Node.js-এ ফাইল আপলোডের একটি বিস্তারিত নির্দেশিকা: Multer-এর আদ্যোপান্ত ব্যাখ্যা

ভূমিকা: ওয়েব অ্যাপ্লিকেশনে ফাইল আপলোড এবং Multer-এর প্রাসঙ্গিকতা

আধুনিক ওয়েব অ্যাপ্লিকেশনগুলোতে ফাইল আপলোডের সুবিধা একটি অপরিহার্য বৈশিষ্ট্য। ব্যবহারকারীর প্রোফাইল ছবি আপলোড করা থেকে শুরু করে বিভিন্ন ডকুমেন্ট, ভিডিও বা অন্যান্য বাইনারি ফাইল সার্ভারে পাঠানোর প্রয়োজনীয়তা প্রায়শই দেখা যায়। এই প্রক্রিয়াটি আপাতদৃষ্টিতে সহজ মনে হলেও, এটি প্রযুক্তিগতভাবে একটি জটিল কাজ। এর মূল কারণ হলো ওয়েব ফর্ম ডেটা সাধারণত application/x-www-form-urlencoded বা application/json ফরম্যাটে পাঠানো হয়, যা বাইনারি ডেটা যেমন ফাইল হ্যান্ডেল করার জন্য উপযুক্ত নয়। এই চ্যালেঞ্জ মোকাবিলায় multipart/form-data নামক একটি বিশেষ এনকোডিং টাইপ ব্যবহার করা হয় [1]।

multipart/form-data এনকোডিংয়ের মাধ্যমে একটি একক HTTP রিকোয়েস্টের মধ্যে টেক্সট ডেটা এবং বাইনারি ডেটা উভয়ই পাঠানো সম্ভব হয়। প্রতিটি ডেটার অংশকে একটি নির্দিষ্ট "বাউন্ডারি" দ্বারা আলাদা করা হয় এবং প্রতিটি অংশের নিজস্ব Content-Type এবং Content-Disposition থাকে। এই প্রক্রিয়াটি ব্রাউজার দ্বারা স্বয়ংক্রিয়ভাবে পরিচালিত হয় যখন HTML ফর্মে enctype="multipart/form-data" অ্যাট্রিবিউটটি ব্যবহার করা হয় [1]। তবে, ক্লায়েন্ট থেকে এই জটিল ফরম্যাটে আসা ডেটা সার্ভারে গ্রহণ এবং পার্স করার জন্য একটি বিশেষ টুলের প্রয়োজন হয়। এখানেই Node.js ইকোসিস্টেমে Multer-এর আগমন হয়। Multer একটি শক্তিশালী মিডলওয়্যার যা এই

multipart/form-data রিকোয়েস্টগুলো দক্ষতার সাথে হ্যান্ডেল করার মাধ্যমে ডেভেলপারদের জন্য ফাইল আপলোডের প্রক্রিয়াকে অত্যন্ত সহজ করে তোলে [4]।

Multer কী এবং এটি কীভাবে কাজ করে?

Multer হলো Express.js-এর জন্য তৈরি একটি জনপ্রিয় Node.js মিডলওয়্যার প্যাকেজ যা বিশেষভাবে multipart/form-data হ্যান্ডেল করার জন্য ডিজাইন করা হয়েছে [4]। এটি মূলত ফাইল আপলোডের

প্রক্রিয়াকে সহজ করার জন্য ব্যবহৃত হয়। Multer একা কাজ করে না; এর কার্যকারিতার পেছনে রয়েছে

busboy নামক একটি লাইব্রেরি, যার উপরে এটি নির্মিত। busboy অত্যন্ত দক্ষতার সাথে স্ট্রিম থেকে multipart/form-data পার্স করে এবং Multer সেই পার্সিংয়ের ফলাফলকে Express.js-এর রিকোয়েস্ট অবজেক্টের সাথে যুক্ত করে [4]।

Multer-এর মিডলওয়্যার হিসেবে কাজ করার মূল সুবিধা হলো এটি রিকোয়েস্ট-রেসপন্স সাইকেলে একটি গুরুত্বপূর্ণ স্তরে যুক্ত হয়। যখন কোনো ক্লায়েন্ট একটি multipart ফর্ম ডেটা সম্বলিত POST রিকোয়েস্ট পাঠায়, Multer সেই রিকোয়েস্টটি গ্রহণ করে এবং তার মধ্যে থাকা ফাইল এবং টেক্সট ফিল্ডগুলোকে আলাদাভাবে প্রক্রিয়াকরণ করে। এই প্রক্রিয়াকরণের পর, Multer দুটি নতুন অবজেক্টকে রিকোয়েস্ট (req) অবজেক্টে যোগ করে: req.body এবং req.file (অথবা req.files)। req.body অবজেক্টে ফর্মের সমস্ত টেক্সট ফিল্ডের মানগুলো থাকে, আর req.file বা req.files অবজেক্টে আপলোড করা ফাইলের সমস্ত তথ্য পাওয়া যায় [5]। এই কাঠামোগত ব্যবস্থা ডেভেলপারদের জন্য ফাইল এবং ডেটা অ্যাক্সেস করা অত্যন্ত সহজ করে তোলে।

Multer-এর কার্যকারিতা তার নির্ভরশীলতা (dependencies) লাইব্রেরিগুলোর উপর কতটা নির্ভরশীল, তা একটি সাম্প্রতিক দুর্বলতার ঘটনা থেকে স্পষ্ট হয়ে উঠেছে। Multer-এর সংস্করণ 1.4.4-lts.1 থেকে 2.0.2 পর্যন্ত একটি গুরুতর DoS (Denial of Service) দুর্বলতা (CVE-2025-7338) ছিল [5]। এই দুর্বলতার মূল কারণ ছিল

busboy লাইব্রেরির ত্রুটি হ্যান্ডলিংয়ে সমস্যা। যখন একটি বিকৃত (malformed) রিকোয়েস্ট (যেমন একটি খালি name অ্যাট্রিবিউট সহ ফর্ম ফিল্ড) পাঠানো হতো, busboy একটি ত্রুটি (error event) তৈরি করত। Multer-এর পুরোনো সংস্করণগুলোতে এই ত্রুটি ধরার কোনো ব্যবস্থা ছিল না, যার ফলে এটি একটি uncaught exception হিসেবে বিবেচিত হতো। যেহেতু Node.js একটি সিঙ্গেল-থ্রেডেড ইভেন্ট লুপে চলে, এই ধরনের একটি অপ্রত্যাশিত ত্রুটি পুরো অ্যাপ্লিকেশনটিকে তাৎক্ষণিকভাবে ক্র্যাশ করে দিত [5]। এই ঘটনাটি প্রমাণ করে যে, কোনো লাইব্রেরি ব্যবহার করার সময় শুধু তার নিজের নিরাপত্তা নয়, বরং তার নির্ভরশীল লাইব্রেরিগুলোর নিরাপত্তা সংক্রান্ত দিকগুলো সম্পর্কেও সচেতন থাকা জরুরি। এই বিশেষ DoS সমস্যাটি Multer v2.0.2-তে প্যাচ করা হয়েছে, যা ডেভেলপারদের জন্য সর্বদা লাইব্রেরির সর্বশেষ স্থিতিশীল সংস্করণ ব্যবহার করার গুরুত্ব তুলে ধরে [5]।

Multer-এর কনফিগারেশন এবং ফাইল আপলোডের বিভিন্ন পদ্ধতি

Multer ব্যবহার করার জন্য প্রথমে এটি Node Package Manager (npm) এর মাধ্যমে ইনস্টল করে নিতে হয়: $ npm install multer [6]। এরপর

multer() ফাংশন ব্যবহার করে একটি ইনস্ট্যান্স তৈরি করতে হয়, যেখানে একটি অপশনস অবজেক্ট পাস করা যায়। এই অবজেক্টটি Multer-এর কার্যকারিতা নিয়ন্ত্রণ করে [5]। সবচেয়ে সাধারণ অপশনগুলো হলো:

- **dest**: এই অপশনটি একটি স্ট্রিং যা আপলোড করা ফাইল সংরক্ষণের গন্তব্য ডিরেক্টরি নির্দেশ করে। যদি এই ডিরেক্টরি বিদ্যমান না থাকে, তবে Multer স্বয়ংক্রিয়ভাবে তা তৈরি করে। এটি ফাইল আপলোডের সবচেয়ে সহজ পদ্ধতি, যেখানে Multer স্বয়ংক্রিয়ভাবে ফাইলগুলোর নাম পরিবর্তন করে

যাতে কোনো নাম সংক্রান্ত দ্বন্দ্ব এড়ানো যায় [5]।

- **storage**: যদি আপলোডের উপর আরো বেশি নিয়ন্ত্রণ প্রয়োজন হয়, তবে dest-এর পরিবর্তে এই অপশনটি ব্যবহার করা হয়। এটি একটি স্টোরেজ ইঞ্জিন অবজেক্ট নেয়, যা DiskStorage বা MemoryStorage হতে পারে [6]।
- **limits**: এটি আপলোড করা ফাইলের আকার, ফাইলের সংখ্যা, বা ফর্মের ফিল্ডের সংখ্যা সীমিত করার জন্য ব্যবহৃত হয়। এই অপশনটি নিরাপত্তা নিশ্চিত করার জন্য অত্যন্ত গুরুত্বপূর্ণ, কারণ এটি DoS আক্রমণ প্রতিরোধে সহায়তা করে [6]।
- **fileFilter**: এটি একটি ফাংশন যা নির্দিষ্ট শর্ত সাপেক্ষে ফাইল গ্রহণ বা বাতিল করার জন্য ব্যবহৃত হয়। যেমন, শুধুমাত্র নির্দিষ্ট ধরনের ফাইল (যেমন JPEG, PNG) আপলোড করার অনুমতি দেওয়া [6]।

Multer ফাইল আপলোডের বিভিন্ন পরিস্থিতি হ্যান্ডেল করার জন্য কয়েকটি পদ্ধতি প্রদান করে, যা একটি রুটে মিডলওয়্যার হিসেবে ব্যবহৃত হয়:

- **.single(fieldname)**: এটি একটি একক ফাইল আপলোডের জন্য ব্যবহৃত হয়। এখানে fieldname হলো <input type="file"> ট্যাগের name অ্যাট্রিবিউটের মান। এই পদ্ধতির মাধ্যমে আপলোড করা ফাইলের তথ্য req.file অবজেক্টে পাওয়া যায় [6]।
- **.array(fieldname[, maxCount])**: যখন একই fieldname থেকে একাধিক ফাইল আপলোড করা হয়, তখন এই পদ্ধতিটি ব্যবহার করা হয়। maxCount ঐচ্ছিকভাবে ফাইলের সংখ্যা সীমিত করতে পারে। এক্ষেত্রে, আপলোড করা ফাইলগুলোর একটি অ্যারে req.files অবজেক্টে পাওয়া যায় [6]।
- **.fields(fields)**: এটি বিভিন্ন fieldname থেকে আপলোড করা একাধিক ফাইল হ্যান্ডেল করার জন্য ব্যবহৃত হয়। এক্ষেত্রে fields হলো একটি অ্যারে যেখানে প্রতিটি অবজেক্টে name এবং ঐচ্ছিক maxCount থাকে। আপলোড করা ফাইলগুলোর তথ্য req.files অবজেক্টে পাওয়া যায়, যেখানে প্রতিটি fieldname একটি কী (key) হিসেবে কাজ করে [6]।
- **.none()**: এটি একটি বিশেষ পদ্ধতি যা multipart ফর্ম থেকে শুধুমাত্র টেক্সট ডেটা পার্স করার জন্য ব্যবহৃত হয়। যদি এই ধরনের একটি রিকোয়েস্টে কোনো ফাইল পাওয়া যায়, Multer একটি ত্রুটি ("LIMIT_UNEXPECTED_FILE") তৈরি করে [6]। এই পদ্ধতিটি প্রমাণ করে যে Multer কেবল একটি ফাইল আপলোড লাইব্রেরি নয়, বরং এটি একটি সম্পূর্ণ multipart/form-data পার্সার। এটি ডেভেলপারদেরকে একটি একক টুল ব্যবহার করে ফাইল এবং টেক্সট ডেটা উভয়ই হ্যান্ডেল করার সুবিধা দেয়, যা অন্যথায় আলাদা লাইব্রেরির প্রয়োজন হতো [1]।

স্টোরেজ ইঞ্জিনের গভীর বিশ্লেষণ: DiskStorage বনাম MemoryStorage

Multer-এর কার্যকারিতার একটি মূল অংশ হলো এর স্টোরেজ ইঞ্জিন, যা নির্ধারণ করে আপলোড করা ফাইলগুলো কোথায় এবং কীভাবে সংরক্ষণ করা হবে। Multer দুটি বিল্ট-ইন স্টোরেজ ইঞ্জিন নিয়ে আসে: DiskStorage এবং MemoryStorage [6]।

DiskStorage (ডিস্ক স্টোরেজ)

DiskStorage ইঞ্জিন আপলোড করা ফাইলগুলোকে সরাসরি সার্ভারের ফাইল সিস্টেমে সংরক্ষণ করার সম্পূর্ণ নিয়ন্ত্রণ দেয় [6]। এটি বেশিরভাগ অ্যাপ্লিকেশনের জন্য প্রস্তাবিত পদ্ধতি। এই ইঞ্জিন কনফিগার করার জন্য দুটি ফাংশন প্রয়োজন:

1. **destination(req, file, cb)**: এই ফাংশনটি ফাইলটিকে কোন ফোল্ডারে সংরক্ষণ করা হবে তা নির্ধারণ করে। এটি একটি ফাংশন হিসেবে ব্যবহৃত হলে ডেভেলপারকে ম্যানুয়ালি গন্তব্য ডিরেক্টরি তৈরি করতে হয় [5]।
2. **filename(req, file, cb)**: এই ফাংশনটি ফাইলের নাম কী হবে তা নির্ধারণ করে। নিরাপত্তা ঝুঁকি এড়াতে এবং নাম সংক্রান্ত দ্বন্দ্ব (naming conflicts) এড়াতে ব্যবহারকারীর দেওয়া মূল ফাইলের নাম সরাসরি ব্যবহার করা উচিত নয়। এর পরিবর্তে, Date.now(), uuid বা অন্যান্য অনন্য নাম তৈরি করা উচিত [6]। একটি গুরুত্বপূর্ণ বিষয় হলো, Multer স্বয়ংক্রিয়ভাবে ফাইলের এক্সটেনশন যোগ করে না; তাই ডেভেলপারকে
   path মডিউল ব্যবহার করে মূল এক্সটেনশন (file.originalname) নিয়ে তা নতুন ফাইলের নামের সাথে যুক্ত করতে হয় [5]।

MemoryStorage (মেমোরি স্টোরেজ)

MemoryStorage ইঞ্জিন আপলোড করা ফাইলগুলোকে সার্ভারের RAM-এ একটি Buffer অবজেক্ট হিসেবে সংরক্ষণ করে [6]। এই ইঞ্জিনটির কোনো কনফিগারেশন অপশন নেই। এটি ছোট ফাইল, যেমন প্রোফাইল ছবি, বা এমন ফাইল যা আপলোডের পরপরই অন্য কোনো সেবায় (যেমন Amazon S3) পাঠানো হবে, সেগুলোর জন্য উপযোগী। এই পদ্ধতিতে ডিস্কে কোনো I/O অপারেশন হয় না, যা কিছু ক্ষেত্রে পারফরম্যান্স উন্নত করতে পারে। তবে, এর একটি বড় নিরাপত্তা ঝুঁকি রয়েছে। বড় আকারের ফাইল বা খুব দ্রুত অনেকগুলো ছোট ফাইল আপলোড করা হলে এটি সার্ভারের মেমোরি দ্রুত ফুরিয়ে দিতে পারে, যা একটি DoS (Denial of Service) আক্রমণের কারণ হতে পারে [5]। তাই, যদি

MemoryStorage ব্যবহার করা হয়, তবে limits অপশন দিয়ে আপলোড ফাইলের আকার সীমিত করা অত্যন্ত জরুরি।

তৃতীয়-পক্ষের স্টোরেজ ইঞ্জিনের সাহায্যে Multer-কে বিভিন্ন ক্লাউড পরিষেবা যেমন Amazon S3, Google Cloud Storage, বা Azure Blob Storage এর সাথে যুক্ত করা যায়। এই ইঞ্জিনগুলো Multer-এর মূল কাঠামো ব্যবহার করে কিন্তু ফাইল সংরক্ষণের কাজটি সংশ্লিষ্ট ক্লাউড পরিষেবার উপর ছেড়ে দেয় [13]।

DiskStorage এবং MemoryStorage-এর মধ্যে সঠিক নির্বাচন করা অ্যাপ্লিকেশনের কার্যকারিতা এবং নিরাপত্তার জন্য অত্যন্ত গুরুত্বপূর্ণ। নিচের সারণীটি এই দুটি স্টোরেজ ইঞ্জিনের মূল পার্থক্যগুলো তুলে ধরে:

বৈশিষ্ট্য	DiskStorage	MemoryStorage

সংরক্ষণ মাধ্যম	ডিস্ক (হার্ড ড্রাইভ/SSD)	মেমোরি (RAM)
মূল ব্যবহার	স্থায়ীভাবে ফাইল সংরক্ষণ, যেমন ডকুমেন্ট বা মিডিয়া ফাইল।	ক্ষণস্থায়ী প্রক্রিয়াকরণ, যেমন ফাইল আপলোড করে ক্লাউড স্টোরেজে পাঠানো।
নিয়ন্ত্রণ	সম্পূর্ণ নিয়ন্ত্রণ (destination এবং filename ফাংশন ব্যবহার করে)।	কোনো কনফিগারেশন নেই, ফাইল Buffer হিসেবে থাকে।
পারফরম্যান্স	ডিস্ক I/O অপারেশনের কারণে মেমোরি স্টোরেজের চেয়ে ধীর।	দ্রুততর, যেহেতু কোনো ডিস্ক I/O নেই। তবে, বড় ফাইল মেমোরিতে রাখে।
প্রধান ঝুঁকি	Path Traversal আক্রমণ।	DoS (Denial of Service) আক্রমণের ঝুঁকি, যদি ফাইল আকার সীমিত না করা হয়।
ফাইল অ্যাক্সেস	আপলোডের পর req.file.path থেকে ফাইলটি অ্যাক্সেস করা হয়।	আপলোডের পর req.file.buffer থেকে সরাসরি বাইনারি ডেটা অ্যাক্সেস করা হয়।

এই তুলনামূলক বিশ্লেষণটি ডেভেলপারকে তাদের অ্যাপ্লিকেশনের প্রয়োজন অনুযায়ী সঠিক স্টোরেজ ইঞ্জিন বেছে নিতে সাহায্য করে। স্থায়ী সংরক্ষণের প্রয়োজন হলে DiskStorage হলো আদর্শ সমাধান, আর ক্ষণস্থায়ী প্রক্রিয়াকরণের জন্য MemoryStorage উপযুক্ত হতে পারে, তবে তা অবশ্যই limits অপশন ব্যবহার করে সুরক্ষিত রাখা উচিত।

নিরাপত্তা: Multer ব্যবহারের সময় যে বিষয়গুলো খেয়াল রাখতে হবে

Multer একটি শক্তিশালী টুল হলেও, এর ভুল ব্যবহার গুরুতর নিরাপত্তা ঝুঁকির কারণ হতে পারে। ফাইল আপলোডের প্রক্রিয়াটি নিজেই একটি সংবেদনশীল ক্ষেত্র, কারণ এটি সার্ভারের ফাইল সিস্টেমের সাথে সরাসরি ইন্টারঅ্যাক্ট করে। একজন বিশেষজ্ঞ হিসেবে Multer ব্যবহারের সময় কয়েকটি গুরুত্বপূর্ণ নিরাপত্তা অনুশীলন অনুসরণ করার পরামর্শ দেওয়া হয়।

DoS (Denial of Service) আক্রমণ প্রতিরোধ

MemoryStorage ব্যবহার করার সময় DoS আক্রমণের ঝুঁকি সবচেয়ে বেশি। যদি আপলোড করা ফাইলের আকার সীমিত না করা হয়, তবে একজন আক্রমণকারী বড় আকারের ফাইল বারবার আপলোড করে সার্ভারের RAM পূর্ণ করে দিতে পারে [5]। যখন মেমোরি ফুরিয়ে যায়, তখন Node.js অ্যাপ্লিকেশনটি ক্র্যাশ করে যায়, যা ব্যবহারকারীদের জন্য পরিষেবাটি বন্ধ করে দেয়। এই ঝুঁকি এড়াতে

multer() কনস্ট্রাক্টরে limits অপশনটি ব্যবহার করা অত্যাবশ্যক। উদাহরণস্বরূপ, limits: { fileSize: 1000000 } ব্যবহার করে ফাইলের আকার 1MB-তে সীমাবদ্ধ করা যায় [8]।

এছাড়াও, পূর্বে উল্লিখিত CVE-2025-7338 দুর্বলতাটি একটি ভিন্ন ধরনের DoS আক্রমণকে প্রকাশ করেছে। এই দুর্বলতাটি malformed বা বিকৃত রিকোয়েস্টের কারণে হতো, যা কোনো ফাইল আপলোড না করেও সার্ভারকে ক্র্যাশ করে দিতে পারতো [5]। এর প্রতিকার হিসেবে, Multer-এর সর্বশেষ সংস্করণ ব্যবহার করা অপরিহার্য, কারণ এটি বিকৃত রিকোয়েস্টের কারণে সৃষ্ট ত্রুটিগুলোকে সঠিকভাবে হ্যান্ডেল করার জন্য প্যাচ করা হয়েছে [9]।

Path Traversal (ডিরেক্টরি ট্রাভার্সাল) আক্রমণ প্রতিরোধ

DiskStorage ব্যবহার করার সময় আরেকটি প্রধান ঝুঁকি হলো Path Traversal আক্রমণ। এই ধরনের আক্রমণে একজন আক্রমণকারী আপলোড করা ফাইলের নামের মধ্যে ../ (ডট-ডট-স্ল্যাশ) সিকোয়েন্স ব্যবহার করে সার্ভারের রুট ডিরেক্টরিতে বা সংবেদনশীল ফাইল সিস্টেমে প্রবেশ করার চেষ্টা করে [15]। যদি ডেভেলপার

filename ফাংশনে ব্যবহারকারীর দেওয়া ফাইলের নাম সরাসরি ব্যবহার করেন, তবে এই আক্রমণটি সম্ভব হতে পারে [14]।

এই ঝুঁকি প্রতিরোধের জন্য, ফাইলনেম ফাংশনে সর্বদা একটি অনন্য এবং নিরাপদ নাম তৈরি করা উচিত, যেমন Date.now() বা uuid ব্যবহার করে। ফাইলের মূল এক্সটেনশনটি path.extname() ব্যবহার করে নিরাপদভাবে নেওয়া যেতে পারে এবং নতুন নামের সাথে যোগ করা যেতে পারে। এটি নিশ্চিত করে যে ব্যবহারকারীর ইনপুট ফাইলের পাথকে প্রভাবিত করতে পারবে না।

ফাইল ফিল্টারিং এবং অন্যান্য সেরা অনুশীলন

- **ফাইল ফিল্টারিং (fileFilter)**: শুধুমাত্র অনুমোদিত ফাইল প্রকার (MIME type) বা এক্সটেনশন গ্রহণ

করার জন্য fileFilter অপশন ব্যবহার করা উচিত। এটি আক্রমণকারীকে অনিরাপদ বা ম্যালিশিয়াস ফাইল আপলোড করা থেকে বিরত রাখে [6]।

- **গ্লোবাল মিডলওয়্যার হিসেবে ব্যবহার না করা**: Multer কখনোই একটি গ্লোবাল মিডলওয়্যার হিসেবে (app.use(multer())) ব্যবহার করা উচিত নয় [5]। এটি শুধুমাত্র সেই রুটগুলোতে ব্যবহার করা উচিত যেখানে ফাইল আপলোড হ্যান্ডেল করা হয়।
- **ত্রুটি হ্যান্ডলিং**: Multer-এর ত্রুটিগুলো দক্ষতার সাথে পরিচালনা করা অত্যন্ত জরুরি। যখন কোনো ত্রুটি ঘটে, যেমন ফাইলের আকার সীমা অতিক্রম করে, তখন Multer Express-এর ত্রুটি হ্যান্ডলিং সিস্টেমে একটি ত্রুটি প্রেরণ করে। ডেভেলপারদের উচিত এই ত্রুটিগুলো সঠিক কোড ব্যবহার করে ক্যাচ করা এবং ব্যবহারকারীকে একটি স্পষ্ট বার্তা প্রদান করা [6]।

একটি সম্পূর্ণ ব্যবহারিক প্রয়োগ: Express.js-এ Multer ব্যবহার করে ফাইল আপলোড

একটি বাস্তবসম্মত উদাহরণ Multer-এর ব্যবহারকে আরও স্পষ্ট করে তুলবে। এখানে একটি সাধারণ Express.js সার্ভার এবং একটি HTML ফর্ম দেখানো হয়েছে যা একটি একক ফাইল আপলোড হ্যান্ডেল করে।

ধাপ ১: প্রয়োজনীয় প্যাকেজ ইনস্টল

প্রথমে, আপনার প্রজেক্টের রুটে কমান্ড লাইনে গিয়ে Multer এবং Express ইনস্টল করুন:

```
$ npm install express multer 6
```

ধাপ ২: সার্ভার-সাইড কোড (server.js)

একটি server.js ফাইল তৈরি করুন এবং নিচের কোডটি লিখুন। এটি uploads নামক একটি ফোল্ডারে ফাইল সংরক্ষণ করবে এবং একটি অনন্য নাম তৈরি করবে।

JavaScript

```javascript
const express = require('express');
const multer = require('multer');
```

```
const path = require('path');
const app = express();
const port = 3000;

// DiskStorage ইঞ্জিন কনফিগার করা
const storage = multer.diskStorage({
 destination: function (req, file, cb) {
  cb(null, 'uploads/'); // ফাইল আপলোডের গন্তব্য ফোল্ডার
 },
 filename: function (req, file, cb) {
  const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
  // মূল ফাইলের এক্সটেনশন যোগ করা
  cb(null, file.fieldname + '-' + uniqueSuffix + path.extname(file.originalname));
 }
});

const upload = multer({
 storage: storage,
 // নিরাপত্তা নিশ্চিত করতে ফাইলের আকার 1MB-তে সীমাবদ্ধ করা
 limits: { fileSize: 1000000 },
 // শুধুমাত্র নির্দিষ্ট ফাইল টাইপ গ্রহণ করা
 fileFilter: function (req, file, cb) {
  const filetypes = /jpeg|jpg|png|gif/;
  const mimetype = filetypes.test(file.mimetype);
  const extname = filetypes.test(path.extname(file.originalname).toLowerCase());

  if (mimetype && extname) {
   return cb(null, true);
  }
  cb("Error: File upload only supports the following filetypes - " + filetypes);
 }
});

// একটি রুট তৈরি করা যা একক ফাইল আপলোড হ্যান্ডেল করবে
app.post('/upload', upload.single('myFile'), (req, res, next) => {
 if (req.file) {
  console.log('File uploaded:', req.file);
  // req.body তে ফর্মের টেক্সট ডেটা থাকে
  console.log('Text fields:', req.body);
  res.send('File uploaded successfully!');
 } else {
  res.status(400).send('No file uploaded.');
 }
```

```
});

// Multer-এর ত্রুটি হ্যান্ডেল করা
app.use((err, req, res, next) => {
  if (err instanceof multer.MulterError) {
    res.status(500).send({ error: err.message });
  } else {
    res.status(500).send({ error: 'An unknown error occurred during upload.' });
  }
});

app.listen(port, () => {
  console.log(`Server listening at http://localhost:${port}`);
});
```

ধাপ ৩: ক্লায়েন্ট-সাইড HTML ফর্ম (index.html)

একটি index.html ফাইল তৈরি করুন। মনে রাখতে হবে, এখানে enctype="multipart/form-data" অ্যাট্রিবিউটটি অপরিহার্য [1]।

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>File Upload</title>
</head>
<body>
  <h2>File Upload Example</h2>
  <form action="/upload" method="post" enctype="multipart/form-data">
    <label for="myFile">Select file to upload:</label><br>
    <input type="file" name="myFile" id="myFile"><br><br>
    <label for="description">File Description:</label><br>
    <input type="text" name="description" id="description"><br><br>
    <button type="submit">Upload File</button>
```

```
  </form>
</body>
</html>
```

এই উদাহরণটি Multer-এর একটি সম্পূর্ণ কার্যপ্রণালী তুলে ধরে, যেখানে ইনস্টলেশন থেকে শুরু করে সার্ভার এবং ক্লায়েন্ট-সাইড কোড এবং ত্রুটি হ্যান্ডলিং পর্যন্ত সবকিছু অন্তর্ভুক্ত রয়েছে [6]।

উপসংহার: Multer-এর সারসংক্ষেপ এবং পরবর্তী নির্দেশনা

Multer Node.js-এর জন্য ফাইল আপলোডের একটি শক্তিশালী এবং নির্ভরযোগ্য সমাধান। এটি multipart/form-data এনকোডিং হ্যান্ডেল করার মাধ্যমে জটিল ফাইল আপলোডের প্রক্রিয়াকে সরল করে তোলে। এর বিভিন্ন স্টোরেজ ইঞ্জিন, যেমন DiskStorage এবং MemoryStorage, ডেভেলপারদেরকে তাদের প্রয়োজন অনুযায়ী ফাইল সংরক্ষণের পদ্ধতি বেছে নেওয়ার স্বাধীনতা দেয়। DiskStorage স্থায়ী সংরক্ষণের জন্য উপযুক্ত, আর MemoryStorage ক্ষণস্থায়ী প্রক্রিয়াকরণের জন্য কার্যকর, তবে উভয় ক্ষেত্রেই নিরাপত্তা সংক্রান্ত সতর্কতা অবলম্বন করা অপরিহার্য।

একজন দক্ষ ডেভেলপার হিসেবে Multer ব্যবহার করার সময় কিছু গুরুত্বপূর্ণ বিষয় সবসময় মনে রাখা উচিত:

- ক্লায়েন্ট-সাইডে HTML ফর্মে অবশ্যই enctype="multipart/form-data" অ্যাট্রিবিউট ব্যবহার করতে হবে।
- অ্যাপ্লিকেশনের প্রয়োজন অনুযায়ী DiskStorage এবং MemoryStorage-এর মধ্যে সঠিক ইঞ্জিনটি নির্বাচন করতে হবে।
- limits অপশন এবং fileFilter ফাংশন ব্যবহার করে DoS এবং অন্যান্য নিরাপত্তা ঝুঁকি থেকে অ্যাপ্লিকেশনকে রক্ষা করতে হবে।
- Path Traversal আক্রমণ এড়াতে ফাইলের নাম তৈরি করার সময় ব্যবহারকারীর ইনপুট সরাসরি ব্যবহার না করে একটি অনন্য নাম তৈরি করতে হবে।
- Multer কখনোই গ্লোবাল মিডলওয়্যার হিসেবে ব্যবহার করা উচিত নয়, বরং শুধুমাত্র নির্দিষ্ট রুটে এর ব্যবহার সীমিত রাখা উচিত।

Multer সম্পর্কে আরও গভীরে জানার জন্য এর অফিসিয়াল ডকুমেন্টেশন একটি চমৎকার উৎস [4]। বাংলা ভাষায় প্রোগ্রামিং শেখার জন্য আরও কিছু সহায়ক সংস্থান রয়েছে যা একজন ডেভেলপারকে Node.js এবং অন্যান্য প্রযুক্তি সম্পর্কে জ্ঞান বৃদ্ধিতে সাহায্য করতে পারে:

- শাফায়েত'স প্ল্যানেট (Shafaetsplanet): এই ব্লগে বাংলায় অ্যালগরিদম এবং ডেটা স্ট্রাকচার নিয়ে বিস্তারিত আলোচনা রয়েছে [17]।
- বাংলাদেশ কম্পিউটার সোসাইটি (BCS): এটি বাংলাদেশে প্রোগ্রামিং কমিউনিটির একটি বড় অংশ, যেখানে প্রতিযোগিতামূলক প্রোগ্রামিং সংক্রান্ত বিভিন্ন আলোচনা এবং রিসোর্স পাওয়া যায় [18]।
- ইউটিউব চ্যানেল: লার্ন উইথ সুমিত (Learn with Sumit - LWS - Bangladesh) এবং অন্যান্য বাংলা ইউটিউব চ্যানেলে Node.js, Express.js এবং অন্যান্য ওয়েব প্রযুক্তির উপর অনেক মানসম্মত টিউটোরিয়াল পাওয়া যায় [19]।

Works cited

1. Define Multipart Form Data - GeeksforGeeks, accessed on August 30, 2025, https://www.geeksforgeeks.org/html/define-multipart-form-data/
2. HTTP Multipart and Security Implications - F5 DevCentral, accessed on August 30, 2025, https://community.f5.com/kb/technicalarticles/http-multipart-and-security-implications/316894
3. Multipart form data post method using express js - Stack Overflow, accessed on August 30, 2025, https://stackoverflow.com/questions/49067423/multipart-form-data-post-method-using-express-js
4. www.npmjs.com, accessed on August 30, 2025, https://www.npmjs.com/package/multer#:~:text=Multer%20is%20a%20node.js,of%20busboy%20for%20maximum%20efficiency.
5. multer - npm, accessed on August 30, 2025, https://www.npmjs.com/package/multer
6. Express multer middleware, accessed on August 30, 2025, https://expressjs.com/en/resources/middleware/multer.html
7. expressjs/multer: Node.js middleware for handling `multipart/form-data`. - GitHub, accessed on August 30, 2025, https://github.com/expressjs/multer
8. Handling File Uploads and file Validations in Node.js with Multer | by Mohsin Ansari, accessed on August 30, 2025, https://medium.com/@mohsinansari.dev/handling-file-uploads-and-file-validations-in-node-js-with-multer-a3716ec528a3
9. CVE-2025-7338 Detail - NVD, accessed on August 30, 2025, https://nvd.nist.gov/vuln/detail/CVE-2025-7338
10. Multer DoS Vulnerability (CVE-2025-7338): How a Single Malformed Upload Can Crash Your Node.js App - ZeroPath Blog, accessed on August 30, 2025, https://zeropath.com/blog/cve-2025-7338-multer-dos-vulnerability
11. multer - NPM - GeeksforGeeks, accessed on August 30, 2025, https://www.geeksforgeeks.org/node-js/multer-npm/
12. What is Multer and How to Use It to Upload Files in Express - Luis Llamas, accessed on August 30, 2025, https://www.luisllamas.es/en/upload-multiple-files-multer/
13. Node Js File-upload to S3 Using Multer Custom Storage Engine | by Raja Singh | Medium, accessed on August 30, 2025, https://medium.com/@singhcoolish/node-js-file-upload-to-s3-using-multer-custom-storage-engine-292a2e92cf12
14. Weak Multer File Name Manipulation | Learn Node.js Security, accessed on August 30, 2025, https://www.nodejs-security.com/learn/secure-file-handling/weak-multer-file-name-manipulation
15. Path Traversal | OWASP Foundation, accessed on August 30, 2025, https://owasp.org/www-community/attacks/Path_Traversal

16. What is path traversal, and how to prevent it? | Web Security Academy - PortSwigger, accessed on August 30, 2025, https://portswigger.net/web-security/file-path-traversal
17. Home || Shafaetsplanet, accessed on August 30, 2025, https://www.shafaetsplanet.com/home/
18. BCS - Bangladesh Competitive Programming Society, accessed on August 30, 2025, https://therealbcs.com/
19. Bangla (বাংলা) Programming Tutorials - YouTube, accessed on August 30, 2025, https://www.youtube.com/playlist?list=PLHiZ4m8vCp9O5_QoMvQbOAOR5WolyZ4OS
20. জাভা প্রোগ্রামিং শেখা - বাংলা টিউটোরিয়াল | Java Programming Tutorial in Bengali | Part-01 - YouTube, accessed on August 30, 2025, https://www.youtube.com/watch?v=vocx69ufZH4