**ERRORS**

Section:40

App.use() this is something where it runs, unnecessary of what request comes in if it’s on the top of other routes.

So if any requests comes in and we are using the app.use then we need to go to the next middleware.

**Section:42**

* When we use the next command then it tries to find the next middle ware, where as we are passing a parameter like next(err) then this means that we are transferring it to the app.use((err,req,res,next)) validator so that we can use or throw the error from it.
* JOI is a javascript validator tool.

**Section:44**

* In this comes the mongoose relationships, like the linking and corelating of each table database etc. Now in this case we are having one to one, one to few, one to many , one to bajillions
* In one to few , we can store the names and like in amazon swiggy we can have 2-3 addresses then the person may have those addresses. So in this way we can make an object of our schema and then define an array inside the addresses part for those addresses, and by using the push operation keep on pushing it inside the database.
* Now if we want to store the reference id of a table or object then we can use products: [{type:mongoose.schema.Types.ObjectId , ref: ‘ ‘}] (This whole part is while defining the schema). Now here ref is termed as the reference, this means that we are referencing to which particular element or the object.

For example we are having a farmstand as the parent ,and the products as the child. Here the point is that products will have their own table like their name, seasons , etc. so whenever we push this whole product in the farm table then in the console we see that the whole object gets stored but in the database only their Object\_Id gets stored.

* Populate means that as in the farms part everything was stored in the form of id’s then so populate will give us the descriptive representation of it showing and giving the access to edit those products.

**Section 45:**

* In this section we are dealing with the Mongo Relationships related to express. Here we are in real linking the farms and the products database.

Means to say that we are having a show , new page for Farms database, then in that page for a particular farm, we will be adding the products to a particular farm only.

* Hence we have declared a farm Schema , in which it is provided the name , city and objectId with reference from Products and in ProductSchema here , ObjectId for the farmstand.
* So now what we have done is we have stored the products in the farm databse, farm.products.push(product). And in the products page we are pushing the id of the farm, product.farm = farm as we have already found the farm. So in this we we have access to both the farms and the products from both side.
* Now comes the Middleware for deleting. This means that If we are deleting a farm then with that all the products should also be deleted. One simple way can be to findByIdAndDelete but in this what if we have multiple such arrays or entries, so another thing can be used which is mongoose middleware.

Here we are using the

farmSchema.post(‘findOneAndDelete’, async function(farm){

if(farm.products.length){

const res = await Product.deleteMany({ \_id: { $in: farm.products}})

console.log(res)

}

})

Now this is put above the model.