (500):

JSON


```
{
 "error": "Internal server error"
}
```

GET /api/entity_basic/basic

Get basic representations (id, name/title) of all entity types (Admin only).

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Query Parameters:

Name	Type	Status	Description
page	integer	Optional	Page number. Default: 1.
per_page	integer	Optional	Items per page. Default: 20.
include_deleted	boolean	Optional	Include soft-deleted items. Default: false.

Request Body:

N/A

Responses:

Success (200):

JSON

```
{
 "metadata": { ... },
 "items": [
  { "id": 1, "name": "Admin", "type": "roles" },
  { "id": 1, "title": "Daily Safety Check", "type": "forms" },
  { ... }
 ]
}
```

Error (403):

JSON

```
{
 "error": "Unauthorized. Admin access required."
}
```

GET /api/entity_basic/{entity_type}/basic

Get basic representations (id, name/title) for a specific entity type.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:

Path Parameters:

Name	Type	Status	Description
entity_type	string	Required	Entity type (e.g., 'users', 'forms').

Query Parameters:

Name	Type	Status	Description
page	integer	Optional	Page number. Default: 1.
per_page	integer	Optional	Items per page. Default: 20 (max 100).
include_deleted	boolean	Optional	Include soft-deleted items (Admin only). Default: false.

Request Body:
N/A
Responses:
Success (200):
JSON
```
{
 "metadata": { ... },
 "items": [
  { "id": 1, "name": "Daily Safety Check" },
  { "id": 2, "name": "Monthly Maintenance Report" },
  { ... }
 ]
}
```
Error (400):
JSON
```
{
 "error": "Invalid entity type | Invalid pagination parameters"
}
```

GET /api/entity_basic/{entity_type}/{entity_id}/basic
Get basic representation (id, name/title) for a specific entity instance.
Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>

URL Parameters:
Path Parameters:

Name	Type	Status	Description
entity_type	string	Required	Entity type (e.g., 'users', 'forms').
entity_id	integer	Required	ID of the entity.

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include if soft-deleted (Admin only). Default: false.

Request Body:
N/A
Responses:
Success (200):
JSON
```
{
 "id": 1,
 "name": "Daily Safety Check"
}
```
Error (404):
JSON
```
{
```

```
 "error": "Entity not found"
}
```

POST /api/entity_basic/{entity_type}/basic/batch

Get basic representations for multiple entities of the same type by their IDs.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

Path Parameters:

Name	Type	Status	Description
entity_type	string	Required	Entity type (e.g., 'users', 'forms').

Query Parameters:

Name	Type	Status	Description
include_deleted	boolean	Optional	Include soft-deleted items (Admin only). Default: false.

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
ids	array	Required	List of entity IDs.

Example:

JSON

```
{
 "ids": [1, 5, 12]
}
```

Responses:

Success (200):

JSON

```
{
 "items": [
  { "id": 1, "name": "Daily Safety Check" },
  { "id": 5, "name": "Incident Report" },
  { "id": 12, "name": "Equipment Calibration" }
 ]
}
```

Error (400):

JSON

```
{
 "error": "Invalid request format. Expected 'ids' list in JSON body"
}
```

Report API Request Examples

This document provides comprehensive examples for the `/api/reports/generate` endpoint, showcasing different report types, formats, and parameter combinations. These examples will help developers understand how to effectively use this API for generating various kinds of reports from the system.

Basic Report Requests

Example 1: Basic Users Report (XLSX Default)

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "users"
}
```

This simple request generates a users report with default columns, no filters, and default sorting. The output is XLSX format (default).

Example 2: Form Submissions with Column Selection

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "form_submissions",
  "columns": [
    "id",
    "form.title",
    "submitted_by",
    "submitted_at",
    "form.creator.username",
    "created_at"
  ]
}
```

This request generates a form submissions report with only the specified columns.

Example 3: Filtered Environment Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "environments",
```

```
  "filters": [
    {
      "field": "name",
      "operator": "like",
      "value": "Production"
    },
    {
      "field": "created_at",
      "operator": "gte",
      "value": "2025-01-01T00:00:00Z"
    }
  ]
}
```

This request filters environments with names containing "Production" created on or after January 1, 2025.

Example 4: Sorted Role Permissions Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "role_permissions",
  "sort_by": [
    {
      "field": "role.name",
      "direction": "asc"
    },
    {
      "field": "permission.entity",
      "direction": "asc"
    }
  ]
}
```

This request sorts role permissions by role name and permission entity in ascending order.

Alternative Output Formats

Example 5: CSV Format Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "forms",
  "output_format": "csv",
  "columns": [
```

```
    "id",
    "title",
    "description",
    "creator.username",
    "is_public",
    "created_at"
  ]
}
```

This request generates a forms report in CSV format instead of the default XLSX.

Example 6: PDF Format Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "form_submissions",
  "output_format": "pdf",
  "report_title": "Form Submissions Summary Report",
  "filters": [
    {
      "field": "submitted_at",
      "operator": "between",
      "value": ["2025-04-01T00:00:00Z", "2025-04-30T23:59:59Z"]
    }
  ]
}
```

This request generates a PDF report for form submissions in April 2025 with a custom title.

Example 7: DOCX Format Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "users",
  "output_format": "docx",
  "filename": "active_users_report",
  "filters": [
    {
      "field": "is_deleted",
      "operator": "eq",
      "value": false
    }
  ]
}
```

This request generates a Word document report for active (not deleted) users with a custom filename.

Example 8: PPTX Format Report (Form Submissions)

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "form_submissions",
  "output_format": "pptx",
  "report_title": "Monthly Submission Trends",
  "include_data_table_in_ppt": true,
  "max_ppt_table_rows": 10
}
```

This request generates a PowerPoint presentation with submission trends and includes a data table with up to 10 rows.

Advanced Report Requests

Example 9: Complex Filtered and Sorted Report

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "form_submissions",
  "columns": [
    "id",
    "form.title",
    "submitted_by",
    "submitted_at",
    "answers.Safety Check Completed",
    "answers.Equipment Condition",
    "answers.Notes"
  ],
  "filters": [
    {
      "field": "form.id",
      "operator": "eq",
      "value": 12
    },
    {
      "field": "submitted_at",
      "operator": "between",
      "value": ["2025-03-01T00:00:00Z", "2025-03-31T23:59:59Z"]
    },
    {
```

```
      "field": "answers_submitted.question",
      "operator": "eq",
      "value": "Safety Check Completed"
    },
    {
      "field": "answers_submitted.answer",
      "operator": "eq",
      "value": "Yes"
    }
  ],
  "sort_by": [
    {
      "field": "submitted_at",
      "direction": "desc"
    }
  ],
  "output_format": "xlsx",
  "filename": "march_safety_checks_report"
}
```

This complex request generates a report of March 2025 form submissions for form ID 12 where "Safety Check Completed" answer is "Yes", showing specific question answers and sorted by submission date.

Example 10: Multi-Entity Report (All Tables)

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": "all",
  "output_format": "xlsx",
  "filename": "full_system_data_report"
}
```

This request generates a multi-sheet XLSX workbook containing data from all available entities, using default columns and settings for each.

Example 11: Multi-Entity Specific Selection

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": ["users", "forms", "form_submissions"],
  "output_format": "csv",
  "filename": "user_form_activity"
}
```

This request generates a ZIP archive containing CSV files for users, forms, and form submissions data.

Example 12: Multi-Entity with Sheet Name Customization

```
POST /api/reports/generate HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Content-Type: application/json

{
  "report_type": ["users", "environments"],
  "output_format": "xlsx",
  "users_sheet_name": "System Users",
  "environments_sheet_name": "Available Environments",
  "filename": "system_configuration",
  "table_options": {
    "style": "Table Style Medium 2",
    "banded_rows": true
  }
}
```

This request generates an XLSX workbook with customized sheet names and table styling.

Database Schema Request

Example 13: Database Schema Information

```
GET /api/reports/schema HTTP/1.1
Host: example.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

This request retrieves the database schema information (only accessible to administrators).

GET /api/reports/schema

Retrieves the database schema information for generating reports. This endpoint provides detailed metadata about available tables, columns, and relationships to help with report creation.

Headers:

Name	Purpose	Status	Example
Authorization	Authentication token	Required	Bearer <JWT_TOKEN>
Content-Type	Specify content type	Required	application/json

URL Parameters:

N/A

Request Body:

Format: JSON **Fields:**

Name	Type	Status	Description
report_type	string	Optional	Type of report to get schema for (e.g., 'users', 'forms', 'submissions', 'all'). Default: 'all'

Example:

```
{
 "report_type": "all"
}
```

Responses:

Success (200):

```
{
 "database_info": {
  "application_models": 14,
  "name": "database_name",
  "total_tables": 15,
  "version": "PostgreSQL version info"
 },
 "model_mapping": {
  // Maps model names to table names
 },
 "tables": {
  // Detailed table schema information including:
  // - columns (name, type, nullable, etc.)
  // - primary keys
  // - foreign keys and relationships
  // - row counts (active, deleted, total)
 }
}
```

Error (400):

```
{
 "error": "Invalid report type"
}
```

Error (401):

```
{
```

"error": "Unauthorized access"
}

Error (403):

{
"error": "Insufficient permissions, Admin role required"
}

Notes:

- This endpoint is intended for Admin users to understand the database structure for creating custom reports
- The schema information is useful for constructing advanced filters and column selections when using the POST /api/reports/generate endpoint
- Response includes comprehensive metadata about each table's structure, relationships, and row counts

---

This includes documentation for the main endpoints identified in the provided files. Additional endpoints related to permissions, environments, question types, answers, form questions, form answers, and role permissions follow similar patterns and are largely administrative or relate to setting up the structure of forms. Their specific parameters and responses can be inferred from the corresponding view files (permission_views.py, environment_views.py, etc.).