**Method 1  
 @Operation(summary = "Api to interact with database to add Event")  
 @ApiResponses(value = {@ApiResponse(responseCode = "201", description = "Event added successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))}),  
 @ApiResponse(responseCode = "200", description = "Event already exists",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @PostMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ENDPOINT*, consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 public ResponseEntity<EventResponse> saveEvent(@Valid @RequestBody final EventRequest eventRequest)  
 throws JsonProcessingException {  
 String correlationId = GenericUtil.*sanitizeValues*(eventRequest.getCorrelationId());  
 *LOG*.info("Document Event data correlation id: {}", correlationId);  
  
 Optional<EventResponse> optionalEventResponse = DocumentGeneratorEventStoreService.*saveEvent*(eventRequest);  
  
 if (optionalEventResponse.isPresent()  
 && optionalEventResponse.get().getCode().equals(HttpStatus.*CREATED*.value())) {  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(optionalEventResponse.get());  
 }  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(optionalEventResponse.orElse(null));  
 }  
Annotations: The method is annotated with various Spring and OpenAPI annotations that provide metadata about the API endpoint.**

**@Operation: This annotation is used to specify the summary of the operation, which is a brief description of what the API endpoint does.**

**@ApiResponses: This annotation is used to define possible responses for the API operation. In this case, it defines two responses: HTTP status code 201 (Created) and HTTP status code 200 (OK).**

**HTTP POST Mapping: The method is mapped to HTTP POST requests using @PostMapping. The value attribute specifies the URL path for this endpoint.**

**Request Body: The @RequestBody annotation indicates that the eventRequest parameter should be bound to the JSON payload in the request body. It also specifies that the request body should be validated using @Valid.**

**Processing: Inside the method, the code starts by sanitizing the correlationId from the eventRequest. This likely involves some form of data cleaning or validation.**

**Logging: It logs the sanitized correlationId using a logger.**

**Service Call: It then calls DocumentGeneratorEventStoreService.saveEvent(eventRequest) to save the event data in the database. This service method appears to return an Optional<EventResponse>.**

**Response Handling: Depending on the result of the service call, the code constructs an appropriate response. If the optionalEventResponse is present and has a status code of 201 (Created), it returns an HTTP response with status 201 and the EventResponse in the response body. If not, it returns an HTTP response with status 200 (OK) and either the EventResponse if it's present or null if it's not.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Method 2  
 @Operation(summary = "API to interact with Database to add status for event.")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "Event status added successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @PostMapping(  
 value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ENDPOINT* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ID\_PATH\_PARAM*,  
 consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 public ResponseEntity<EventResponse> createEventStatus(@NotNull @PathVariable final String eventId,  
 @Valid @RequestBody final CreateEventStatusRequest  
 createEventStatusRequest) {  
 String correlationId = GenericUtil.*sanitizeValues*(createEventStatusRequest.getCorrelationId());  
 *LOG*.info("Document Event data correlation id: {}", correlationId);  
 Optional<EventResponse> responseMetadataOptional =  
 documentGeneratorEventStoreService.saveEventStatus(eventId, createEventStatusRequest);  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(responseMetadataOptional.orElse(null));  
 }  
Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to interact with a database to add status for an event.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventResponse.**

**HTTP POST Mapping: The method is mapped to HTTP POST requests using**

**@PostMapping. The value attribute specifies the URL path for this endpoint. It includes placeholders such as {eventId} that are intended to be replaced with actual values when a request is made.**

**@PathVariable: The eventId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. The value of this path variable will be used as an identifier for the event.**

**@RequestBody: The createEventStatusRequest parameter is annotated with @RequestBody, which means it will be bound to the JSON payload in the request body. It is also validated using @Valid, which typically means that the framework will validate the incoming JSON against a predefined schema or validation rules.**

**The code starts by sanitizing the correlationId from the createEventStatusRequest. This likely involves some form of data cleaning or validation.**

**It logs the sanitized correlationId using a logger.**

**The method then calls documentGeneratorEventStoreService.saveEventStatus(eventId, createEventStatusRequest). This service method is responsible for saving the event status data associated with the given eventId.**

**After the service call, the code constructs an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is set to the result of the service call wrapped in an Optional. If the responseMetadataOptional is present, it returns the EventResponse. If not, it returns null.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Method 3  
 @Operation(summary = "API to retrieve event by eventId")  
  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "Event retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ENDPOINT* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ID\_PATH\_PARAM*)  
 public ResponseEntity<EventResponse> retrieveEventByEventId(  
 final HttpServletRequest httpRequest, @NotNull @PathVariable final String eventId) {  
 *LOG*.info("Getting event data from eventId:: {}", GenericUtil.*sanitizeValues*(eventId));  
 Optional<EventResponse> eventDataResponse = documentGeneratorEventStoreService.fetchEventByEventId(eventId);  
 return eventResponseMapper(httpRequest, eventDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve an event by its eventId.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventResponse.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint. It includes placeholders such as {eventId} that are intended to be replaced with actual values when a request is made.**

**Path Variables:**

**@PathVariable: The eventId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. The value of this path variable will be used as an identifier for the event that needs to be retrieved.**

**Request Handling:**

**The method takes an additional parameter, HttpServletRequest httpRequest. This allows access to the details of the incoming HTTP request, which can be useful for extracting headers, query parameters, or other information about the request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving event data based on the eventId. The GenericUtil.sanitizeValues(eventId) likely performs some data cleaning or validation on the eventId.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchEventByEventId(eventId). This service method is responsible for retrieving the event data associated with the given eventId. The result is wrapped in an Optional<EventResponse>.**

**Response Handling:**

**After the service call, the code calls a eventResponseMapper method, passing in the httpRequest and the eventDataResponse as parameters. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is the result of the eventResponseMapper method, which likely prepares the final response based on the eventDataResponse.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 4  
 @Operation(summary = "API to retrieve error details by eventId")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "Error retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ENDPOINT* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ID\_PATH\_PARAM* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*ERROR\_ENDPOINT*)  
 public ResponseEntity<EventResponse> retrieveErrorsResponseByEventId(  
 final HttpServletRequest httpRequest, @NotNull @PathVariable final String eventId) {  
 *LOG*.info("Getting errors data from eventId: {}", GenericUtil.*sanitizeValues*(eventId));  
 Optional<EventResponse> eventDataResponse =  
 documentGeneratorEventStoreService.fetchErrorsByEventId(eventId);  
 return eventResponseMapper(httpRequest, eventDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve error details by eventId.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventResponse.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint. It includes placeholders such as {eventId} that are intended to be replaced with actual values when a request is made. Additionally, it includes the path segment for "error," suggesting that this endpoint is specifically for retrieving error details.**

**Path Variables:**

**@PathVariable: The eventId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. The value of this path variable will be used as an identifier for the event for which error details are being retrieved.**

**Request Handling:**

**The method takes an additional parameter, HttpServletRequest httpRequest. This allows access to the details of the incoming HTTP request, which can be useful for extracting headers, query parameters, or other information about the request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving error details based on the eventId. The GenericUtil.sanitizeValues(eventId) likely performs some data cleaning or validation on the eventId.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchErrorsByEventId(eventId). This service method is responsible for retrieving error details associated with the given eventId. The result is wrapped in an Optional<EventResponse>.**

**Response Handling:**

**After the service call, the code calls a eventResponseMapper method, passing in the httpRequest and the eventDataResponse as parameters. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is the result of the eventResponseMapper method, which likely prepares the final response based on the error details retrieved for the event.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 5  
 @Operation(summary = "API to retrieve notify details by eventId")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "notify retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ENDPOINT* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ID\_PATH\_PARAM* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*NOTIFY\_ENDPOINT*)  
 public ResponseEntity<EventResponse> retrieveNotifyResponseByEventId(  
 final HttpServletRequest httpRequest, @NotNull @PathVariable final String eventId) {  
 *LOG*.info("Getting Notify data from eventId: {}", GenericUtil.*sanitizeValues*(eventId));  
 Optional<EventResponse> eventDataResponse =  
 documentGeneratorEventStoreService.fetchNotifyDetailsByEventId(eventId);  
 return eventResponseMapper(httpRequest, eventDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve notification details by eventId.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventResponse.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint. It includes placeholders such as {eventId} that are intended to be replaced with actual values when a request is made. Additionally, it includes the path segment for "notify," suggesting that this endpoint is specifically for retrieving notification details.**

**Path Variables:**

**@PathVariable: The eventId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. The value of this path variable will be used as an identifier for the event for which notification details are being retrieved.**

**Request Handling:**

**The method takes an additional parameter, HttpServletRequest httpRequest. This allows access to the details of the incoming HTTP request, which can be useful for extracting headers, query parameters, or other information about the request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving notification details based on the eventId. The GenericUtil.sanitizeValues(eventId) likely performs some data cleaning or validation on the eventId.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchNotifyDetailsByEventId(eventId). This service method is responsible for retrieving notification details associated with the given eventId. The result is wrapped in an Optional<EventResponse>.**

**Response Handling:**

**After the service call, the code calls a eventResponseMapper method, passing in the httpRequest and the eventDataResponse as parameters. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is the result of the eventResponseMapper method, which likely prepares the final response based on the notification details retrieved for the event.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 6  
 @Operation(summary = "API to retrieve summary report by date range")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "summary report is retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventSummaryResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_SUMMARY\_ENDPOINT*)  
 public ResponseEntity<EventSummaryResponse> retrieveSummaryReportByDateRange(  
 final HttpServletRequest httpRequest, @NotNull @RequestParam final String type,  
 @NotNull @RequestParam final String startDate, @NotNull @RequestParam final String endDate) {  
 *LOG*.info("Getting Summary report Dates: {}", GenericUtil.*sanitizeValues*(startDate) + " "  
 + GenericUtil.*sanitizeValues*(endDate));  
 Optional<EventSummaryResponse> eventDataResponse =  
 documentGeneratorEventStoreService.fetchEventSummaryByDateRange(GenericUtil.*sanitizeValues*(type),  
 GenericUtil.*sanitizeValues*(startDate), GenericUtil.*sanitizeValues*(endDate));  
 return eventControlResponseMapper(httpRequest, eventDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve a summary report by date range.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventSummaryResponse. This suggests that the response will contain summary information about events within the specified date range.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint, which includes the path segment for "event\_summary." Clients can access this endpoint to request summary reports.**

**Request Parameters:**

**@RequestParam: The method expects three query parameters:**

**type: This parameter is used to specify the type of summary report to retrieve.**

**startDate: This parameter represents the start date of the date range for the summary report.**

**endDate: This parameter represents the end date of the date range for the summary report. These parameters are annotated with @NotNull, indicating that they are required for the request.**

**Request Handling:**

**The method takes an additional parameter, HttpServletRequest httpRequest. This allows access to the details of the incoming HTTP request, which can be useful for extracting headers, query parameters, or other information about the request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving a summary report for the specified date range and type. The GenericUtil.sanitizeValues(startDate) and GenericUtil.sanitizeValues(endDate) likely perform some data cleaning or validation on the input date values.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchEventSummaryByDateRange with the sanitized type, startDate, and endDate as parameters. This service method is responsible for fetching the summary report data based on the provided date range and type. The result is wrapped in an Optional<EventSummaryResponse>.**

**Response Handling:**

**After the service call, the code calls an eventControlResponseMapper method, passing in the httpRequest and the eventDataResponse as parameters. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is the result of the eventControlResponseMapper method, which likely prepares the final response based on the summary report data retrieved for the specified date range and type. The response type is EventSummaryResponse.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 7  
 @Operation(summary = "API to retrieve document details by customerAccountUuid and inputId")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "document retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*CUSTOMER\_ACCOUNTS* + DocumentGeneratorEventStoreConstants.*SLASH* + *CUSTOMER\_ACCOUNT\_PATH\_PARAM* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*DOCUMENT*,  
 params = {DocumentGeneratorEventStoreConstants.*INPUT\_ID*})  
 // /customer-accounts/{customerAccountUuid}/document  
 public ResponseEntity<EventResponse> retrieveDocumentResponseByInputId(  
 final HttpServletRequest httpRequest, @NotNull @PathVariable final UUID customerAccountUuid,  
 @NotNull @RequestParam(value = DocumentGeneratorEventStoreConstants.*INPUT\_ID*) final String inputId) {  
 *LOG*.info("Getting document data from customerAccountUuid and inputId: {}", GenericUtil.*sanitizeValues*(inputId));  
  
 Optional<EventResponse> eventDataResponse =  
 documentGeneratorEventStoreService.fetchDocumentByInputId(customerAccountUuid, inputId);  
 return eventResponseMapper(httpRequest, eventDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve document details by customerAccountUuid and inputId.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventResponse.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint, which includes placeholders for customerAccountUuid and inputId. Clients can access this endpoint to request document details.**

**Path Variables:**

**@PathVariable: The customerAccountUuid parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. This UUID represents the customer account for which the document is being retrieved.**

**Request Parameters:**

**@RequestParam: The method expects a query parameter named inputId. This parameter is annotated with @NotNull, indicating that it is required for the request. The value attribute specifies the name of the query parameter.**

**Request Handling:**

**The method takes an additional parameter, HttpServletRequest httpRequest. This allows access to the details of the incoming HTTP request, which can be useful for extracting headers, query parameters, or other information about the request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving document data based on the provided inputId. The GenericUtil.sanitizeValues(inputId) likely performs some data cleaning or validation on the input inputId.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchDocumentByInputId with the customerAccountUuid and inputId as parameters. This service method is responsible for fetching the document details associated with the specified customerAccountUuid and inputId. The result is wrapped in an Optional<EventResponse>.**

**Response Handling:**

**After the service call, the code calls an eventResponseMapper method, passing in the httpRequest and the eventDataResponse as parameters. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**The response's content type is specified as JSON, and the response body is the result of the eventResponseMapper method, which likely prepares the final response based on the document details retrieved for the specified customerAccountUuid and inputId. The response type is EventResponse.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 8  
 @Operation(summary = "API to retrieve document details by eventId and requestStatus")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "documentList retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = DocumentRetrievalResponse.class))})})  
 @GetMapping(  
 value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*INVOICE* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*HISTORY* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*EVENT\_ID\_PATH\_PARAM*)  
 public ResponseEntity<DocumentRetrievalResponse> retrieveDocumentListByEventIdAndStatus(  
 final HttpServletRequest httpRequest,  
 @NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER) final String correlationId,  
 @NotNull @RequestHeader(BOSConstants.APPLICATION\_LABEL\_HEADER) final String applicationLabel,  
 @NotNull @PathVariable final String eventId,  
 @NotNull @RequestParam(value = DocumentGeneratorEventStoreConstants.*NOTSTATUS*, required = false)  
 final String notStatus) {  
 *LOG*.debug("Getting list of document from eventId: {} where status is not: {}", eventId, notStatus);  
 Optional<DocumentRetrievalResponse> documentResponse =  
 documentGeneratorEventStoreService.fetchDocumentListByEventIdAndStatus(eventId, notStatus);  
 return new ResponseMapper().documentResponseMapper(httpRequest, documentResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve a list of document details by eventId and requestStatus.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type DocumentRetrievalResponse. This response is expected to contain details about a list of documents.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint, which includes placeholders such as {eventId}. Clients can access this endpoint to request a list of document details.**

**Path Variables:**

**@PathVariable: The eventId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. This eventId is used to identify the event for which documents are being retrieved.**

**Request Parameters:**

**@RequestParam: The method expects an optional query parameter named notStatus. This parameter is annotated with @NotNull, indicating that it is required for the request. The value attribute specifies the name of the query parameter.**

**Request Headers:**

**The method expects several request headers, including correlationId, applicationLabel, which are annotated with @NotNull. These headers are expected to be present in the incoming HTTP request.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving a list of documents based on the provided eventId and optionally a notStatus.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchDocumentListByEventIdAndStatus with the eventId and notStatus as parameters. This service method is responsible for fetching a list of document details based on the specified eventId and optionally filtering by notStatus. The result is wrapped in an Optional<DocumentRetrievalResponse>.**

**Response Handling:**

**After the service call, the code creates a new instance of ResponseMapper (which is presumably a custom mapper class) and calls documentResponseMapper on it. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It wraps the response from the documentResponseMapper method. The response's content type is specified as JSON, and the response body is the result of the mapping operation, which is expected to be of type DocumentRetrievalResponse.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 9  
 @Operation(summary = "API to interact with Database to save document service request status.")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "Request status saved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = DocumentRetrievalResponse.class))})})  
 @PostMapping(  
 value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*INVOICE* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*DOCUMENT* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*DOCUMENT\_ID\_PATH\_PARAM*,  
 consumes = MediaType.*APPLICATION\_JSON\_VALUE*)  
 public ResponseEntity<DocumentRetrievalResponse> createDocumentServiceRequestStatus(  
 @NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER) final String correlationId,  
 @NotNull @RequestHeader(BOSConstants.APPLICATION\_LABEL\_HEADER) final String applicationLabel,  
 @NotNull @PathVariable final UUID documentId,  
 @Valid @RequestBody final CreateDocumentServiceRequestStatus createDocumentServiceRequestStatus) {  
 *LOG*.debug("DocumentServiceRequest data correlation id: {}", correlationId);  
 Optional<DocumentRetrievalResponse> responseMetadataOptional =  
 documentGeneratorEventStoreService.saveDocumentServiceRequestStatus(documentId,  
 createDocumentServiceRequestStatus);  
 if (responseMetadataOptional.get().getCode().equals(HttpStatus.*CREATED*.value())) {  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(responseMetadataOptional.get());  
 }  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(responseMetadataOptional.orElse(null));  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to save the status of a document service request in the database.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type DocumentRetrievalResponse. This response is expected to contain information about the status of the document service request.**

**HTTP POST Mapping: The method is mapped to HTTP POST requests using @PostMapping. The value attribute specifies the URL path for this endpoint, which includes placeholders such as {documentId}. Clients can access this endpoint to submit a document service request status.**

**Path Variables:**

**@PathVariable: The documentId parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. This UUID represents the identifier of the document for which the service request status is being saved.**

**Request Headers:**

**The method expects certain request headers to be present in the HTTP request, including correlationId and applicationLabel. These headers are annotated with @NotNull, indicating that they are required for the request. These headers typically provide context and identification for the request.**

**Request Body:**

**@RequestBody: The createDocumentServiceRequestStatus parameter is annotated with @RequestBody, which means it will be bound to the JSON payload in the request body. It is also validated using @Valid, which typically means that the framework will validate the incoming JSON against predefined schema or validation rules. This parameter likely contains data related to the document service request status.**

**Logging:**

**The code logs a message using a logger, indicating that it is handling a document service request with a specific correlation ID.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.saveDocumentServiceRequestStatus with the documentId and createDocumentServiceRequestStatus as parameters. This service method is responsible for saving the document service request status in the database. The result is wrapped in an Optional<DocumentRetrievalResponse>.**

**Response Handling:**

**After the service call, the code checks the HTTP status code of the response. If the response indicates that the request was successful (HTTP status code 201 - Created), it returns an HTTP response with a status of 201 and includes the response body as the DocumentRetrievalResponse.**

**If the response status is not 201, it returns an HTTP response with a status of 200 (OK), and includes the response body as the DocumentRetrievalResponse. In this case, if the response was not successful, the DocumentRetrievalResponse is typically expected to provide information about the failure.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 10  
 private ResponseEntity<EventResponse> eventResponseMapper(final HttpServletRequest httpRequest,  
 final Optional<EventResponse> eventResponse) {  
 final TransactionContext context = (TransactionContext) httpRequest  
 .getAttribute(BOSConstants.TRANSACTION\_CONTEXT);  
  
 if (eventResponse.isPresent()) {  
 eventResponse.get().setCode(HttpStatus.*OK*.value());  
 eventResponse.get().setApplicationLabel(context.getApplicationLabel());  
 eventResponse.get().setCorrelationId(context.getCorrelationID());  
 eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.*SUCCESS*);  
  
 if (!Optional.*ofNullable*(eventResponse.get().getEventDataResponse()).isPresent()) {  
 eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.*NOT\_FOUND*);  
 }  
 }  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(eventResponse.orElse(null));  
 }**

**Method Parameters:**

**final HttpServletRequest httpRequest: This parameter represents the incoming HTTP request. It is used to access the TransactionContext associated with the request.**

**final Optional<EventResponse> eventResponse: This is an optional parameter that represents the EventResponse object to be mapped and processed.**

**Transaction Context:**

**The method first retrieves the TransactionContext from the httpRequest object using the getAttribute method. The TransactionContext typically contains contextual information related to the transaction or request, such as the application label and correlation ID.**

**Modification of EventResponse:**

**If the eventResponse is present (i.e., not empty), the method proceeds to modify it:**

**It sets the HTTP status code to 200 (OK) using eventResponse.get().setCode(HttpStatus.OK.value()).**

**It sets the application label and correlation ID from the TransactionContext into the eventResponse object using eventResponse.get().setApplicationLabel(context.getApplicationLabel()) and eventResponse.get().setCorrelationId(context.getCorrelationID()), respectively.**

**It sets the message in the eventResponse to "SUCCESS" using eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.SUCCESS).**

**Additional Message Setting:**

**The code checks if the eventDataResponse property of the eventResponse is not present or is null (!Optional.ofNullable(eventResponse.get().getEventDataResponse()).isPresent()).**

**If it's not present, it sets the message in the eventResponse to "NOT\_FOUND" using eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.NOT\_FOUND).**

**Response Creation:**

**Finally, the method creates and returns an HTTP response (ResponseEntity) with a status of 200 (OK). It sets the content type to JSON using .contentType(MediaType.APPLICATION\_JSON) and the response body is the modified eventResponse object using .body(eventResponse.orElse(null))**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 11  
 private ResponseEntity<EventSummaryResponse> eventControlResponseMapper(final HttpServletRequest httpRequest,  
 final Optional<EventSummaryResponse>  
 eventResponse) {  
 final TransactionContext context = (TransactionContext) httpRequest  
 .getAttribute(BOSConstants.TRANSACTION\_CONTEXT);  
  
 if (eventResponse.isPresent()) {  
 eventResponse.get().setCode(HttpStatus.*OK*.value());  
 eventResponse.get().setApplicationLabel(context.getApplicationLabel());  
 eventResponse.get().setCorrelationId(context.getCorrelationID());  
 eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.*SUCCESS*);  
 }  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(eventResponse.orElse(null));  
 }**

**Method Parameters:**

**final HttpServletRequest httpRequest: This parameter represents the incoming HTTP request. It is used to access the TransactionContext associated with the request.**

**final Optional<EventSummaryResponse> eventResponse: This is an optional parameter that represents the EventSummaryResponse object to be mapped and processed.**

**Transaction Context:**

**The method first retrieves the TransactionContext from the httpRequest object using the getAttribute method. The TransactionContext typically contains contextual information related to the transaction or request, such as the application label and correlation ID.**

**Modification of EventSummaryResponse:**

**If the eventResponse is present (i.e., not empty), the method proceeds to modify it:**

**It sets the HTTP status code to 200 (OK) using eventResponse.get().setCode(HttpStatus.OK.value()).**

**It sets the application label and correlation ID from the TransactionContext into the eventResponse object using eventResponse.get().setApplicationLabel(context.getApplicationLabel()) and eventResponse.get().setCorrelationId(context.getCorrelationID()), respectively.**

**It sets the message in the eventResponse to "SUCCESS" using eventResponse.get().setMessage(DocumentGeneratorEventStoreConstants.SUCCESS).**

**Response Creation:**

**Finally, the method creates and returns an HTTP response (ResponseEntity) with a status of 200 (OK). It sets the content type to JSON using .contentType(MediaType.APPLICATION\_JSON) and the response body is the modified eventResponse object using .body(eventResponse.orElse(null)).**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 12  
 @Operation(summary = "API to retrieve document history details by customerAccountUuid")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "response Retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = EventHistoryResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*GET\_HISTORY\_API\_URL*)  
 public ResponseEntity<EventHistoryApiResponse> retrieveDocumentResponseByCustomerAccountUuid(  
 final HttpServletRequest httpRequest,  
 @NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER)  
 final String correlationId,  
 @NotNull @RequestHeader(BOSConstants.APPLICATION\_LABEL\_HEADER)  
 final String applicationLabel,  
 @NotBlank @PathVariable final String customerAccountUuid) throws JsonProcessingException {  
 List<InvoiceHistoryDetails> eventResponseHistory =  
 documentGeneratorEventStoreService.fetchResponseByCustomerAccountUuid(customerAccountUuid);  
 return new ResponseMapper().createResponseEntity(httpRequest, eventResponseHistory);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve document history details by customerAccountUuid.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response of type EventHistoryApiResponse. This response is expected to contain history details related to documents.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint, which is defined as DocumentGeneratorEventStoreConstants.GET\_HISTORY\_API\_URL. Clients can access this endpoint to request document history details.**

**Request Headers:**

**The method expects certain request headers to be present in the HTTP request, including correlationId and applicationLabel. These headers are annotated with @NotNull, indicating that they are required for the request. These headers typically provide context and identification for the request.**

**Path Variable:**

**@PathVariable: The customerAccountUuid parameter is annotated with @PathVariable, indicating that it should be extracted from the URL path. This parameter represents the customer account UUID for which document history details are being retrieved.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving document history data for a specific customerAccountUuid.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchResponseByCustomerAccountUuid with the customerAccountUuid as a parameter. This service method is responsible for fetching history details related to documents associated with the specified customerAccountUuid. The result is a list of InvoiceHistoryDetails.**

**Response Handling:**

**After the service call, the code creates an instance of ResponseMapper (which is presumably a custom mapper class) and calls createResponseEntity on it, passing in the httpRequest and the eventResponseHistory. This suggests that there might be some additional processing or mapping of the response before it is returned to the client.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It wraps the response from the createResponseEntity method. The response's content type is specified as JSON, and the response body is the result of the mapping operation, which is expected to be of type EventHistoryApiResponse.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Method 13  
 @Operation(summary = "API to retrieve pending document request by status")  
 @ApiResponses(value = {  
 @ApiResponse(responseCode = "200", description = "Document retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = DocumentRequestServiceResponse.class))})})  
 @GetMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*DOCUMENT*)  
 public ResponseEntity<List<DocumentRequestServiceResponse>> retrieveDocumentDetailsByTypeAndStatus(  
 final HttpServletRequest httpRequest,  
 @NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER)  
 final String correlationId,  
 @NotNull @RequestHeader(BOSConstants.APPLICATION\_LABEL\_HEADER)  
 final String applicationLabel,  
 @RequestParam(required = false) final String documentType,  
 @RequestParam(required = false, defaultValue =  
 DocumentGeneratorEventStoreConstants.*DOCUMENT\_STATUS\_PENDING*) final String documentStatus,  
 @RequestParam(required = false, defaultValue =  
 DocumentGeneratorEventStoreConstants.*NO\_OF\_RECORDS\_COUNT*) final int noOfRecords) {  
 List<DocumentRequestServiceResponse> documentDataResponse = documentServiceRequestService.  
 fetchByDocumentTypeAndRequestStatus(documentType,  
 documentStatus, noOfRecords);  
 return ResponseEntity.*status*(HttpStatus.*OK*).contentType(MediaType.*APPLICATION\_JSON*)  
 .body(documentDataResponse);  
 }**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve pending document requests by status.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response containing a list of DocumentRequestServiceResponse.**

**HTTP GET Mapping: The method is mapped to HTTP GET requests using @GetMapping. The value attribute specifies the URL path for this endpoint, which is defined as DocumentGeneratorEventStoreConstants.SLASH + DocumentGeneratorEventStoreConstants.DOCUMENT. Clients can access this endpoint to request document details.**

**Request Headers:**

**The method expects certain request headers to be present in the HTTP request, including correlationId and applicationLabel. These headers are annotated with @NotNull, indicating that they are required for the request. These headers typically provide context and identification for the request.**

**Request Parameters:**

**@RequestParam: The method accepts several query parameters:**

**documentType (optional): This parameter allows clients to filter document requests by type. If not provided, it's treated as optional.**

**documentStatus (optional, defaulting to "PENDING"): This parameter allows clients to specify the status of document requests. If not provided, it defaults to "PENDING."**

**noOfRecords (optional, defaulting to a predefined constant): This parameter allows clients to specify the maximum number of records to retrieve. If not provided, it defaults to a predefined constant value.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving document data based on the provided parameters, including documentType, documentStatus, and noOfRecords.**

**Service Call:**

**The method then calls documentServiceRequestService.fetchByDocumentTypeAndRequestStatus with the provided parameters (documentType, documentStatus, and noOfRecords). This service method is responsible for fetching document details based on the specified criteria. The result is a list of DocumentRequestServiceResponse objects.**

**Response Handling:**

**After the service call, the method creates an HTTP response using ResponseEntity. It sets the HTTP status to 200 (OK) to indicate that the operation was successful.**

**Response Entity:**

**The response's content type is specified as JSON using .contentType(MediaType.APPLICATION\_JSON), and the response body is the list of DocumentRequestServiceResponse objects returned by the service call.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Method 14  
 @Operation(summary = "API to retrieve document details by documentId/documentIds")  
 @ApiResponses(value = {@ApiResponse(responseCode = "200", description = "response Retrieved successfully",  
 content = {@Content(mediaType = "application/json",  
 schema = @Schema(implementation = DocumentIdsResponse.class))})})  
 @PostMapping(value = DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*INVOICE* + DocumentGeneratorEventStoreConstants.*SLASH* + DocumentGeneratorEventStoreConstants.*DOCUMENTS*)  
 public ResponseEntity<DocumentIdsResponse> retrieveDocumentsByDocumentIds(  
 final HttpServletRequest httpRequest,  
 @Valid @RequestBody final DocumentIdsRequest documentIds,  
 @NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER)  
 final String correlationId,  
 @NotNull @RequestHeader(BOSConstants.APPLICATION\_LABEL\_HEADER)  
 final String applicationLabel) {  
 List<DocumentResponse> documentResponse = documentGeneratorEventStoreService  
 .fetchDocumentsByDocumentIds(documentIds);  
 return new ResponseMapper().createResponseEntity(documentResponse, correlationId, applicationLabel);  
 }  
}**

**Annotations:**

**@Operation: This annotation provides metadata about the API operation. The summary attribute briefly describes the purpose of this API, which is to retrieve document details by document ID or IDs.**

**@ApiResponses: This annotation defines a single response for the API operation. It specifies that when the operation is successful (HTTP status code 200), it will return a JSON response containing DocumentIdsResponse.**

**HTTP POST Mapping: The method is mapped to HTTP POST requests using @PostMapping. The value attribute specifies the URL path for this endpoint, which is constructed using various constants like DocumentGeneratorEventStoreConstants.INVOICE and DocumentGeneratorEventStoreConstants.DOCUMENTS. Clients can access this endpoint to request document details.**

**Request Headers:**

**The method expects certain request headers to be present in the HTTP request, including correlationId and applicationLabel. These headers are annotated with @NotNull, indicating that they are required for the request. These headers typically provide context and identification for the request.**

**Request Body:**

**@RequestBody: The documentIds parameter is annotated with @RequestBody, which means it will be bound to the JSON payload in the request body. It represents a DocumentIdsRequest object containing document IDs for which details are requested.**

**Logging:**

**The code logs a message using a logger, indicating that it is retrieving document data based on the provided document IDs.**

**Service Call:**

**The method then calls documentGeneratorEventStoreService.fetchDocumentsByDocumentIds with the documentIds parameter. This service method is responsible for fetching document details based on the specified document IDs. The result is a list of DocumentResponse objects.**

**Response Handling:**

**After the service call, the code creates an instance of ResponseMapper (which is presumably a custom mapper class) and calls createResponseEntity on it, passing in the documentResponse, correlationId, and applicationLabel. This suggests that there might be some additional processing or mapping of the response before it is returned.**

**Response Entity:**

**Finally, the method returns an HTTP response using ResponseEntity. It wraps the response from the createResponseEntity method. The response's content type is specified as JSON, and the response body is the result of the mapping operation, which is expected to be of type DocumentIdsResponse.**