**Hypothetical API Design for the Project**

This API is designed to handle operations such as adding, removing, and updating elements. It focuses on simplicity, scalability, and long-term maintenance. The API adheres to RESTful principles and includes endpoints for basic CRUD operations.

Endpoints

1. Fetch All Items

Endpoint: GET /api/cards

Description: Retrieves the list of all cards with their current order and details.

Response:

[

{

"id": 1,

"type": "bank-draft",

"title": "Bank Draft",

"position": 0,

"thumbnail": "thumbnail1.png"

},

{

"id": 2,

"type": "bill-of-lading",

"title": "Bill of Lading",

"position": 1,

"thumbnail": "thumbnail2.png"

}

]

Status Codes:

200 OK: Successfully fetched all items.

2. Fetch a Single Item

Endpoint: GET /api/cards/{id}

Description: Retrieves the details of a specific card by its ID.

Response:

{

"id": 1,

"type": "bank-draft",

"title": "Bank Draft",

"position": 0,

"thumbnail": "thumbnail1.png"

}

Status Codes:

200 OK: Successfully fetched the item.

404 Not Found: Item with the given ID does not exist.

3. Add a New Item

Endpoint: POST /api/cards

Description: Adds a new card to the list.

Request Body:

{

"type": "invoice",

"title": "Invoice",

"position": 2,

"thumbnail": "thumbnail.png"

}

Response:

{

"message": "Card added successfully.",

"id": 3

}

Status Codes:

201 Created: Successfully added the card.

400 Bad Request: Invalid request data.

4. Update an Item

Endpoint: PUT /api/cards/{id}

Description: Updates the details of an existing card.

Request Body:

{

"title": "Updated Title",

"position": 1

}

Response:

{

"message": "Card updated successfully."

}

Status Codes:

200 OK: Successfully updated the item.

404 Not Found: Item with the given ID does not exist.

400 Bad Request: Invalid request data.

5. Delete an Item

Endpoint: DELETE /api/cards/{id}

Description: Removes a card from the list by its ID.

Response:

{

"message": "Card deleted successfully."

}

Status Codes:

200 OK: Successfully deleted the item.

404 Not Found: Item with the given ID does not exist.

**Additional Considerations for Long-Term Maintenance**

1. Pagination and Filtering

For large datasets, include query parameters for pagination and filtering:

Example:

GET /api/cards?limit=10&offset=0&type=bank-draft

2. Validation

Use robust input validation for all endpoints to ensure data integrity and security.

3. Error Handling

Standardize error messages to simplify debugging.

Example:

{

"error": {

"code": 400,

"message": "Invalid request data",

"details": ["Field 'title' is required."]

}

}

4. Authentication

Secure the API using token-based authentication (e.g., JWT). Add an Authorization header to all requests:

Authorization: Bearer <token>

5. Rate Limiting

Implement rate limiting to prevent abuse:

Allow a certain number of requests per minute per user/IP.