Category API Documentation

This documentation describes the RESTful API endpoints for performing CRUD operations on the `category` table. The API is built using Node.js with Express.js and connects to a MySQL database (via the `./connection` module).

Base URL

- **Protocol**: `http://`
- **Hostname**: `localhost` (or your server IP/domain)
- **Port**: `5000`
- **Example Base URL**: `http://localhost:5000`

Middleware

- Supports `application/x-www-form-urlencoded` (via `urlencoded` parser).
- Supports `application/json` (via `body-parser.json`).

Common Response Formats

- **Success (Create/Update/Delete)**: `[{ "error": "no" }, { "success": "yes" }, { "message": "operation message" }]`
- **Not Found (Update/Delete)**: `[{ "error": "no" }, { "success": "no" }, { "message": "category not found" }]`
- **Error (Missing Input)**: `[{ "error": "input missing" }]`
- **Error (Database)**: `[{ "error": "oops, something went wrong, please try after sometimes" }]`
- **Read Success**: JSON array of category objects: `[{ "id": number, "name": string, "photo": string, "detail": string }, ...]`

Endpoints

1. Get All Categories (Paginated)

- **Method**: `GET`
- **Path**: `/category/:start?` (optional path parameter; defaults to `0` if omitted)
- **Description**: Retrieves a list of non-deleted categories (`is_deleted = 0`), ordered by `id` descending. Supports basic pagination with a fixed page size of 20 records.
- **Parameters**:

Name	Type	Location	Required	Description	Default
start	number	Path	No	Starting offset for pagination	0

- **Request Example**:

```
text
GET http://localhost:5000/category/0
```

- **Response Example (200 OK)**:

```
JSON
[
  {
    "id": 1,
    "name": "Electronics",
    "photo": "electronics.jpg",
    "detail": "Category for electronic gadgets"
  },
  ...
]
```

- **Error Responses**:
  - Database errors are logged to console; response is empty or error object (not explicitly handled in code).

2. Create New Category

- **Method**: `POST`
- **Path**: `/category`
- **Description**: Inserts a new category into the database. All fields are required.
- **Request Body** (JSON or URL-encoded):

Field	Type	Required	Description
name	string	Yes	Category name
photo	string	Yes	Photo URL or filename
detail	string	Yes	Category description

- **Request Example**:

```
JSON

POST http://localhost:5000/category
Content-Type: application/json

{
  "name": "Books",
  "photo": "books.jpg",
  "detail": "Category for literature"
}
```

- **Response Examples**:
  - **Success (201 Created)**:

```
JSON

[
  { "error": "no" },
  { "success": "yes" },
  { "message": "category inserted" }
]
```

  - **Missing Input (400 Bad Request)**:

```
JSON

[{ "error": "input missing" }]
```

3. Update Existing Category

- **Method**: `PUT`
- **Path**: `/category`
- **Description**: Updates an existing category by `id`. Performs a soft update (no hard delete). Note: The request body uses `title` for the name field, but the database column is `name`—this is handled in the SQL (value from `title` is assigned to `name`).
- **Request Body** (JSON or URL-encoded):

Field	Type	Required	Description
title	string	Yes	Updated category name
photo	string	Yes	Updated photo URL/filename
detail	string	Yes	Updated description
id	number	Yes	Category ID to update

- **Request Example**:

```
JSON

PUT http://localhost:5000/category
Content-Type: application/json

{
  "title": "Updated Books",
  "photo": "updated_books.jpg",
  "detail": "Updated literature category",
  "id": 1
}
```

- **Response Examples**:

  - **Success (200 OK)**:

```
JSON

[
  { "error": "no" },
  { "success": "yes" },
  { "message": "category updated" }
]
```

  - **Not Found (404 Not Found)**:

```
JSON

[
  { "error": "no" },
  { "success": "no" },
  { "message": "category not found" }
]
```

  - **Missing Input (400 Bad Request)**:

```
JSON

[{ "error": "input missing" }]
```

4. Delete Category (Soft Delete)

- **Method**: `DELETE`
- **Path**: `/category`
- **Description**: Soft-deletes a category by setting `is_deleted = 1` (not hard delete). Requires category `id`.
- **Request Body** (JSON or URL-encoded):

Field	Type	Required	Description
id	number	Yes	Category ID to delete

- **Request Example**:

```
JSON

DELETE http://localhost:5000/category
Content-Type: application/json

{
  "id": 1
}
```

- **Response Examples**:
  - **Success (200 OK)**:

```
JSON

[
  { "error": "no" },
  { "success": "yes" },
  { "message": "category deleted" }
]
```

  - **Not Found (404 Not Found)**:

```
JSON

[
  { "error": "no" },
  { "success": "no" },
  { "message": "category not found" }
]
```

  - **Missing Input (400 Bad Request)**:

```
JSON

[{ "error": "input missing" }]
```

Notes

- **Pagination**: Fixed limit of 20 records per request. Use `start` to offset (e.g., `/category/20` for next page).
- **Error Handling**: Database errors are logged to the console but return a generic error message. No detailed error codes.
- **Security**: No authentication or validation (e.g., SQL injection prevention relies on parameterized queries). Add middleware for production.
- **Testing**: Use tools like Postman or curl to test endpoints.
- **Database Schema Assumption**: Table `category` with columns: `id` (PK), `name`, `photo`, `detail`, `is_deleted` (boolean/tinyint).