Response Best Practices:

Response Header:

- Provide proper http response status code.
- Provide proper content type, file type if any.
- Provide cache status if any.
- Authentication token should provide via response header.
- Only string data is allowed for response header.
- Provide content length if any.
- Provide response date and time.
- Follow request-response model described before.

Response Body:

- Avoid providing response status, code, message via response body
- Use JSON best practices for JSON response body.
- For single result, can use String, Boolean directly.
- Provide proper JSON encode-decode before writing JSON Body.
- Follow discussion on JSON described before.

Response Cookies:

- A Restful API may send cookies just like a regular Web Application that serves HTML
- Avoid using response cookies as it is violate stateless principle.
- If required use cookie encryption, decryption and other policies

Request Handling Best Practices:

When use GET():

- GET is used to request something from server with less amount of data to pass.
- When nothing should change on the server because of your action.
- When request only retrieves data from a web server by specifying parameters

- Get method only carries request url & header not request body.

When use POST():

- POST should be used when the server state changes due to that action.
- When request needs its body, to pass large amount of data.
- When want to upload documents , images , video from client to server

Request Body:

- Request body should be structured in JSON Array/ Object pattern
- Request body hold multipart/ form-data like images, audio, video etc
- Request body should not hold any auth related information.
- Request body should associated with specific request data model, setter getter can used for this

Request Header:

- Request header should carry all security related information, like token, auth etc.
- Only string Key:Pair value is allowed for header .
- Request header should provide user agent information of client application.
- If necessary CSRF/ XSRF should provide via header.
- Request header should associated with middleware controller, where necessary

API Controller Best Practices

- The controllers should always be as clean as possible. We shouldn't place any business logic inside it.
- Controllers should be responsible for accepting http request
- Consider API versioning
- Use async/await if at all possible.
- Follow solid principles to manage controller classes.
- Mention which method is responsible for GET() and which for POST().
- Controller should be only responsible for calling model, return response , redirect to action etc.

API Middleware Controller Best Practices

Middleware is a special types of controller executed after request but before in response. It is a type of filtering mechanism to ensure API securities and more. Middleware acts as a bridge between a request and a response.

Middleware Uses:

- Use to implement API key, user-agent restriction, CSRF, XSRF security, token based API authentication.
- Use to implement API request rate limit.
- Logging of incoming HTTP requests.
- Redirecting the users based on requests.
- Middleware can inspect a request and decorate it, or reject it, based on what it finds.

- Middleware is most often considered separate from your application logic.
- Middleware gives you enough freedom to create your own security mechanism.