**Schema Validation in Mongo-db**

MongoDB is widely recognized as a schema-less database, a core characteristic that provides flexibility. However, there are scenarios where maintaining a strict structure for data within collections is necessary. In such cases, schema validation in MongoDB becomes essential, enabling the enforcement of a structured, SQL-like schema within collections to ensure data integrity and consistency. Its more advanced use case is found while using mongoose. Here comes the role of schema validation in mongodb.

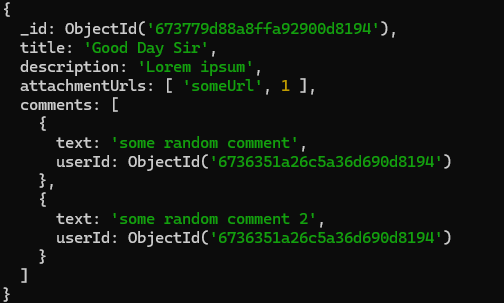
Schema Validation in mongodb can be done by two ways :

1. Manually Creating a Collection.
2. Modifying The Existing Collection.

* **Manually Creating a Collection :** By default in mongodb collections are generated on the fly or lazily when first transaction occurs with them and they are generate without any validation configuration.

On the other hand we generate a collection manually using *createCollection(‘collectionName’, options)*method, with it we can also apply validations to our collection and many more other configurations.

For Example: To create a Post collection which stores the document (all fields are required) that looks like:



We need to create the Post Collection Manually like this:

**Note:** Along side validator, we can also set the validationLevel, validationAction etc.



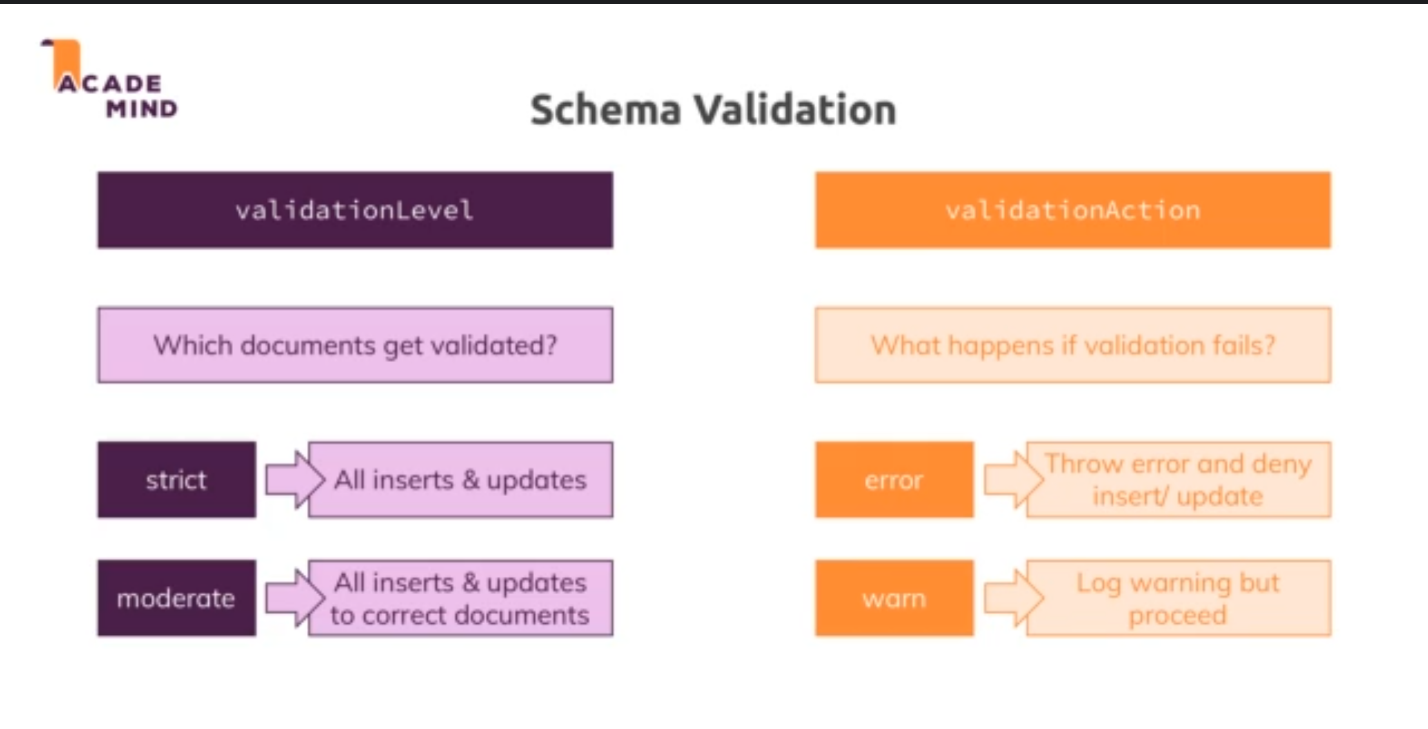
* **Modifying The existing Collection**:  If the collection already exist in your database and you want to apply validations to the existing documents or want to change the validationLevel or validationAction.

**validationLevel** : There are two types of validation level, that we can set to a collection

* + - **strict:** will validate all the insert/update operation, as well as allow updation on only corrected documents.
    - **Moderate:** will validate all the insert operations only.

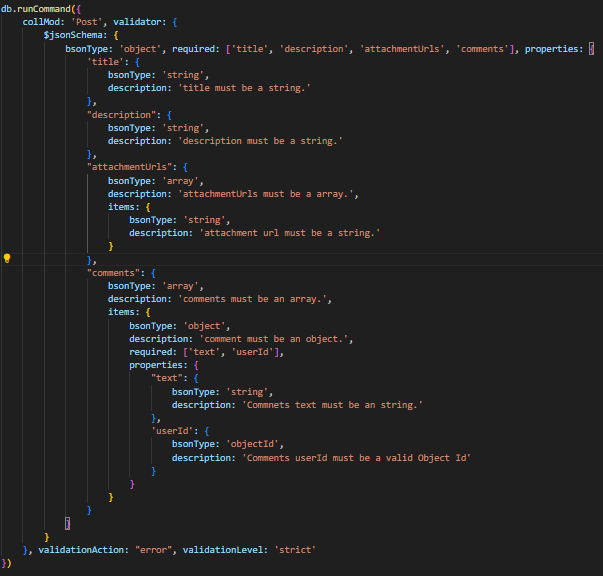
**validationAction :**  There are two types of validation actions, that we can set to a collection on how to respond when validation fails.

* + - **error:**  will throw a validation failed error and restrict the document insertion/updation.
    - **warn:** will log a warning in the log file but allow the insertion/updation of the document.



we can utilize the *db.runCommand({field1: value, field2: value, field3: value})* Method as shown below:

**Note:** we can omit the validator field if the validation is already set at the time of collection creation.



**Terminologies:**

* + **collMod (Collection Modifier) :**  Defines on which collection the command is to be run or which collection is to be modified.
  + **validator :** Used to define the validation schema for collection.
  + **validationLevel:** Used to define the scope of the validation (strict or moderate).
  + **validationAction:** Used to define the action to be taken when validation fails (error or warn).
  + **$jsonSchema:** Used to define the structure of the collection.
  + **bsonType:**  Used to define the type of the property/field (mongodb specific types) [can be seen in mongodb compass or official docs].
  + **required:** Used to define the fields/properties of the document that are required.
  + **properties:** Used to define the types or error text for the document fields.
  + **description:**  Used to define the helperText/ErrorMessage when validation fails.
  + **Items:** Used to define the type of the data that is to be stored inside the array.