const express = require('express');

const app = express();

const mongoose = require('mongoose');

const packageModel = require('./packageModel');

app.use(express.json());

// Connect to MongoDB

mongoose.connect('mongodb:

  .then(() => {

    console.log('//localhost:27017/packages', { useNewUrlParser: true, useUnifiedTopology: true })

  .then(() => {

    console.log('Connected to MongoDB');

  })

  .catch((error) => {

    console.error('Error connecting to MongoDB:', error);

  });

const packageSchema = new mongoose.Schema({

  name: String,

  minInvestment: Number,

  maxInvestment: Number,

  dailyProfitPercentage: Number,

  duration: Number

});

const Package = mongoose.model('// Package schema

const packageSchema = new mongoose.Schema({

  name: String,

  minInvestment: Number,

  maxInvestment: Number,

  dailyProfitPercentage: Number,

  duration: Number

});

// Package model

const Package = mongoose.model('Package', packageSchema);

[app.post](http://app.post/)('// Create package

[app.post](http://app.post/)('/packages', async (req, res) => {

  try {

    const packageData = req.body;

    const packageDoc = new Package(packageData);

    await packageDoc.save();

    res.json(packageDoc);

  } catch (error) {

    res.status(500).json({ message: 'Failed to create package' });

  }

});

app.get('// Get all packages

app.get('/packages', async (req, res) => {

  try {

    const packages = await Package.find().exec();

    res.json(packages);

  } catch (error) {

    res.status(500).json({ message: 'Failed to retrieve packages' });

  }

});

app.get('// Get package

app.get('/packages/:packageId', async (req, res) => {

  try {

    const packageId = req.params.packageId;

    const packageDoc = await Package.findById(packageId).exec();

    if (!packageDoc) {

      res.status(404).json({ message: 'Package not found' });

    } else {

      res.json(packageDoc);

    }

  } catch (error) {

    res.status(500).json({ message: 'Failed to retrieve package' });

  }

});

app.put('// Update package

app.put('/packages/:packageId', async (req, res) => {

  try {

    const packageId = req.params.packageId;

    const packageData = req.body;

    const packageDoc = await Package.findByIdAndUpdate(packageId, packageData, { new: true });

    if (!packageDoc) {

      res.status(404).json({ message: 'Package not found' });

    } else {

      res.json(packageDoc);

    }

  } catch (error) {

    res.status(500).json({ message: 'Failed to update package' });

  }

});

app.delete('// Delete package

app.delete('/packages/:packageId', async (req, res) => {

  try {

    const packageId = req.params.packageId;

    await Package.findByIdAndRemove(packageId);

    res.json({ message: 'Package deleted successfully' });

  } catch (error) {

    res.status(500).json({ message: 'Failed to delete package' });

  }

});

const predefinedPackages = [

  {

    name: '// Predefined packages

const predefinedPackages = [

  {

    name: 'Beginners',

    minInvestment: 10,

    maxInvestment: 200,

    dailyProfitPercentage: 7.5,

    duration: 7

  },

  {

    name: 'Intermediate',

    minInvestment: 200,

    maxInvestment: 500,

    dailyProfitPercentage: 13.5,

    duration: 30

  },

  {

    name: 'Advance',

    minInvestment: 500,

    maxInvestment: 3000,

    dailyProfitPercentage: 16,

    duration: 30

  },

  {

    name: 'Long-term',

    minInvestment: 3000,

    maxInvestment: 5000,

    dailyProfitPercentage: 21,

    duration: 30

  },

  {

    name: 'Core Focus',

    minInvestment: 5000,

    maxInvestment: 20000,

    dailyProfitPercentage: 30,

    duration: 30

  },

  {

    name: 'Go All',

    minInvestment: 20000,

    maxInvestment: Infinity,

    dailyProfitPercentage: 35,

    duration: 30

  }

];

[app.post](http://app.post/)('// Initialize predefined packages

[app.post](http://app.post/)('/init-packages', async (req, res) => {

  try {

    await Package.deleteMany();

    await Package.insertMany(predefinedPackages);

    res.json({ message: 'Predefined packages initialized successfully' });

  } catch (error) {

    res.status(500).json({ message: 'Failed to initialize predefined packages' });

  }

});

app.listen(3000, () => {

  console.log('Server listening on port 3000');

});