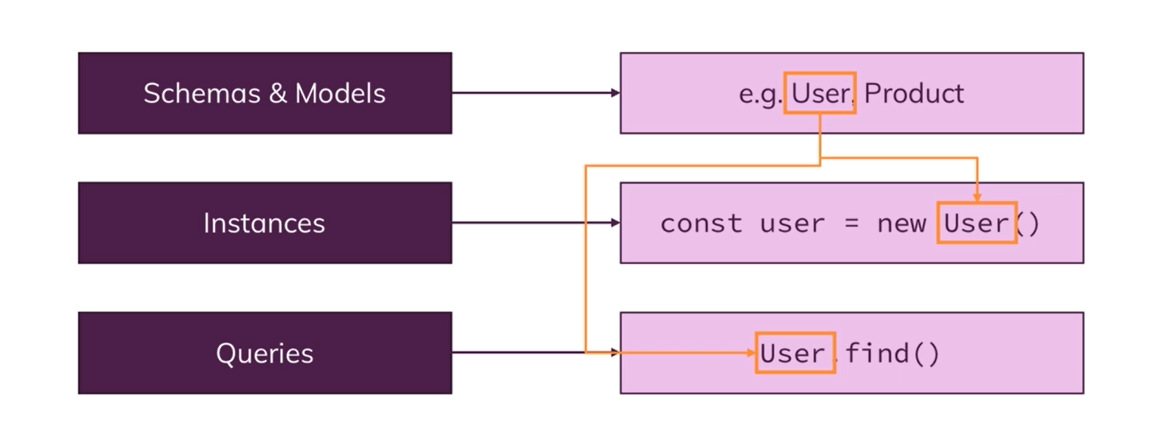
**Mongoose**

**Table of Content:**

* What is Mongoose
* Using Mongoose in Node.js App

1. **What is Mongoose**a. Like mysql has sequelize, mongoDB has mongoose.  
   b. it is ODM (Object Document Mapping) Library which is very much similar to ORM (Object Relation Mapping) Library (eg. sequelize)  
   c. core concept of mongoose  
   
2. **Set-up**a. Install mongoose package.  
   b. In this, we don’t need database.js file. Reason all functionality of that file is handled by one method of mongoose package.  
   c. import mongoose in app.js file and use connect() method to connect with db.  
    eg. mongoose.connect(<DB-URL>);
3. **Creating a model & schema**a. to create a schema we have to import mongoose package.  
    const mongoose = require('mongoose');  
   b. to create a schema we have to Schema constructor of mongoose package.  
    const Schema = mongoose.Schema;  
    const productSchema = new Schema();  
   c. this schema object take map which define your document(data schema) where key is field name and value is property of that field (as in mongodb is nosql database still there are some structure in data)  
   d. we export schema using model property of mongoose.   
    module.exports = mongoose.model('Product', productSchema);1st parameter is name of model and 2nd parameter is schema
4. **Adding data to Db.**a. to save data we will import model that we have export in schema.  
   b. to add data we will create an object of that model and we will pass a map of data.  
   c. on the reference of object we use save() method to store data to db and on then() on it we will get result object.
5. **Fetching data from Db**Fetching All dataa. to get data we will use find() on the model that we have imported.   
    Product.find()  
   b. on then() method we will get all the products in this case.  
   c. find() will not return cursor to get cursor object we have to use  
    Product.find().cursor().next()  
     
   Fetching specific document  
   a. to get specific document we will findById(id) property of the model we have imported.  
   b. id we pas here is string not ObjectId reason mongoose convert string to ObjectId.  
     
   select()  
   a. this method is used to fetch selected field from the document.  
   eg: Product.find().select('title price')  
   this will include id default  
   b. to remove id  
    Product.find().select('title price -\_id')
6. **Updating data**a. to update data, we will findById() this will return a document (in our case product) where we can use all mongoose methods.  
   b. in then(), we will update all the data. As,  
    product.title = updatedTitle;  
   c. after that we will use save() on the product which will modify the data.
7. **Deleting data from Db**a. to delete document we will use findByIdAndRemove(id) or findByIdAndDelete(id)
8. **Add Relation**a. to relate product with user add userId in product schema where it will refer to user schema.  
   userId : {type: Schema.Types.ObjectId, ref: ‘User’, required: true}  
   value of ref should be same as model name.  
   Similar, in cart productId refer to product model.  
   b. now in product document we have userId of type ObjectId now incase if we also want to fetch user data with product we can use populate() method on Product.find()  
   populate(‘key’) where key is the field which we want to populate.  
   in our case is userId.  
   c. populate all take optional 2nd parameter which tell which fields from document to be fetch. (similar to select())  
   eg. populate('userId', 'name email')  
   this will include id default.  
   d. if we want don’t want id   
    populate('userId', 'name email -\_id')  
   ######### for old version ##########  
   e. populate don’t return promise to get promise use execPopulate() after populate()  
   populate('userId', 'name email -\_id'). execPopulate().then()
9. **userSchema.methods**a. is used to add methods inside schema which we can use in model