app.js

const express = require("express");

const app = express();

const cors = require("cors"); // وارد کردن پکیج CORS

const tasks = require("./routes/tasks");

const connectDB = require("./db/connect");

require("dotenv").config();

const notFound = require("./middleware/not-found");

const errorHandlerMiddleware = require("./middleware/error-handler");

**توضیح:**

1. **const express = require("express");** 
   * این خط پکیج **express** را به پروژه شما وارد می‌کند. Express یک فریمورک برای ساخت برنامه‌های وب و API در **Node.js** است.
   * **مثال:** فرض کنید شما می‌خواهید یک وب‌سایت بسازید که صفحه اصلی را نشان دهد، برای این کار باید از Express استفاده کنید.
2. **const app = express();** 
   * این خط یک **برنامه Express** جدید به نام app ایجاد می‌کند که برای مدیریت درخواست‌ها و پاسخ‌ها استفاده می‌شود.
   * **مثال:** وقتی یک درخواست به سرور شما می‌آید، app مسئول پردازش آن درخواست است.
3. **const cors = require("cors");** 
   * این خط پکیج **CORS** را وارد می‌کند. CORS (Cross-Origin Resource Sharing) به شما این امکان را می‌دهد که درخواست‌ها از منابع مختلف (domain‌ها یا پورت‌های مختلف) را قبول کنید.
   * **مثال:** اگر API شما روی پورت 3000 در حال اجراست و فایل‌های استاتیک شما روی پورت 5500 هستند، برای جلوگیری از مشکلات امنیتی هنگام دسترسی به داده‌ها، از CORS استفاده می‌شود.
4. **const tasks = require("./routes/tasks");** 
   * این خط یک فایل دیگر به نام tasks.js را از پوشه routes وارد می‌کند. این فایل شامل **مسیرهای مربوط به درخواست‌های tasks** است.
   * **مثال:** فرض کنید شما یک API برای مدیریت تسک‌ها دارید. این خط مربوط به مسیرهایی است که در فایل tasks.js تنظیم شده است.
5. **const connectDB = require("./db/connect");** 
   * این خط یک فایل connect.js را از پوشه db وارد می‌کند که وظیفه اتصال به پایگاه داده MongoDB را دارد.
   * **مثال:** این کد باعث می‌شود که سرور شما به پایگاه داده MongoDB وصل شود تا داده‌ها ذخیره و بازیابی شوند.
6. **require("dotenv").config();** 
   * این خط پکیج **dotenv** را وارد می‌کند که به شما این امکان را می‌دهد تا متغیرهای محیطی (environment variables) مانند MONGO\_URI را از یک فایل .env بارگذاری کنید.
   * **مثال:** به جای سخت‌کدنویسی اطلاعات حساس مانند اطلاعات اتصال به پایگاه داده در کد، این متغیرها را در یک فایل .env قرار می‌دهید.
7. **const notFound = require("./middleware/not-found");** 
   * این خط فایل not-found.js را از پوشه middleware وارد می‌کند که مسئول مدیریت درخواست‌هایی است که به مسیرهای ناموجود می‌روند.
   * **مثال:** اگر کاربر به مسیری درخواست دهد که وجود ندارد، این فایل مسئول ارسال پیغام خطای "مسیر یافت نشد" است.
8. **const errorHandlerMiddleware = require("./middleware/error-handler");** 
   * این خط فایل error-handler.js را وارد می‌کند که برای مدیریت خطاها در درخواست‌ها استفاده می‌شود.
   * **مثال:** اگر در حین اجرای یک درخواست خطایی پیش بیاید (مثل اتصال به پایگاه داده)، این فایل مسئول مدیریت و ارسال پاسخ مناسب به کاربر است.

app.use(express.static("./public"));

app.use(express.json());

**توضیح:**

1. **app.use(express.static("./public"));** 
   * این خط به Express می‌گوید که تمام فایل‌های استاتیک (مثل HTML، CSS، JS، تصاویر) از پوشه public ارائه شوند.
   * **مثال:** اگر شما یک فایل index.html در پوشه public داشته باشید، این خط باعث می‌شود که کاربر بتواند آن را در مرورگر مشاهده کند.
2. **app.use(express.json());** 
   * این خط به Express می‌گوید که تمام درخواست‌ها را به فرمت JSON تجزیه کند. این برای زمانی مفید است که شما بخواهید داده‌های دریافتی از کاربر را در قالب JSON ارسال و دریافت کنید.
   * **مثال:** اگر کاربر یک درخواست POST با محتوای JSON ارسال کند، این خط باعث می‌شود که آن داده‌ها به طور خودکار به صورت یک شیء JavaScript در دسترس قرار بگیرند.

const corsOptions = {

origin: 'http://127.0.0.1:5500', // یا آدرس دیگری که فایل‌های استاتیک شما در آن قرار دارند

methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH'], // متدهای مجاز

allowedHeaders: ['Content-Type', 'Authorization'], // هدرهای مجاز

};

**توضیح:**

* در اینجا شما تنظیمات **CORS** را مشخص کرده‌اید. یعنی فقط از دامنه خاصی (در اینجا http://127.0.0.1:5500) اجازه دسترسی به API شما را می‌دهید.
* methods متدهای HTTP مجاز را مشخص می‌کند (مثل GET، POST و...).
* allowedHeaders هدرهایی که می‌توانید در درخواست‌ها استفاده کنید را تعیین می‌کند.

app.use(cors(corsOptions));

* این خط باعث می‌شود که **CORS** با تنظیماتی که در corsOptions مشخص کرده‌اید، فعال شود.

app.options('\*', cors(corsOptions)); // این باعث می‌شود که سرور به درخواست‌های OPTIONS پاسخ دهد

این خط به سرور شما می‌گوید که به درخواست‌های **OPTIONS** (که معمولاً قبل از ارسال درخواست‌های اصلی برای بررسی CORS ارسال می‌شود) پاسخ دهد.

### مسیرها (Routes):

app.use("/api/v1/tasks", tasks);

#### توضیح:

* این خط می‌گوید که تمام درخواست‌هایی که با مسیر /api/v1/tasks شروع می‌شوند، باید به فایل tasks.js در پوشه routes ارسال شوند.
* **مثال:** اگر کاربر به /api/v1/tasks درخواست GET ارسال کند، کد داخل فایل tasks.js اجرا می‌شود.

### مدیریت خطاها:

app.use(notFound);

app.use(errorHandlerMiddleware);

#### توضیح:

* notFound: این خط مسئول هدایت درخواست‌های ناموجود به یک پیغام "مسیر یافت نشد" است.
* errorHandlerMiddleware: این خط برای مدیریت خطاها استفاده می‌شود. به عنوان مثال، اگر در طول اجرای کد خطایی پیش بیاید، این میانه‌افزار مسئول ارسال پیام خطا به کاربر است.

### راه‌اندازی سرور:

const port = process.env.PORT || 3000;

#### توضیح:

* این خط می‌گوید که پورت سرور از متغیر محیطی PORT گرفته شود، اگر این متغیر وجود نداشته باشد، از پورت 3000 استفاده می‌شود.

const start = async () => {

try {

await connectDB(process.env.MONGO\_URI);

app.listen(port, () =>

console.log(`Server is listening on port ${port}...`)

);

} catch (error) {

console.log(error);

}

};

start();

#### توضیح:

* **start** یک تابع است که ابتدا به پایگاه داده MongoDB وصل می‌شود (با استفاده از connectDB)، سپس سرور را در پورت مشخص‌شده اجرا می‌کند.
* **await connectDB(process.env.MONGO\_URI)**: اتصال به پایگاه داده MongoDB با استفاده از URI موجود در فایل .env.
* **app.listen(port)**: سرور Express را روی پورت مشخص‌شده راه‌اندازی می‌کند.

این کد برای ایجاد یک API ساده است که به پایگاه داده متصل می‌شود و درخواست‌ها را مدیریت می‌کند.