Index:

* **mixed** type in mongoose schema
* **Date and moment**
* **findOne vs find**
* **CRUD** in mongo
  + esp **updateMany** and **findOneAndUpdate**
  + **new** flag in update operations
  + **upsert** flag in update operations
* **isDeleted** key in mongoose schema
* [**Assignment**](#_o5lbjx9k7822) **( we have not discussed ref and populate yet)**

# Content

**Mixed** data type



e.g. from schema

summary: mongoose.Schema.Types.Mixed,

summary key can have all these types of book summaries now:

// summary : "this is a suspense novel"

// summary : ["ch1: Intro to backend", "ch2: intro to mongodb", "ch3: intro to nodejs:"]

// summary : {

// chapter1: "How to get started with tech",

// chapter2: "lets start with basics"

// }

**Date type** and Date.now, new Date(), momentJs,

* **Date** maniupaltions( brief intro) + shared this resource on **moment** module :<https://www.sitepoint.com/managing-dates-times-using-moment-js/>

**find vs findOne**

* find gives all the documents that match the condition | findOne gives only the first document that matches
* find gives an array | findOne gives an object
* If no match is found - find gives [] empty array which is a truthy value || findOne gives null which is a falsey value …… user=findOne() and then if(user) :- will fail with find as it will be truthy even if no user is found

**updateMany and findOneAndUpdate**

// let books = await BookModel.updateMany ( {isPublished: false } , {author : "PK"} ); // first json is the query condition || second condition is the required update or change

// let books = await BookModel.findOneAndUpdate( {isPublished: true } , {author : "Sabiha"} ); // it updates only the first matching doc

// let books = await BookModel.findOneAndUpdate( {isPublished: true } , {author : "Sabiha 3"} , { new: true} ); // third param : new: true - will give you the updated document

**upsert: true** - it finds and updates the document but if the doc is not found(i.e it does not exist) then it creates a new document

// let books = await BookModel.findOneAndUpdate( {bookName : "Hi Pritesh2" } , {bookName : "Hi My New Book" , ISBN : "basd87g8h7a88b"} , { upsert: true} );

**isDeleted flag**

// how to delete a document: never ever use remove.. always maintain a flag(a key in schema) "isDeleted: false" and whenever a doc is being deleted change this to "isDeleted: true” (mark dirty)

# ASSIGNMENT : **( dont use ref and populate)**

You have to replicate the below data in your database. With this in mind, create a node application and APIs to do the following:

1. Write down the schemas for book and authors (keeping the data given below in mind). Also create the documents (corresponding to the data given below) in your database.

2. CRUD operations. Write API's to do the following:

* Write create APIs for both books and authors ---> If author\_id is not available then do not accept the entry(in neither the author collection nor the books collection)
* List out the books written by "Chetan Bhagat" ( this will need 2 DB queries one after another- first query will find the author\_id for "Chetan Bhagat”. Then next query will get the list of books with that author\_id )
* find the author of “Two states” and update the book price to 100; Send back the author\_name and updated price in response. ( This will also need 2 queries- 1st will be a findOneAndUpdate. The second will be a find query aith author\_id from previous query)
* Find the books which costs between 50-100(50,100 inclusive) and respond back with the author names of respective books..

bookModel.find( { price : { $gte: 50} , price: {$lte: 100} } ) // WRONG

bookModel.find( { price : { $gte: 50, $lte: 100} } ).select({ author\_id :1})..run a map(or forEach) loop and get all the authorName corresponding to the authorId’s ( by querying authorModel)

**DATA:**

// \_id:ObjectId("8781263871293"), \_id will be automatically generated

**Authors:**

{

author\_id:1,

author\_name:"Chetan Bhagat",

age:25,

address:"New delhi"

} ,

{

author\_id:2,

author\_name:"J.k Rowling",

age:60,

address:"Britain"

} ,

{

author\_id:3,

author\_name:"Ramanujan",

age:100,

address:"Tamilnadu"

}

**Books:**

{

name:"Two states",

author\_id:1,

price:50,

ratings:4.5,

} ,

{

name:"Five Point Someone",

author\_id:1,

price:50,

ratings:4.5,

} ,

{

name:"The 3 Mistakes of My Life",

author\_id:1,

price:50,

ratings:4.5,

} ,

{

name:"One Arranged Murder",

author\_id:1,

price:50,

ratings:4.5,

} ,

{

name:"Harry Porter",

author\_id:2,

price:50,

ratings:4.5,

} ,

{

name:"Harry Porter",

author\_id:2,

price:50,

ratings:4.5,

}