**✅ Updated config/ Directory Tree**

src/

└── config/

├── app.config.ts

├── auth.config.ts

├── cache.config.ts

├── mail.config.ts

├── sms-gateway.config.ts

├── notification.config.ts

├── file.config.ts

├── kafka.config.ts

├── queue.config.ts

├── rate-limit.config.ts

├── websocket.config.ts

├── external-api.config.ts

└── config.module.ts

Absolutely — let me now provide a **fully updated config/ directory structure** with **detailed inline comments** for every file. Each config file will explain:

* 🧩 What it does
* 💬 Why it exists
* 🛠️ How it is used
* 🔐 Which env variables are needed
* 📘 Realistic use-case comments

**✅ Updated config/ Directory with Detailed Comments**

**📁 config/app.config.ts**

// Application-wide configuration (port, app name, env, etc.)

export default () => ({

name: process.env.APP\_NAME || 'MyAwesomeApp', // Name of the application

environment: process.env.NODE\_ENV || 'development', // 'development' | 'production' | 'test'

port: parseInt(process.env.APP\_PORT, 10) || 3000, // The port where app will run

});

**📁 config/auth.config.ts**

// Configuration for JWT Auth system (used in guards & strategies)

export default () => ({

jwt: {

secret: process.env.JWT\_SECRET, // Secret used to sign JWT tokens

expiresIn: process.env.JWT\_EXPIRES\_IN || '1h' // Token validity (e.g., '1h', '2d')

},

});

**📁 config/cache.config.ts**

// Redis Cache settings for session store, rate-limiting, or caching

export default () => ({

redis: {

host: process.env.REDIS\_HOST || 'localhost', // Redis server address

port: parseInt(process.env.REDIS\_PORT, 10) || 6379, // Redis port

ttl: parseInt(process.env.CACHE\_TTL, 10) || 300, // Default cache TTL in seconds

},

});

**📁 config/mail.config.ts**

// SMTP configuration for sending system emails (password reset, OTP, etc.)

export default () => ({

smtp: {

host: process.env.MAIL\_HOST, // SMTP host (e.g., smtp.mailtrap.io)

port: parseInt(process.env.MAIL\_PORT, 10), // SMTP port

user: process.env.MAIL\_USER, // SMTP username

pass: process.env.MAIL\_PASS, // SMTP password

fromEmail: process.env.MAIL\_FROM, // Default "from" email address

},

});

**📁 config/sms-gateway.config.ts**

// Configuration for sending SMS via providers (e.g., Twilio)

export default () => ({

provider: process.env.SMS\_PROVIDER || 'twilio', // Can be 'twilio', 'nexmo', etc.

twilio: {

sid: process.env.TWILIO\_SID, // Twilio SID

authToken: process.env.TWILIO\_AUTH\_TOKEN, // Twilio Auth Token

fromNumber: process.env.TWILIO\_FROM, // Registered Twilio number

},

});

**📁 config/notification.config.ts**

// Configuration for Push Notifications (FCM, OneSignal, etc.)

export default () => ({

fcm: {

apiKey: process.env.FCM\_API\_KEY, // Firebase Cloud Messaging API Key

},

onesignal: {

appId: process.env.ONESIGNAL\_APP\_ID, // OneSignal App ID

apiKey: process.env.ONESIGNAL\_API\_KEY, // OneSignal API Key

},

});

**📁 config/file.config.ts**

// File Upload/Storage configuration for local or cloud (e.g., AWS S3)

export default () => ({

upload: {

maxSizeMb: parseInt(process.env.FILE\_MAX\_SIZE\_MB, 10) || 5, // Max file size in MB

localPath: process.env.FILE\_UPLOAD\_PATH || './uploads', // Local directory for file storage

useS3: process.env.FILE\_USE\_S3 === 'true', // Use AWS S3 if true

},

s3: {

accessKey: process.env.S3\_ACCESS\_KEY, // AWS Access Key

secretKey: process.env.S3\_SECRET\_KEY, // AWS Secret Key

bucket: process.env.S3\_BUCKET\_NAME, // AWS Bucket Name

region: process.env.S3\_REGION, // AWS Region

},

});

**📁 config/kafka.config.ts**

// Kafka configuration for microservice event streaming

export default () => ({

brokers: (process.env.KAFKA\_BROKERS || '').split(','), // Comma-separated Kafka broker addresses

clientId: process.env.KAFKA\_CLIENT\_ID || 'my-service', // Kafka client identifier

});

**📁 config/queue.config.ts**

// Queue management settings for BullMQ or RabbitMQ

export default () => ({

type: process.env.QUEUE\_TYPE || 'bullmq', // Queue engine type

bullmq: {

connection: {

host: process.env.BULLMQ\_REDIS\_HOST, // Redis for BullMQ

port: parseInt(process.env.BULLMQ\_REDIS\_PORT, 10),

},

},

rabbitmq: {

uri: process.env.RABBITMQ\_URI, // Full AMQP URI for RabbitMQ

},

});

**📁 config/rate-limit.config.ts**

// Global Rate Limiting and Throttling options

export default () => ({

ttl: parseInt(process.env.RATE\_LIMIT\_TTL, 10) || 60, // Time to live (seconds)

limit: parseInt(process.env.RATE\_LIMIT\_MAX, 10) || 20, // Max requests per TTL

});

**📁 config/websocket.config.ts**

// WebSocket real-time communication settings

export default () => ({

port: parseInt(process.env.WS\_PORT, 10) || 3001, // WebSocket server port

enable: process.env.ENABLE\_WS === 'true', // Toggle real-time features

});

**📁 config/external-api.config.ts**

// Configs for 3rd-party APIs (e.g., Stripe, OpenAI, etc.)

export default () => ({

stripe: {

secretKey: process.env.STRIPE\_SECRET\_KEY, // Stripe secret key

publicKey: process.env.STRIPE\_PUBLIC\_KEY, // Stripe publishable key

},

openai: {

apiKey: process.env.OPENAI\_API\_KEY, // OpenAI access key

},

});

**📁 config/config.module.ts**

// Main module to globally load and expose all configs using @nestjs/config

import { Module } from '@nestjs/common';

import { ConfigModule as NestConfigModule } from '@nestjs/config';

import appConfig from './app.config';

import authConfig from './auth.config';

import cacheConfig from './cache.config';

import mailConfig from './mail.config';

import smsGatewayConfig from './sms-gateway.config';

import notificationConfig from './notification.config';

import fileConfig from './file.config';

import kafkaConfig from './kafka.config';

import queueConfig from './queue.config';

import rateLimitConfig from './rate-limit.config';

import websocketConfig from './websocket.config';

import externalApiConfig from './external-api.config';

@Module({

imports: [

NestConfigModule.forRoot({

isGlobal: true, // Makes ConfigService available app-wide

envFilePath: [`.env.${process.env.NODE\_ENV || 'development'}`, '.env'], // Loads environment files in order

load: [

appConfig,

authConfig,

cacheConfig,

mailConfig,

smsGatewayConfig,

notificationConfig,

fileConfig,

kafkaConfig,

queueConfig,

rateLimitConfig,

websocketConfig,

externalApiConfig,

],

}),

],

})

export class ConfigModule {}

**📘 Usage Example Inside a Service**

import { ConfigService } from '@nestjs/config';

import { Injectable } from '@nestjs/common';

@Injectable()

export class NotificationService {

constructor(private readonly configService: ConfigService) {}

sendPushNotification(message: string) {

const fcmKey = this.configService.get<string>('notification.fcm.apiKey');

const onesignalKey = this.configService.get<string>('notification.onesignal.apiKey');

// ...use those keys to send notification

}

}