Certainly! Here’s the combined table with the "Example Scenarios" merged into the main table:

Spring Boot Status Code	Name	Typical Use Case	Detailed Use Case	Example Scenarios
HttpStatus.OK (200)	OK	Successful GET, PUT, PATCH, or POST request.	Indicates that the request was successful and the server has responded with the requested data or completed the action.	- **GET** `/users/123` returns user details. - **PUT** `/users/123` updates user details successfully. - **PATCH** `/users/123` modifies specific user fields.
HttpStatus.CREATED (201)	Created	Resource successfully created (typically for POST).	Used when a new resource has been created as a result of the request. The server should return the URL of the new resource.	- **POST** `/users` creates a new user and returns the URL of the newly created user. - **POST** `/orders` creates a new order and includes the order ID in the response.
HttpStatus.NO_CONTENT (204)	No Content	Successful request but no content to return (e.g., DELETE).	Indicates that the request was successful, but there is no additional information to send in the response body.	- **DELETE** `/users/123` successfully removes the user, no content is returned. - **PUT** `/users/123` updates user details without returning any data. - **PATCH** `/items/456` applies updates without returning the modified resource.
HttpStatus.BAD_REQUEST (400)	Bad Request	Malformed request syntax or invalid request parameters.	Used when the server cannot process the request due to client-side errors, such as invalid data formats or missing parameters.	- **POST** `/users` with an invalid JSON payload. - **PUT** `/users/123` with missing required fields. - **PATCH** `/items/789` with incorrect data format in the request body.

Spring Boot Status Code	Name	Typical Use Case	Detailed Use Case	Example Scenarios
HttpStatus.UNAUTHORIZED (401)	Unauthorized	Authentication required or failed.	Indicates that authentication is needed or provided credentials are incorrect. The response may include a `WWW-Authenticate` header.	- **GET** `/profile` without proper authentication headers. - **POST** `/secure-data` with invalid or expired token. - **GET** `/settings` with missing or invalid credentials.
HttpStatus.FORBIDDEN (403)	Forbidden	Server refuses to authorize the request.	Indicates that the client is authenticated but does not have permission to access the requested resource.	- **GET** `/admin/dashboard` where the user does not have admin rights. - **POST** `/users/123/privileged-action` where the user lacks permissions to perform the action. - **DELETE** `/admin/users/456` where user is not authorized to delete the user.
HttpStatus.NOT_FOUND (404)	Not Found	Requested resource could not be found.	Used when the requested resource or endpoint does not exist. This may be due to a mistyped URL or non-existent resource.	- **GET** `/products/999` where product ID 999 does not exist. - **GET** `/users/abc` with a non-numeric user ID. - **POST** `/data` to a non-existent endpoint.
HttpStatus.METHOD_NOT_ALLOWED (405)	Method Not Allowed	HTTP method used is not supported for the resource.	Indicates that the method used in the request is not allowed for the specified resource. The server should include an `Allow` header with supported methods.	- **POST** `/users/123` when only GET and PUT are allowed for that resource. - **PUT** `/articles` on an endpoint that only supports GET. - **DELETE** `/orders` on a resource that only supports POST and GET.

Spring Boot Status Code	Name	Typical Use Case	Detailed Use Case	Example Scenarios
HttpStatus.CONFLICT (409)	Conflict	Conflict with the current state of the resource (e.g., duplicate).	Used when the request could not be processed because of a conflict with the current state of the resource, such as a duplicate entry.	- **POST** `/users` with a username that already exists. - **PUT** `/products/123` with a conflicting update due to a data inconsistency. - **PATCH** `/orders/456` with an attempt to modify an order that has already been shipped.
HttpStatus.UNPROCESSABLE_ENTITY (422)	Unprocessable Entity	Request is well-formed but contains semantic errors.	Used when the request is syntactically correct but semantically invalid, such as validation errors or business logic constraints.	- **POST** `/users` with valid JSON but invalid data (e.g., a birthdate in the future). - **PUT** `/items/789` with data that violates business rules (e.g., a negative price). - **PATCH** `/profiles/123` with invalid data fields that fail validation.
HttpStatus.INTERNAL_SERVER_ERROR (500)	Internal Server Error	Generic server-side error.	Indicates that the server encountered an unexpected condition that prevented it from fulfilling the request. This often indicates a bug or server misconfiguration.	- Unexpected error occurs during processing of **GET** `/orders` due to a database failure. - **POST** `/users` results in a server-side exception. - **PATCH** `/profiles/123` fails due to an internal error.
HttpStatus.BAD_GATEWAY (502)	Bad Gateway	Invalid response from an upstream server.	Used when the server, while acting as a gateway or proxy, receives an invalid response from an upstream server. This often points to issues with dependent services.	- **GET** `/api/data` where the server receives an invalid response from an external API. - **POST** `/external-service` where the external service returns malformed data. - **GET** `/user/info` with a failed response from a backend service.

Spring Boot Status Code	Name	Typical Use Case	Detailed Use Case	Example Scenarios
HttpStatus.SERVICE_UNAVAILABLE (503)	Service Unavailable	Server temporarily unable to handle the request (e.g., overload).	Indicates that the server is currently unable to handle the request due to temporary overload or maintenance.	- **GET** `/reports` when the server is undergoing maintenance. - **POST** `/data` when the server is temporarily overloaded. - **GET** `/stats` during a temporary service disruption.
HttpStatus.GATEWAY_TIMEOUT (504)	Gateway Timeout	Timeout while waiting for a response from an upstream server.	Used when the server, acting as a gateway, does not receive a timely response from an upstream server or service.	- **GET** `/user/profile` when the request to an upstream service times out. - **POST** `/orders` when the backend service does not respond in time. - **GET** `/data` while waiting for a slow third-party service.

This format merges the "Example Scenarios" directly into the main table for a unified view.