Templates Management System

You've been given the following task to implement the back-end side of a template management system.
A **template** is a reference to a Xara Cloud document.
A **category** is a grouping that contains templates and also other categories.

During a requirements elicitation meeting, loads of questions were asked out during the divergence phase:

- Can a template belong to more than one category?
- What would happen to such a template if it's delete from one category?
- Can we have categories that _cross along_ elements of other categories?
- How fast can we retrieve categories?
- If we later tag categories and/or templates how fast can we retrieve those that match the tags?

  and so on...

But, at least for the **proof of concept** you'll be working on, the requirements were restricted to the following and are to be delivered in a micro-service.

1. A template must have a `displayName` (meaning that you should ignore momentarily any `refId` to the actual Xara Cloud document or alike and just make It capable of bearing a name).
2. A category must have a `displayName`.
3. You must be able to **create** via RESTful API a **template** with the following inputs: `displayName` (of the template), `categoryId` (the category where the template belongs). This must be recorded in the database.
4. You must be able to **create** via RESTful API a **category** with the following inputs: `displayName` (of the category) , `categoryId` (the category to which this new category will belong). This must be recorded in the database.
5. You must be able to **delete** via RESTful API any **template** or **category**; deleting a category means that the categories and templates that belong to the first must also be deleted. This must be reflected in the database.
6. You must be able to **move** via RESTful API a **category/template** to another category. Just like when deleting, moving a category also takes along the categories and templates group into the first. This must be reflected in a database.
7. All operations that write to the database MUST be ACID.

Note: The RESTful APIs must accept and respond with JSON

What you can do (and probably should)

- Rename the parameters of any method/function.
- Ask further questions.
- Your solution must comply with the requirements given, of course, and also be easily extendable for those questions asked in the requirements elicitation meeting.
- **Your solution is expected to be robust in terms of validation and error handling.**

What you cannot do

- Skip/Alter any of the requirements other than what's previously mentioned on what you can/should do.

Integration Testing

You're responsible also for creating a script for testing (mocha suite). It should perform the following operations against your solution and in the given order:

1. Insert the category 'Travel Destinations'.
2. Insert the category 'Mexico' grouped under 'Travel Destinations'.
3. Insert the category 'Germany' grouped under 'Travel Destinations'.
4. Insert the template 'acapulco' grouped under 'Mexico'.
5. Insert the template 'munich' grouped under 'Germany'.
6. Insert the category 'Beach' grouped under 'Germany' (this is clearly a semantic mistake, but please do it and play along).
7. Insert the template 'los cabos' grouped under 'Beach'.
8. Insert the category 'Exclusive' grouped under 'Beach'.
9. Insert the template 'la paz' grouped under 'Exclusive'.
10. Move the category 'Beach' to the template 'acapulco' (this should fail and nothing should be altered in the db as a consequence).
11. Move the category 'Beach' to the category 'Mexico' (this is where we correct step 6's mistake)
12. Delete the category 'Beach'.
13. Insert the template 'memo' which is not grouped under no category.

You have the following technologies at your disposal:

- Node vx.x.x
- Mocha 3+
- MongoDB 4+
- Any open source package available through npm

Recommendations

This is the part where I usually tell you 'remember to read everything before implementing...'
But let me give you a more concise and valuable piece of advice: this list of requirements, like almost any other you will find in real life, are written from easy to difficult.
This is due to our human tendency of explaining any matter from simple to complicated, when we speak a natural language. But you are NOT going to program in a natural language! If I were you, I wouldn't even open my editor before making sure that my idea works with the most simple and robust of tools: pen a paper. Seriously, do it.