TYPE THE CODE

```
// /controllers/taskController.js

const Task=require('../models/taskModel.js')
const createTask=async(req,res)=> {
   try{
      const {title, completed}=req.body;
      const task=new Task({title,completed});
      const newTask=await task.save();
      res.status(201).json(newTask);
   }catch (err){
      res.status(500).json({message:err.message});
   }
};
const getAllTasks=async(req,res)=> {
   try{
      const tasks=await Task.find();
      res.status(200).json(tasks);
   }catch(err) {
      res.status(500).json({message:err.message});
   }
};
const getTaskById=async(req,res)=> {
   try{
      const {id}=req.params;
      const task=await Task.findById(req.params.id);
      if(!task){
         return res.status(404).json({message:'Task not found'});
      }
      res.status(200).json(task);
```

```
    }catch(err) {
        res.status(500).json({message:err.message});
    }
};
module.exports={createTask, getAllTasks, getTaskById};


// /models/taskModels.js

const mongoose = require('mongoose');
const taskSchema=new mongoose.Schema({
    title:{type:String, required:[true,'Task title is required']},
    completed:{type:Boolean, default:false}
});
module.exports=mongoose.model('Task', taskSchema);


// /routers/taskRoutes.js

const express=require('express');
const router=express.Router();
const {
    createTask,
    getAllTasks,
    getTaskById,
} = require('../controllers/taskController');
router.post('/tasks',createTask);
router.get('/tasks',getAllTasks);
router.get('/tasks/:id', getTaskById);
module.exports=router;
```

```
// index.js

const express=require('express');
const mongoose=require('mongoose');
const taskRoutes=require('./routers/taskRoutes');
const app=express();
const PORT=8080;
app.use(express.json());
app.use('/',taskRoutes);
mongoose.connect('mongodb://127.0.0.1:27017/appdb', {
   useNewUrlParser:true,
   useUnifiedTopology:true,
})
.then(()=> {
   console.log('Connected to MongoDB');
   app.listen(PORT, ()=> {
      console.log(`Server is running on port ${PORT}`);
   });
})
.catch((err)=> {
   console.error('MongoDB connection error:',err);
});
```

FIX THE CODE

```
// /controllers/itemController.js

const Item = require('../models/itemModel');
const createItem = async (req, res) => {
```

```
  const { name, quantity } = req.body;

  if (!name || quantity == null) {
    return res.status(400).json({ message: 'Name and quantity is required' });
  }
  try {
    const newItem = new Item({
      name,
      quantity
    });

    const savedItem = await newItem.save();
    res.status(201).json(savedItem);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};

const getAllItems = async (req, res) => {
  try {
    const items = await Item.find();
    res.status(200).json(items);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};

module.exports = {
  createItem,
  getAllItems
};
```

```
// /models/itemModel.js

const mongoose = require('mongoose');

const itemSchema = new mongoose.Schema({
  name: {
    type:String,
    required: true
  },
  quantity: {
    type: Number,
    required: true
  }
});

const Item = mongoose.model('Item', itemSchema);

module.exports = Item;



// /routers/itemRouter.js

const express = require('express');
const { createItem, getAllItems } = require('../controllers/itemController');

const router = express.Router();

router.post('/', createItem);
```

```
router.get('/', getAllItems);

module.exports = router;



// index.js

const express = require('express');
const mongoose = require('mongoose');
const itemRoutes = require('./routes/itemRoutes');

const app = express();

app.use(express.json());

mongoose.connect('mongodb://127.0.0.1:27017/itemDB', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.log(err));

app.use('/items', itemRoutes);

const port = 8080;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

PRACTICE AT HOME 1

```
// /controllers/productController.js

const Product=require('../models/productModel');
exports.createProduct=async(req,res,next) => {
   try{
      const {name,price,category,inStock}=req.body;
      const product=new Product({name,price,category,inStock});
      const savedProduct=await product.save();
      res.status(201).json(savedProduct);
   }catch(err) {
      next(err);
   }
};
exports.getAllProducts=async(req,res,next) => {
   try{
      const products=await Product.find();
      res.status(200).json(products);
   }catch(err){
      next(err);
   }
};

// /middlewares/errorHandler.js

const errorHandler=(err,req,res,next) => {
   console.error(err.stack);
   const statusCode=res.statusCode !== 200 ? res.statusCode : 500;
   res.status(statusCode).json({
      error:{
```

```
        message:err.message,
        stack:process.env.NODE_ENV==='production'? 'Stack trace hidden' : err.stack
      }
  });
};
module.exports=errorHandler;

// /models/productModel.js

const mongoose=require('mongoose');
const productSchema=new mongoose.Schema({
  name:{type:String, required:true},
  price:{type:Number, required:true},
  category:{type:String, required:true},
  inStock:{type:Boolean, default:true}
});
module.exports=mongoose.model('Product',productSchema);

// /routers/productRouter.js

const express=require('express');
const router=express.Router();
const ProductController=require('../controllers/productController');
router.post('/products',ProductController.createProduct);
router.get('/products',ProductController.getAllProducts);
module.exports=router;

// index.js

const express=require('express');
const mongoose=require('mongoose');
```

```
const productRoutes=require('./routers/productRoutes');
const errorHandler=require('./middlewares/errorHandler');
const app=express();
const PORT=8080;
app.use(express.json());
mongoose.connect('mongobd://127.0.0.1:27017/productsDB', {
   useNewUrlParser:true,
   useUnifiedTopology:true
})
.then(() => console.log('Connected to MongoDB'))
.catch((err) => console.error('MongoDB connection failed:',err));
app.use('/products',productRoutes);
app.use(errorHandler);
app.listen(PORT, () => {
   console.log(`Server is running on port ${PORT}`);
});
```

PRACTICE AT HOME 2

```
// /controllers/movieController.js

const Movie=require("../models/movieModel");
exports.createMovie=async(req,res) => {
   try{
      const movie=new Movie(req.body);
      const savedMovie=await movie.save();
      res.status(201).json(savedMovie);
   }catch(error){
      res.status(500).json({message:error.message});
   }
```

```
};
exports.getAllMovies=async(req,res) => {
   try{
      const movies=await Movie.find();
      res.status(200).json(movies);
   }catch(error) {
      res.status(500).json({message:error.message});
   }
};
exports.getMovieById=async(req,res) => {
   try{
      const movie=await Movie.findById(req.params.id);
      if(!movie){
         return res.status(404).json({message:"Movie not found"});
      }
      res.status(200).json(movie);
   }catch(error) {
      res.status(500).json({message:error.message});
   }
};
exports.updateMovie=async(req,res) => {
   try{
      const updatedMovie=await Movie.findByIdAndUpdate(req.params.id,req.body, {
         new:true,
         runValidators:true,
      });
      if(!updatedMovie){
         return res.status(404).json({message:"Movie not found"});
      }
      res.status(200).json(updatedMovie);
   }catch(error){
```

```
        res.status(500).json({message:error.message});
    }
};
exports.deleteMovie=async(req,res) => {
    try{
        const deletedMovie=await Movie.findByIdAndDelete(req.params.id);
        if(!deletedMovie){
            return res.status(404).json({message:"Movie not found"});
        }
        res.status(200).json({message:"Movie deleted successfully"});
    }catch(error){
        res.status(500).json({message:error.message});
    }
};

// /models/movieModel.js

const mongoose=require("mongoose");
const movieSchema=new mongoose.Schema({
    title: {
        type:String,
        required:[true, "Movie title is requied"],
    },
    director: {
        type:String,
        required:[true,"Director is required"],
    },
    releaseYear: {
        type:Number,
        required:[true,"Release year is required"],
    },
```

```
    genre: {
        type:String,
        required:[true,"Genre is required"],
    },
});
const Movie=mongoose.model("Movie",movieSchema);
module.exports=Movie;

// routers/movieRoutes.js

const express=require("express");
const router=express.Router();
const {
    getMovieById,
    updateMovieById,
    deleteMovieById,
} = require("../controllers/movieController");
router.get("/movies/:id",getMovieById);
router.put("/movies/:id",updateMovieById);
router.delete("/movies/:id",deleteMovieById);
module.exports=router;

//index.js

const express=require("express");
const mongoose=require("mongoose");
const movieRoutes=require("./routers/movieRoutes");
const app=express();
app.use(express.json());
mongoose
  .connect("mongodb://localhost:27017/movies",{
```

```
    useNewUrlParser:true,
    useUnifiedTopology:true,
  })
  .then(() => console.log("MongoDB connected"))
  .catch((err) => console.log(err));
app.use("/",movieRoutes);
app.use((err,req,res,next) => {
    console.error(err.stack);
    res.status(500).json({error:"Something went wrong!"});
});
const PORT=process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});
```

CHALLENGE YOURSELF

```
// /controllers/customerController.js

const Customer=require('../models/customerModel');
exports.createCustomer=async(req,res,next) => {
    try{
        const {name,email,phone,isActive}=req.body;
        if(!name || !email || !phone){
            return res.status(404).json({message:'Name, email, and phone are required'});
        }
        const newCustomer=new Customer({name,email,phone,isActive});
        const savedCustomer=await newCustomer.save();
        res.status(201).json(savedCustomer);
    }catch(err){
```

```
        next(err);
    }
};
exports.getAllCustomers=async(req,res,next) => {
    try{
        const customers=await Customer.find();
        res.status(200).json(customers);
    }catch(err) {
        next(err);
    }
};
exports.updateCustomer=async(req,res,next) => {
    try{
        const {id}=req.params;
        const {name,email,phone,isActive}=req.body;
        if(!name || !email || !phone) {
            return res.status(404).json({message: 'Name, email, and phone are required to update the
customer'});
        }
        const updated=await Customer.findByIdAndUpdate(
            id,
            {name,email,phone,isActive},
            {new:true}
        );
        if(!updated){
            return res.status(404).json({message:'Customer not found'});
        }
        res.status(200).json(updated);
    }catch(err){
        next(err);
    }
```

```
};

// /middlewares/errorHandler.js

const errorHandler=(err,req,res,next) => {
   console.error(err.stack);
   const status=res.statusCode !== 200 ? res.statusCode : 500;
   res.status(status).json({
      message:err.message,
      ...(process.env.NODE_ENV === 'development' && {stack:err.stack})
   });
};
module.exports=errorHandler;

// /models/customerModel.js

const mongoose=require('mongoose');
const customerSchema=new mongoose.Schema({
   name:{type:String,required:true},
   email:{type:String,required:true},
   phone:{type:String,required:true},
   isActive:{type:Boolean,default:true}
});
module.exports=mongoose.model('Customer',customerSchema);

// /routers/customerRoutes.js

const express=require('express');
const router=express.Router();
const customerController=require('../controllers/customerController');
router.post('/customers',customerController.createCustomer);
```

```
router.get('/customers',customerController.getAllCustomers);
router.put('/customers/:id',customerController.updateCustomer);
module.exports=router;

// index.js

const express=require('express');
const mongoose=require('mongoose');
const customerRoutes=require('./routers/customerRoutes');
const errorHandler=require('./middlewares/errorHandler');
const app=express();
const PORT=8080;
app.use(express.json());
mongoose.connect('mongodb://127.0.0.1:27017/customersDB', {
   useNewUrlParser:true,
   useUnifiedTopology:true
})
.then(() => console.log('MongoDB connected'))
.catch(err => console.error('MongoDB connection error:',err));
app.use('/customers',customerRoutes);
app.use(errorHandler);
app.listen(PORT, () => {
   console.log(`Server running on http://localhost:${PORT}`);
});
```