**1. package.json**

The package.json file is crucial in any Node.js project. It acts as the metadata and dependency manager for your NestJS project. It contains information about the project, its dependencies, and scripts for running the project.

**Example of package.json**

json

CopyEdit

{

"name": "nest-js-project",

"version": "1.0.0",

"description": "A NestJS project",

"main": "dist/main.js",

"scripts": {

"start": "node dist/main.js",

"start:dev": "nest start --watch",

"build": "nest build",

"test": "jest",

"lint": "eslint . --ext .ts",

"format": "prettier --write ."

},

"dependencies": {

"@nestjs/common": "^8.0.0",

"@nestjs/core": "^8.0.0",

"@nestjs/platform-express": "^8.0.0",

"rxjs": "^7.0.0",

"reflect-metadata": "^0.1.13"

},

"devDependencies": {

"@nestjs/cli": "^8.0.0",

"eslint": "^7.32.0",

"prettier": "^2.3.2",

"jest": "^27.0.0",

"ts-jest": "^27.0.0",

"typescript": "^4.4.0"

}

}

**Explanation:**

* **name**: The name of your project.
* **version**: The current version of your project.
* **description**: A short description of what your project does.
* **main**: Points to the entry file (i.e., the main entry file for your app after build).
* **scripts**: The scripts section includes commands you can run using npm run <script-name>. This allows you to run common tasks with simple commands:
  + start: Starts the application in production mode.
  + start:dev: Starts the application in development mode and watches for file changes.
  + build: Builds the NestJS application (compiles it to the dist folder).
  + test: Runs the Jest tests.
  + lint: Runs ESLint to check for code quality and style violations.
  + format: Runs Prettier to auto-format the code.
* **dependencies**: These are the runtime dependencies that your application needs to work properly (e.g., @nestjs/common, rxjs).
* **devDependencies**: These are dependencies required only for development, such as testing tools (jest, ts-jest) and code quality tools (eslint, prettier).

**Key Points:**

* This file helps in setting up project-wide configurations and dependencies.
* When you add packages (e.g., via npm install or yarn add), they will appear in the dependencies or devDependencies section.
* The scripts section allows automation of common tasks (build, test, lint).