MERN Back-End Development Section – 01

Express Js Rest Api Development

* Introduction of Rest Api
* Rest Api Best Practices
* Rest Api Security
* Express Rest Api Essential Packages
* Express Rest Api Structure
* Step By step Express Rest Api Development with Mongo.
* Rest Api Documentation Preparation.
* Practice Project.

Introduction of Rest Api

Rest → Representational State Transfer.

API → Application Programming Interface.

Rest API Best Practice

* Best practices of json.
* Request Response Model.
* Rest Api working flow.
* Rest API Auth and security.

Best Practices of JSON

JSON → JavaScript Object Notation.

* JSON is a light weight data-interchange format that is completely language independent.
* It was derived from JavaScript.
* The offical Internet media type for JSON is application/json.
* It was designed for human-readable data interchange.
* The filename extension is .json.

Uses of JSON

* Javascript based application.

JSON - DATA TYPE

Type	Description
Number	
String	
Boolean	
Array	
Value	
Object	
Whitespace	
null	

Json Bad Special Characters & Solution

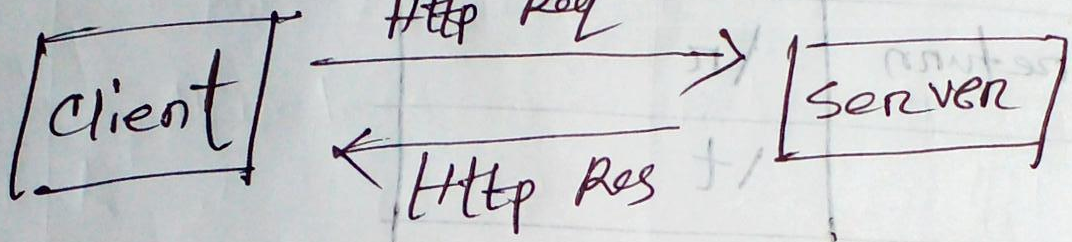
Characters	Replace With
Backspace	\b
Form feed	\f
Newline	\n
Carriage return	\r
Tab	\t
Double qoute	\"
Back slash	\\

→ JSON Lint → for ennore check in Json Data.

* Always create a Root element.

Request Response Model

Q. Client



HTTP CLIENT

* Browser level client → Server side Rendering
* Application level client → Client side Rendering

* Browser level client
  — only Get Request execute. করে।

* Application level client
  — সব ধরনের Request Application level client এ পাঠানো যায়।

HTTP client Library	platform	Language
Volly	Native Android	Java
Retrofit	Native Android	Java
Restsharp	ASP.NET	C#
Axios / JQuery / fetch	Mobile / Web / Desktop	JavaScript
cURL	Web	PhP
Alamofire	Native IOS	Swift

PostMan +Http client

Request compare Get vs Post,

Key Points	Get	POST
Back button/ Reload	Harmless	Data will be re submitted / browsen client ~~ma~~ re-submit -en
Bookmanked	Can be bookmanked	~~Never~~ Can not be bookmank
Cached	Can be	Never
Encoding Type	application/x-www-form-urlencoded	application/x-www-form-urlencoded on multipant/ form data. Use multipart encoding for binary data
History	Parametens remain in browsere history	Parametens are not save in browsen history
Restrictions on data length	Yes, when sending data the Get method adds the data to the URL; and the length of URL Limited Maximun URL Length is 2048 characters.	No restriction

Restriction on data type	Only ASCII characters allowed	No Restriction. Binary data is also allowed.
Security	GET is less secure compare to POST because data sent is part of the URL. Never use GET when sending passwords or other sensetive information	POST is little safer than GET because the parameter are not stored in browser history or in web server logs. (login, signup, transaction)
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Http Request Throttling

- যখন কোন user Rest Api তে Request করে তখন সে কতগুলো Request করতে পারবে সেই Rate কে Limit করে দেওয়াই হলো Throttling.

* Throttling is also known as request rate limiting.

* DOS attack রুখে দেওয়ার জন্য Throttling ব্যবহার করা হয়,

Http Response :-

Response Area	Standard Data Type
Body	Simple string JSON, Download, Redirect XML
Header	key Pair Value
Cookie	key Pair value

Response Status Code :-

Code	Meaning	Description
200 Automatically manage	Ok	Ok is the standard response for successful HTTP request
201	Created	
202	Accepted	
203	NoN Authoritative Information	
204	No content	
205	Reset content	
206	Parcial content	
400 AM	Bad Request	
401	Unauthorized	
403 AM	FoRbidden	
404 AM	Not Found	
405 AM	Method Not Allowed	
408 AM	Request timeOut	
500 AM	Internal Servven Error	
502 AM	Bad Gate Way	
503 AM	Service Unavaible	

REST API Basic Best Practice

API End Naming Best Practice :-

* CRUD (Create, Read, Update, and Delete) ধরনের functionality দিয়ে API End point এর নাম দেওয়া উচিত না.

* Example :- /users/{id} instead of /getUsers. ✗

* Forword Slashes for Hierarchy.

- একটার সাথে আরেকটার সম্পর্ক বোঝানোর জন্য বা indicate করতে "/" ব্যবহার করতে হয়.

* Punctuation for lists:-

- একই ধরনের multiple parameter pass করানোর ক্ষেত্রে বার বার "/" ব্যবহার না করে "," দিয়ে লেখা যায়.

→ /users/{id1},{id2}.

* Query parameters where necessary:-

'?' চিহ্ন ব্যবহার করে কোন কিছু filter or sort করানোর জন্য URL এর মধ্যে Query string বা parameter pass করাতে পারি,

- /users? location = USA

* Lower case letters and dashes:

- url এ সবসময় lowercase এ লিখতে হয়

→ /users/{id}/pending-orders ✓ instead of /users/{id}/Pending_Orders ✗

* No file extension:

Api এর যে end point থাকবে বা Api এর URL এর যে অংশ থাকবে সে অংশে কিভাবে কোনভাবে কোন file extension এর নাম ব্যবহার করা যাবে না

- /users/{id}/pending-orders ✓ instead of /users/{id}/Pending-0.XML ✗

* No trailing forward slash:

→ url এর শেষে slash দেওয়া যাবে না।

→ /users/{id}/pending-orders/ instead of /users/{id}/pending-orders/

✓ ✗

API Response Best Practices

* Response Header :-

→ Res Header এ অবশ্যই proper http response status code provide করতে হবে।

→ অবশ্যই proper content type, file type দেয়া করতে হবে।

→ cache status যদি থাকে তা দিতে হবে।

→ Authentication এর token অবশ্যই header এর মাধ্যমে provide করতে হবে।

→ Only string data is allowed for response Header.

→ যদি content এর length থাকে তা provide করতে হবে।

→ Provide response date and time.

→

API Response Body Best Practice

Response Body:

→ body তে status, code, message দেওয়া Avoid করতে হবে।

→ body যখন JSON ব্যবহার করবে তখন অবশ্যই ~~JSON~~ JSON এর best practice ব্যবহার করতে হবে।

→ For single result, can use string, Boolean directory.

→ Provide proper JSON encode-decode before writing JSON Body.

Response Cookies:

→ বর্তমান ইন্টারনেট web application যেভাবে cookie serve করে ঠিক সেভাবে Rest Api থেকে cookie serve করা হয়।

→ Avoid using Response cookies

→ if required use cookie encryption, decryption and other security policies.

API Request Handling Best Practices

When use GET():

→ যখন Request এর মধ্যে দিয়ে small amount of data pass করানোর প্রয়োজন হবে, যেখানে কোন big-data pass করানোর প্রয়োজন নেই, সেখানে GET() use করতে।

→ যখন আপনার action এর কারনে server এ কোন কিছু change হবে না তখন GET() use হয়।

→ যখন Request হবে database থেকে কোন specific parameter/condition দিয়ে data retrieve করার জন্য Request করবে হবে, তখন select করার জন্য, filter করার জন্য, এ all select, conditional select করার জন্য সেক্ষেত্রে Get() ব্যবহার করা হয়।

→ Get() এর মাধ্যমে খুব বেশি হলে request এর url & header ব্যবহার করা যায়; body ব্যবহার করা যায় না।

When use POST()

→ যখন আমরা Request এর মাধ্যমে কোন Data change করতে চাইব তখন,

→ যখন Request এর body বিশাল হয়, বড় large amount of data pass করতে হয় তখন,

→ client side থেকে যখন কোন documents, image, video, senven side এ upload করতে চাইব তখন

Request Body :-

→ যখন Request body তে কোন data পাঠাতে চাইব তখন structured way তে JSON Array/Object pattern এ পাঠাতে,

→ যখন Request body তে image, audio, video যখন পাঠাতে চাইব তখন সেই Type হবে multipart/form-data.

→ Authentication related কোন data body তে পাঠাব না, সেগুলো Header এ পাঠাব,

→

Request Header

→ Req Header should carry all security related information, like token, auth etc.

→ key : pair is allowed for header.

→ Req header should provide user agent information of client Application.

Rest Api কোথায় থেকে hit করা হচ্ছে সেই user-কে জানাতে user-agent ব্যবহৃত হয়।

→ If necessary CSRF/XSRF should provide via header.

→ Request header should associated with middleware controller, where necessary.

Api Controller Best Practices

→ Controller এর কাজ যে যে কোড লিখা হয় তা অবশ্যই clean হতে হবে, এবং controller এ কখনোই business logic place করা যাবে না।

→ controllers responsible for accepting http request.

→ ~~Con~~ ~~API~~ consider API versioning.

→ Use async/await

→ Follow solid principles to manage controller classes.

→ যেন controller Get() দিয়ে কাজ করে সেটা, কোনটা post() দিয়ে কাজ করে তা mention করে দিতে হবে।

→ Controller responsible for calling model, return response, redirect to action etc.

API Middleware Controller Best Practice

→ Middleware request এর পর, কিন্তু Response এর আগে execute হয়।

→ Middleware কে special types of controller বলা হয়।

Middleware Uses:

→ Use to implement Api key, user-agent restriction, CSRF, XSRF, token based Api Authentication.

→ Use to implement API request rate limit.

→ Logging of incoming HTTP requests.

→ Response এর আগে Redirection এর প্রয়োজন হলে সেকেত্র middleware দিয়ে Re-direct করতে পারি।

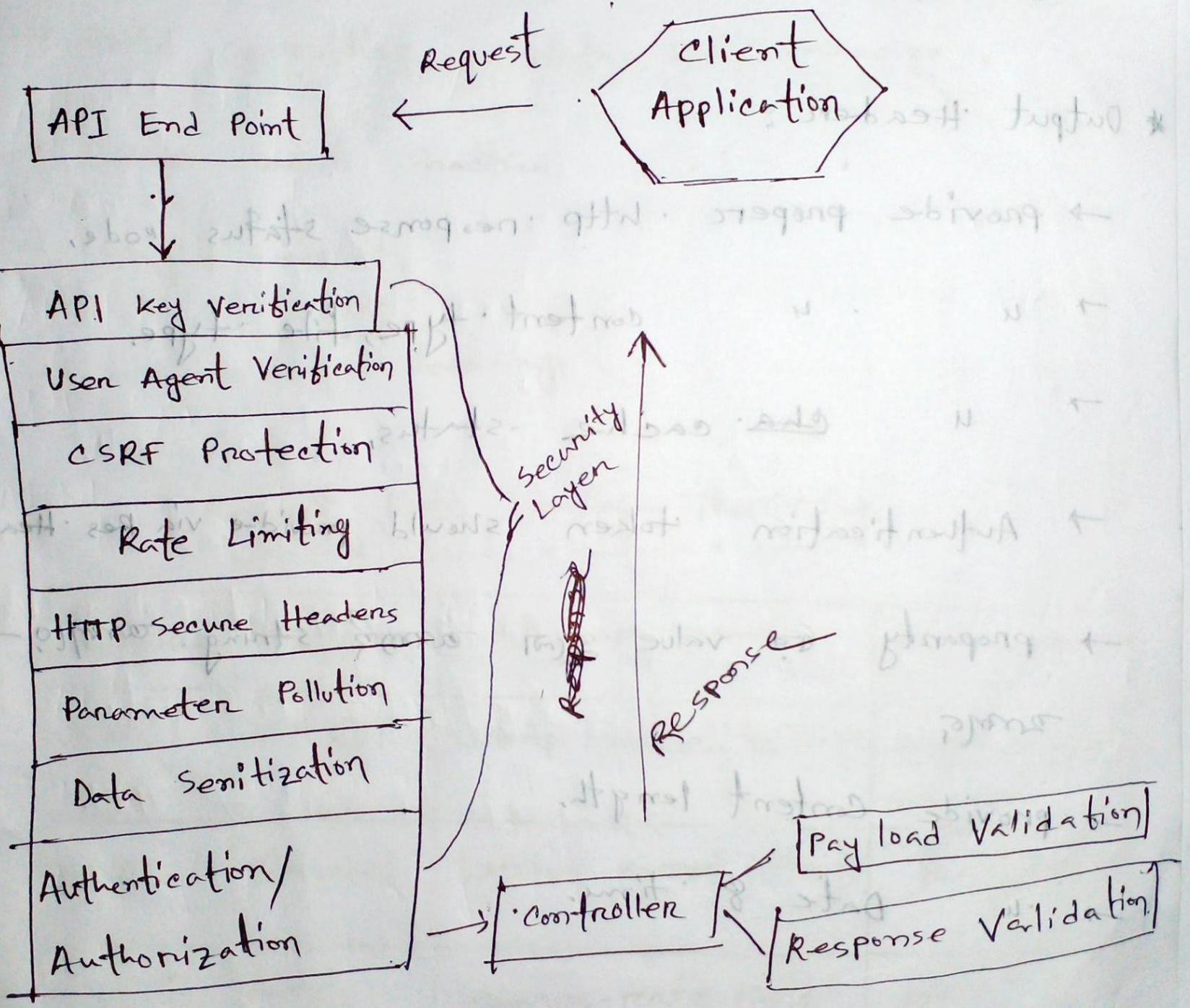
→ Middleware can inspect a request and decorate it or reject it, based on what it finds.

REST API Security Best Practices

Security Practices May varies:

1. May varies from application to application.
2. " " " developers to developers.
3. " " " environment to environment.
4. " " " use case to use case.

REST API SECURITY



Output Validation :-

* Output Header :

→ provide proper http response status code.

→ " " content-type, file-type.

→ " ~~cha~~ cache-status,

→ Authentication token should provide via Res. Header.

→ property ও value গুলো অবশ্যই string আকারে;
হবে;

→ provide content length.

→ " Date & time.

output Body:

→ Avoid providing status, code, massage.

→ Json best Practice.

→ single result.

→ en-co·ding – decoding.

Request Rate Limiting – Thorttling.

Language	Platform	Library name	Library Source
C#	ASP. NET	Web Api Throttle, Mvc Throttle	Nuget package manager
PHP	Laravel	Laravel kernel Default	Packgist
JS	Node/ Express JS	express-rate-limit	NPM

CSRF/XSRF Protection :-

Cross-site scripting থেকে রক্ষা করে CSRF/XSRF.

→ CSRF token সাধারণত header এ pass করা হয়।

→ যখন session শুরু হয় তখন CSRF token তৈরি করা হয় এবং unique হতে হয়।

→ For self Api CSRF token works well.

* C# → Asp.Net → AntiCSRF → Nuget Package Manager.
* PHP → Laravel → Laravel Default → Packagist.
* JS → Node/Express Js → npm i csrf → NPM.

User Agent Protection :-

- কোন Agent থেকে বা কোন ধরনের platform থেকে Rest Api কে call করা হচ্ছে সেই user platform কেই User Agent বলে.

→ User agent use করে Rest api কে search engine indexing, social media sharing এর মতো বিষয় protect করা যায়.

→ কেউ যদি তার Identity hiding করে req করছে সেক্ষেত্রে তার subspecies request stop করে দেওয়া যায়.

→ Rest Api usage history add করা যায়.

→ We can add device/Os usage restriction.

API key :-

→ The most straightforward method and the easiest way for authentication.

→ Api sending username:password/ID/keys

→ Api মুলোকে encoded and decode করে তার safe transmission ensure করা হয়।

→ Api key এর মাধ্যমে যখন কাউকে access দেওয়া হয় তখন cookies, session IDs, login pages, and other such specially solution এর প্রয়োজন হয় না।

Bearer Authentication / Auth 2.0

→ Bearer authentication (also called token auth) is an http authentication scheme that involves security tokens called bearer tokens, passes through request-response header. In General JSON web Tokens (JWT) used for this purpose.

* JWT Authentication token :-

→ যেকোন একটা কিছু property থাকে যেহেতু JWT এর যে library গুলো আছে সেগুলো JSON টাকে encrypte করে token generate করে, সেই token ব্যবহার করে user কে auth করা হয়।

* C# → ASP.NET → Jwt Bearer, Jose-Jwt — Nuget PM
* PHP → Laravel → firebase/php-Jwt → GitHub
* JS → Node/Express JS → npm i jsonwebtoken — NPM

JWT (JSON Web Token) :-

→ Json data/object কে transmiting করার সময়
তাকে encrypte করে একটা token generate
করা হয়। এবং এই token কে decrypte করে
Json data/object বের করা যায়।

তাহলে Json data/object কে securely transmit
করার জন্য এরকম encrypte & dycrypte করার
যে process আছে তাকে JWT.

Compact and self-contained way.

→ This is trusted process.

Uses :-

→ Authorization : Allowing the users to access routes,
services and resources.

→ Information Exchange : Way of securely transmitting
information between parties.