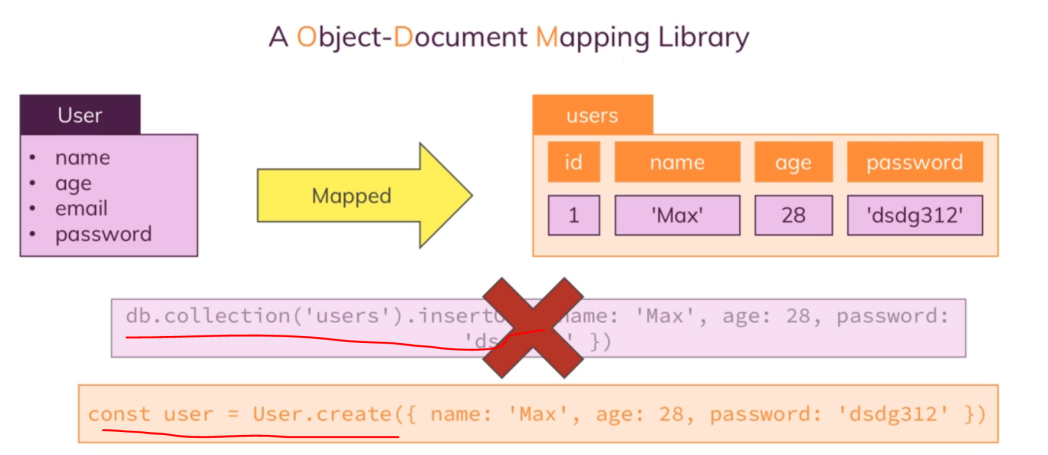
**Official docs of mongoose :** [**https://mongoosejs.com/docs/guide.html**](https://mongoosejs.com/docs/guide.html)

**Installation guidance of mongoose :** [**https://mongoosejs.com/docs/**](https://mongoosejs.com/docs/)

What is mongoose

Mongoose is just like sequelize in which we do not have to write the query from scratch instead it provides a library from which we can use our mongoDb querys

**So mongoose is just – object- Document Mapping Library**



So now instead of creating database path we can just do it in one line in app.js file where we require the mongoose and do connect operation with your id so we do not call the database again and again in files

const mongoose = require('mongoose');

mongoose.connect('mongodb+srv://mayank:Abcd4321@cluster0.f5vog.mongodb.net/shop?retryWrites=true&w=majority')

.then(() => {

  app.listen(3000);

})

Now we now have to define the product data page by using mongoose

here we are doing it by using schemas of mongoose and we are doing it because schema gives us the order or design of our product so if we use schema then data will looks like in reasonable formate

const mongoose = require('mongoose');

// we are using Schema to structure our database

const Schema = mongoose.Schema;

const productSchema = new Schema({

  title: {

    type: String,

    required: true

  },

  price: {

    type: Number,

    required: true

  },

  description: {

    type: String,

    required: true

  },

  imageUrl: {

    type: String,

    required: true

  }

})

// it exports the above product class and and do interact with the name of product in database

**module.exports = mongoose.model('product',productSchema);**

here we export our product schema inside the mongoose model with the name of product so it will define inside the database a by the name of **products collection (it convert into plural of product)**

the use of that is now it give us some inner functions that we can use from it and the one we are using next is save(), in admin controller

exports.postAddProduct = (req, res, next) => {

  const title = req.body.title;

  const imageUrl = req.body.imageUrl;

  const price = req.body.price;

  const description = req.body.description;

**const product = new Product({**

**title: title,**

**price: price,**

**description: description,**

**imageUrl: imageUrl,**

**}**

  );

  product

**.save() // these method is called from exports of product model**

    .then(result => {

      // console.log(result);

      console.log('Created Product');

      res.redirect('/admin/products');

    })

    .catch(err => {

      console.log(err);

    });

};

So we can also fetch our data from shop controller to the user side and we can do it by **find()** which is the mongoose method to fetchAll()

exports.getProducts = (req, res, next) => {

  Product**.find()**

    .then(products => {

      res.render('shop/product-list', {

        prods: products,

        pageTitle: 'All Products',

        path: '/products'

      });

And findById is used to find the product by id

Now if we want to update our product then we can do it by just calling the product exports class and reassign the values

exports.postEditProduct = (req, res, next) => {

  const prodId = req.body.productId;

  const updatedTitle = req.body.title;

  const updatedPrice = req.body.price;

  const updatedImageUrl = req.body.imageUrl;

  const updatedDesc = req.body.description;

**Product.findById(prodId)**

**.then(product => {**

**product.title = updatedTitle,**

**product.price = updatedPrice,**

**product.description = updatedDesc,**

**product.imageUrl = updatedImageUrl**

**return product.save(); // called by mongoose model**

**// in mongoose we can do the operation on specific product like this**

  })

  .then(result => {

    console.log('UPDATED PRODUCT!');

    res.redirect('/admin/products');

  })

  .catch(err => console.log(err));

};

And also we can delete by this method

exports.postDeleteProduct = (req, res, next) => {

  const prodId = req.body.productId;

  // we can delete by this function

  Product**.findByIdAndDelete**(prodId)

    .then(() => {

      console.log('DESTROYED PRODUCT');

      res.redirect('/admin/products');

    })

    .catch(err => console.log(err));

};

Now we just create user

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const userSchema = new Schema({

  name: {

    type: String,

    required: true

  },

  email: {

    type: String,

    required: true

  },

  cart: {

    cart: {

      items: [{

        productId: { type: Schema.Types.ObjectId, required: true},

        quantity: { type: Number, required: true}

      }]

    }

  }

});

**module.exports = mongoose.model('user',userSchema);**

and also create a user in app.js so the user can be created from above code. for that call user

const User = require('./models/user');

mongoose.connect('mongodb+srv://mayank:Abcd4321@cluster0.f5vog.mongodb.net/shop?retryWrites=true&w=majority')

.then(result => {

  User.findOne() // check have any user

  .then(user => {

    if(!user) { // if no user get from above then only

      const user = new User({

        name: 'Jay',

        email: 'jay12@gmail.com',

        cart: {

          items: []

        }

      });

      user.save();

    }

  });

  app.listen(3000);

})

Manage Relation or association

Have to add user id in the product model page because by that only it will create a relation between the product and the user

We can simply do it by calling Schema type Object id which will get the id from ref in the box

 userId: {

    type: Schema.Types.ObjectId, // take objectId from ref user

**ref: 'User'**,

    required: true

  }

Created inside product model and below in the user model

cart: {

    cart: {

      items: [{

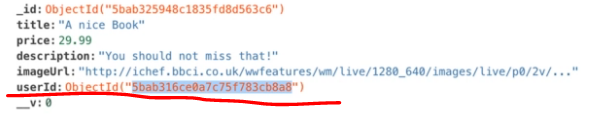
        productId: { type: Schema.Types.ObjectId,**ref:'product'**,required: true},

        quantity: { type: Number, required: true}

      }]

    }

Result is



To get the specific fields from the above modules we can use

exports.getProducts = (req, res, next) => {

  Product.find()

    .select('title price -\_id') // it will give the specific field not the hole

    .populate('userId','name') // it will populate the name from the user id which is inside the product

We also can create our own function or method inside mongoose by just using Schema .method

/ it is used to add your own method in your mongoose library

**userSchema.methods**.addToCart = function(product) {

now let us built the cart in shop js controller or creating a cart we want the list of product that we have and as we know that we already have product id in our user the product that user selected and these products are the item of our cart so we can excess the cart .items and that product id of the user

// here we use populate to get particular item from user

exports.getCart = (req, res, next) => {

  req.user

**.populate('cart.items.productId')**

    .execPopulate() // wait because populate is not a promise

    .then(user => {

      const products = user.cart.items;

      res.render('shop/cart', {

        path: '/cart',

        pageTitle: 'Your Cart',

        products: products

      });

    })

    .catch(err => console.log(err));

};

We can also delete items from the cart for that we have to create delete cart function in our user model so we can call it from there

// user Schema method will create a remove cart method

userSchema.methods.removeFromCart = function(productId) {

  const updatedCartItems = this.cart.items.filter(item => {

    return item.productId.toString() !== productId.toString();

  });

  this.cart.items = updatedCartItems;

  return this.save();

};

Here we filter out the value that is equal; and also change the id style of the view cart.ejs

So after that lets create a order page and for that we have to create a order.js model to interact

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

//creating a order contains the product and user

const orderSchema = new Schema({

  products: [

    {

      product: { type: Object, required: true },

      quantity: { type: Number, required: true }

    }

  ],

  user: {

    name: {

      type: String,

      required: true

    },

    userId: {

      type: Schema.Types.ObjectId,

      required: true,

      ref: 'User'

    }

  }

});

module.exports = mongoose.model('Order', orderSchema);

and after creating it we have to define our post order method in shop controller

exports.postOrder = (req, res, next) => {

  req.user

    .populate('cart.items.productId') // populate particular

    .execPopulate()

    .then(user => {

      const products = user.cart.items.map(i => {

        // here we are mapping object contains quantity and the product object contains array of details

        return { quantity: i.quantity, product: { ...i.productId.\_doc } }; // \_doc will give whole product from id

      });

      const order = new Order({

        user: {

          name: req.user.name,

          userId: req.user

        },

        products: products // passing the product details

      });

**return order.save();**

    })

    .then(result => {

      // clearing items from cart

**return req.user.clearCart();**

    })

Now we create a clearCart item in user model so we create it

userSchema.methods.clearCart = function() {

  this.cart = { items: [] }; // it will store an empty array in items

  return this.save();

};

Now to show the order product you have to compare the id of user with order user id those who have will go further

// it will show the product

exports.getOrders = (req, res, next) => {

**Order.find({ 'user.userId': req.user.\_id })** // we will match id with our user id

    .then(orders => {

      res.render('shop/orders', {

        path: '/orders',

        pageTitle: 'Your Orders',

        orders: orders

      });

    })

    .catch(err => console.log(err));

};

And also change the order ejs file