**🔧 SafeShipping API Backend - Node.js & Express**

This sets up a **RESTful API** for shipment records, including **blockchain-backed smart contracts**.

**📂 Directory Structure**

/safeshipping-api

├── /config

│ ├── db.js

│ ├── blockchain.js

├── /models

│ ├── Shipment.js

├── /routes

│ ├── shipments.js

├── /controllers

│ ├── shipmentController.js

├── server.js

├── package.json

**🚀 Backend Code (Node.js, Express)**

**1️⃣ Initialize the API (server.js)**

javascript

const express = require("express");

const shipmentsRoute = require("./routes/shipments");

const app = express();

app.use(express.json());

app.use("/api/shipments", shipmentsRoute);

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`🚀 SafeShipping API running on port ${PORT}`);

});

**2️⃣ Shipment Model (models/Shipment.js)**

javascript

const mongoose = require("mongoose");

const ShipmentSchema = new mongoose.Schema({

shipmentId: { type: String, required: true, unique: true },

carrier: { type: String, required: true },

transportMode: { type: String, enum: ["Air", "Sea", "Land", "Rail"], required: true },

origin: { type: String, required: true },

destination: { type: String, required: true },

status: { type: String, enum: ["Pending", "In Transit", "Delivered"], required: true },

estimatedArrival: { type: Date },

actualArrival: { type: Date },

blockchainContractHash: { type: String }, // Smart contract reference

IoTData: { type: Map, of: String } // IoT sensor logs

});

module.exports = mongoose.model("Shipment", ShipmentSchema);

**3️⃣ Shipment Routes (routes/shipments.js)**

javascript

const express = require("express");

const { createShipment, updateShipment, getShipments } = require("../controllers/shipmentController");

const router = express.Router();

router.post("/", createShipment);

router.patch("/:shipmentId", updateShipment);

router.get("/", getShipments);

module.exports = router;

**4️⃣ Shipment Controller (controllers/shipmentController.js)**

javascript

const Shipment = require("../models/Shipment");

exports.createShipment = async (req, res) => {

try {

const shipment = new Shipment(req.body);

await shipment.save();

res.status(201).json(shipment);

} catch (error) {

res.status(500).json({ error: error.message });

}

};

exports.updateShipment = async (req, res) => {

try {

const shipment = await Shipment.findOneAndUpdate(

{ shipmentId: req.params.shipmentId },

req.body,

{ new: true }

);

res.status(200).json(shipment);

} catch (error) {

res.status(500).json({ error: error.message });

}

};

exports.getShipments = async (req, res) => {

try {

const shipments = await Shipment.find();

res.status(200).json(shipments);

} catch (error) {

res.status(500).json({ error: error.message });

}

};

**📜 Sample JSON Payload - Shipper Contract**

This represents **blockchain-backed shipping records**, including **IoT tracking fields**.

json

{

"shipmentId": "SHIP-123456",

"carrier": "Maersk",

"transportMode": "Sea",

"origin": "Shanghai",

"destination": "Los Angeles",

"status": "In Transit",

"estimatedArrival": "2025-06-20T15:00:00Z",

"blockchainContractHash": "0xabcdef123456",

"IoTData": {

"GPS": "32.7157,-117.1611",

"Temperature": "12°C",

"Humidity": "60%",

"ShockSensors": "No Impact Detected",

"SealIntegrity": "Secure",

"FuelUsage": "85 liters/hour"

}

}

**🚀 Key Features in This Setup**

✔ **RESTful API with Node.js & Express** ✔ **MongoDB schema for shipment records** ✔ **Smart contract hash reference for blockchain verification** ✔ **IoT data tracking (GPS, temperature, seal integrity, fuel usage, etc.)** ✔ **Extensible architecture for multi-modal transportation records**