**REST API best practises**

1. Rest Api should not indicate any kind of CRUD functionality.
2. Forward slashes are conventionally used to show the hierarchy between individual resources and collection.
3. When there is no hierarchical relationship such as the semicolon or comma should be used.
4. In order to sort of filter a collection a rest api should allow query parameters to be passed in the url.
5. Resources name should be in lowercase latter and dash can be used instead of underscore.
6. Leave file extensions out of your URIS.
7. Do not add a trailing forward slash to the end of uris.

**Response Header**

1. Proper http response status code should be provided.
2. Cash status should be provided.
3. Proper content type, file type is needed.
4. Authentication token should be provided with the help of response header.
5. Request response model should follow for best practices.
6. Response date and time should provide.

**Response Body**

1. Avoid providing response status, code , message via response body.
2. Use JSON best practices for JSON response body.
3. For single result can use string , Boolean directly.
4. Provide proper JSON encode-decode before writing JSON body.

**Response cookies**

1. Avoid using response cookies as it is violate stateless principles.
2. If required use cookie encryption, decryption and other policies.

**Request handling best practice**

**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 and header not request body.

**When use POST()**

* POST should be used when the server state changes due to that action.
* When request needs it body to pass large amount of data.
* When want to upload document, 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 image, video, audio etc
* Request body should be associated with specific request data model.

**Request Header**

* Request header should carry all security related information like token, auth etc.
* Only string key-value pair is allowed for header.
* Request header should associated with middleware controller.

**API controller best practices**

* Controller should be responsible for accepting http request.
* Consider API versioning.
* Use async/await if possible.
* Follow solid principles to manage controller classes
* Mention which method is responsible for GET and which is for POST.
* Controller should be only responsible for calling model, return response, redirect to action etc.

**API Middleware controlled 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 request and response.

**Uses of Middleware**

* Use to implement API key, user agent restrictions ,CSRF, XSRF security, token based API authentication.
* Use to implement API request rate limit.
* Logging of incoming http request.
* Gives enough freedom to create your own security mechanism.
* It can inspect a request and decorate it or reject it based on what it finds.
* Middleware is separate from application logic.
* Redirecting the users based on requests.