

Testing Tasks

Query: pullRequests

1.

Testcase Identifier: testPullRequestsWithNullStatus

Testcase Description:

When status is null, the query should return a complete list of views with pull requests and their changed files, based on pagination, without filtering status.

Inputs:

```
{
  "pageLimit": 5,
  "status": null,
  "offset": 0
}
```

Output:

Response

STATUS 200 | 27.0ms | 1.2

Root > data > pullRequests

CSVJSON

pullRequests										changedFiles							
0	targetBranch	main								0							
	status	PENDING															
	sourceCommit	12H3J4															
	inlineComments	CSV															
		reactionCount	line	file		author		content	filename								
	0	4	lineNum 10	filename	file1.js	username	user1	This is an inline comment	file1.js								
1	generalComments	CSV								1							
		reactionCount	content			author				file2.js							
	0	4	This is a general comment			username	user1										
	description	Fix some issues															
	id	1															
1	targetBranch	main								0							
	status	MERGE_CONFLICT															
	sourceCommit	0B324F															
	inlineComments	[]															
	generalComments	[]															
	description	Add new feature								1							
	id	2								file2.js							
2	targetBranch	main								0							
	status	REJECTED															
	sourceCommit	5DFG34															
	inlineComments	[]															
	generalComments	[]															
	description	Fix some issues								1							
	id	3								file2.js							
3	targetBranch	main								0							
	status	MERGED															
	sourceCommit	9WR234															
	inlineComments	[]															
	generalComments	[]															
	description	Add new feature								1							
	id	4								file2.js							

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testPullRequestWithStatusFiltering

Testcase Description:

When status is not null, the query should return a list of pull requests that contains the designated status based on pagination.

Inputs:

```
{
  "pageLimit": 3,
  "status": "PENDING",
  "offset": 0,
}
```

Output:

Response

STATUS 200 | 41.0ms | 486B

Root > data > pullRequests

CSV

JSON

pullRequests										changedFiles
0	targetBranch	main								0 file1.js
	status	PENDING								
	sourceCommit	12H3J4								
	inlineComments									
		reactionCount	line	file	author	content	filename			
	0	4	lineNum 10	filename file1.js	username user1	This is an inline comment				
	generalComments									
	reactionCount	content			author					
0	4	This is a general comment			username user1					
description										1 file2.js
Fix some issues										
id										
1										

Remarks: The output is correct. The happy path is tested to be successful.

3.

Testcase Identifier: testPaginationOffsetOutOfBound

Testcase Description:

When the offset of current pages is out of bound, which means in current page limits, the offset is larger than the actual number of pages, throw an error.

Inputs:

```
{  
  "pageLimit": 8,  
  "status": null,  
  "offset": 2,  
}
```

Output:

The screenshot shows a REST client interface. At the top, the status is 200, with a response time of 16.0ms and a size of 1.9KB. The root of the response is an object with two properties: 'data' and 'errors'. The 'errors' property is an array with one element at index 0, which is the string 'Offset out of bounds.'.

Root	
data	
errors	
0	Offset out of bounds.

Remarks: The output is correct, since this input means requesting to view the object between the 9th index and the 16th index, which is out of bound. The negative path is handled with an error thrown.

4.

Testcase Identifier: testPaginationUnfilledPage

Testcase Description:

When the last page is not full, the function returns a list of pull requests that has a size smaller than page limit without an error.

Input:

```
{
  "pageLimit": 3,
  "status": null,
  "offset": 1,
}
```

Output:

Root

data

</

Remarks: The output is correct, since the first page contains a limit of 3 objects so the last page only contains 1 object (4 pull requests in DB in total). The happy path is tested to be successful.

Mutation: addPullRequest

1.

Testcase Identifier: testAddPullRequestWithCorrectInput

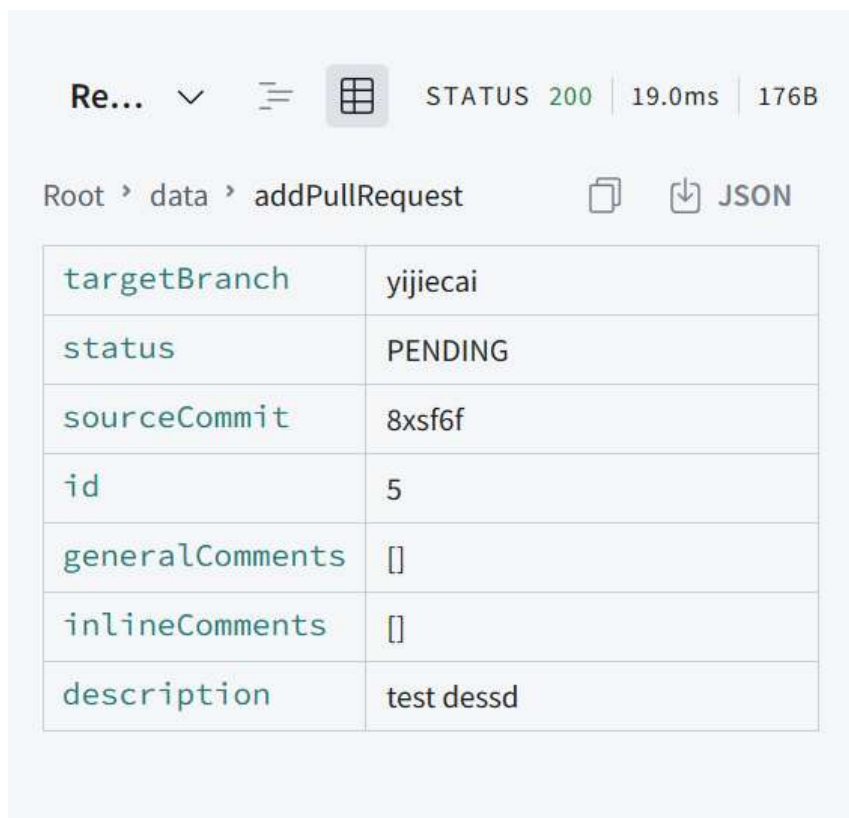
Testcase Description:

When inputting a description, a non-empty source commit and a non-empty target branch, the function will add a pull request with the fields above and an empty list of general comments and inline comments, and a default pending status.

Inputs:

```
{  
  "description": "test dessd",  
  "sourceCommit": "8xsf6f",  
  "targetBranch": "yijiecai"  
}
```

Output:



Re... STATUS 200 | 19.0ms | 176B

Root > data > addPullRequest JSON

targetBranch	yijiecai
status	PENDING
sourceCommit	8xsf6f
id	5
generalComments	[]
inlineComments	[]
description	test dessd

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testAddPullRequestWithEmptySourceCommit




Testcase Description:

When inputting an empty string as a source commit, the function throws an error.

Inputs:

```
{  
  "description": "test empty commit",  
  "sourceCommit": "",  
  "targetBranch": "yijiecai"  
}
```

Output:

Root			 JSON
data			
errors		 CSV	
	0	Source commit or target branch cannot be empty.	

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testAddPullRequestSameSourceCommit

Testcase Description:

Inputs:

```
{
  "description": "test same commit",
  "sourceCommit": "8xsf6f",
  "targetBranch": "yijiecai"
}
```

Output:

The screenshot shows a REST client interface. At the top, the status bar indicates 'STATUS 200', '27.0ms', and '1.9KB'. Below this, the 'Root' section is visible with a 'data' field and an 'errors' field. The 'errors' field is expanded, showing a table with one row containing the error message: 'Pull request with this source commit already exists.'.

Root	
data	
errors	
0	Pull request with this source commit already exists.

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: addGeneralComment

1.

Testcase Identifier: testAddGeneralCommentWithCorrectInput


Testcase Description:

When inputting an existing pull request id, the content for the comment, and an existing author id, a general comment will be added to the general comment database and the comment

Inputs:

```
{
  "prId": "1",
  "content": "this is a newwww general comment",
  "author": "2"
}
```

Output:

Response   					STATUS 200 34.0ms 242B		
Root > data > addGeneralComment						 CSV	 JSON
	reactionCount	content	author.username	id			
0	2	This is a general comment	user1	10001			
1	0	this is a newwww general comment	user2	10002			

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testAddGeneralCommentWithInvalidUser

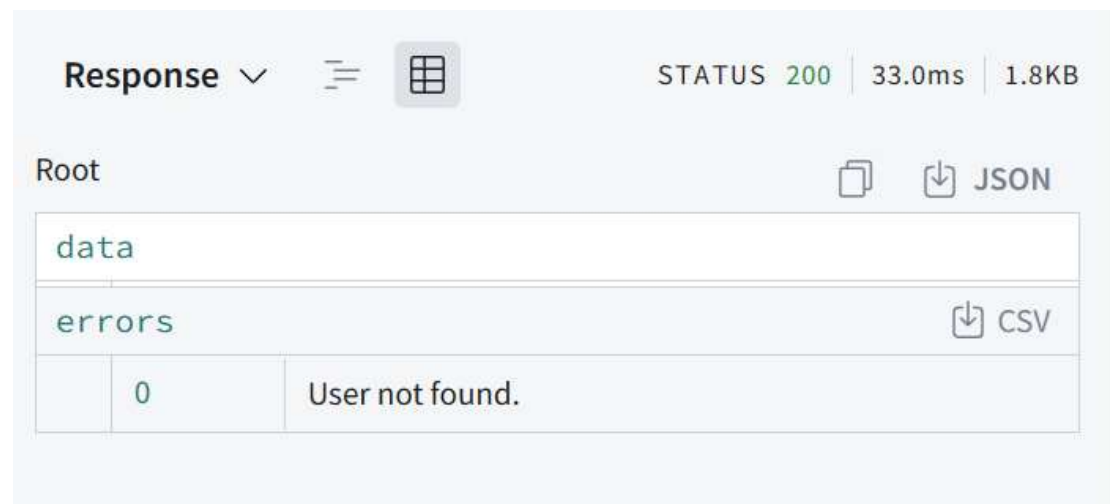
Testcase Description:

If the user ID does not exist in the database, the function throws an error.

Inputs:

```
{
  "prId": "1",
  "content": "this is a no user general comment",
  "author": "1000"
}
```

Output:



The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS 200** | **33.0ms** | **1.8KB**
- Root** (with copy and JSON download icons)
- data** (expandable field)
- errors** (expandable field with CSV download icon)
- | | |
|---|-----------------|
| 0 | User not found. |
|---|-----------------|

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testAddGeneralCommentWithInvalidPRID

Testcase Description:

If the pull request ID does not exist in the database, the function throws an error.

Inputs:

```
{  
  "prId": "1324",  
  "content": "this is a no user general comment",  
  "author": "1"  
}
```

Output:

The screenshot shows a REST client interface. At the top, it displays 'Response' with a dropdown arrow, a list icon, and a table icon. To the right, it shows 'STATUS 200', '50.0ms', and '1.8KB'. Below this, the 'Root' of the response is shown with a copy icon and a 'JSON' label. The response body is a JSON object with two main keys: 'data' and 'errors'. The 'data' key has a value of an empty array. The 'errors' key has a value of an array containing one object. This object has a 'code' of 0 and a 'message' of 'Pull request not found.'.

Root	
data	
errors	
0	Pull request not found.

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: addInlineComment

1.

Testcase Identifier: testAddInlineCommentWithCorrectInput

Testcase Description:

When inputting an existing pull request ID, a string for content, an existing user as author, and existing file ID and line ID, the function creates a new inline comment with the fields above, and adds it to the inline comment database and the corresponding pull request.

Inputs:

```
{
  "prId": "1",
  "content": "this is a nEWWW inline comment",
  "author": "2",
  "fileId": "1",
  "lineNum": 20
}
```

Output:

Response  

STATUS 200 | 36.0ms | 409B

Root > data > addInlineComment   CSV  JSON

	line	file	reactionCount	content	author	id
0	lineNum 10	filename file1.js	2	This is an inline comment	username user1	20001
		changedLines  CSV				
		lineNum				
		0 10				
1	lineNum 20	filename file1.js	0	this is a nEWWW inline comment	username user2	20002
		changedLines  CSV				
		lineNum				
		0 10				

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testAddInlineCommentWithInvalidPRID

Testcase Description:

When the input for pull request ID does not exist in the database, the function throws an error.

Inputs:

```
{
  "prId": "1324",
  "content": "this is a no user inline comment",
  "author": "1",
  "fileId": "1",
  "lineNum": 10
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS 200** (green text)
- 28.0ms** (execution time)
- 1.8KB** (response size)
- Root** (breadcrumb)
- data** (JSON key)
- errors** (JSON key)
- 0** (array index)
- Pull request not found.** (error message)
- JSON** (download button)
- CSV** (download button)

Index	Message
0	Pull request not found.

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testAddInlineCommentWithInvalidUserID

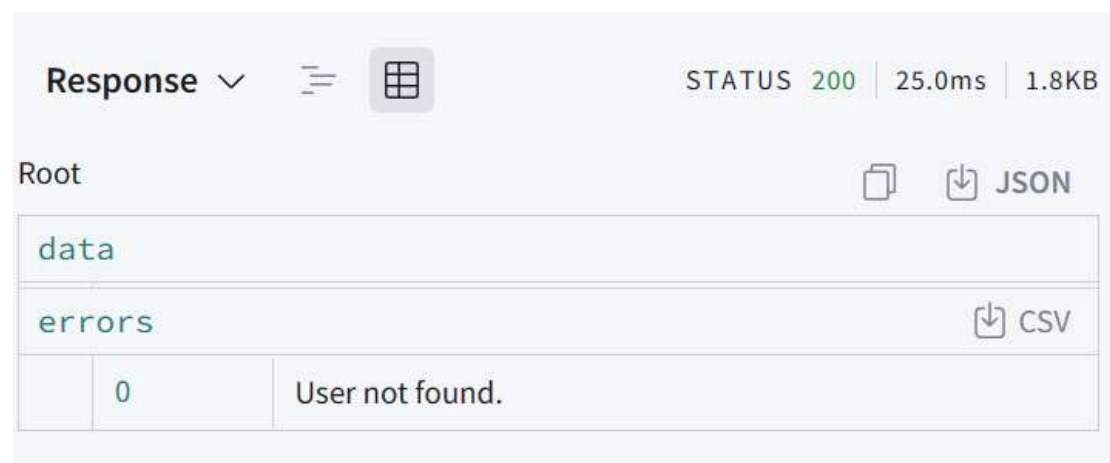
Testcase Description:

When the input for author (user's) ID does not exist in the database, the function throws an error.

Inputs:

```
{
  "prId": "1",
  "content": "this is a nEWWW inline comment",
  "author": "2095",
  "fileId": "1",
  "lineNum": 20
}
```

Output:



The screenshot shows a REST client interface with the following details:

- Response** tab selected, showing a **STATUS 200**, **25.0ms** execution time, and **1.8KB** size.
- Root** of the response structure is displayed.
- The response body is a JSON object with two main fields: **data** and **errors**.
- The **errors** field is expanded, showing a table with one entry:

0	User not found.
---	-----------------

Remarks: The output is correct. The negative path is handled with an error thrown.

4.

Testcase Identifier: testAddInlineCommentWithInvalidFileID

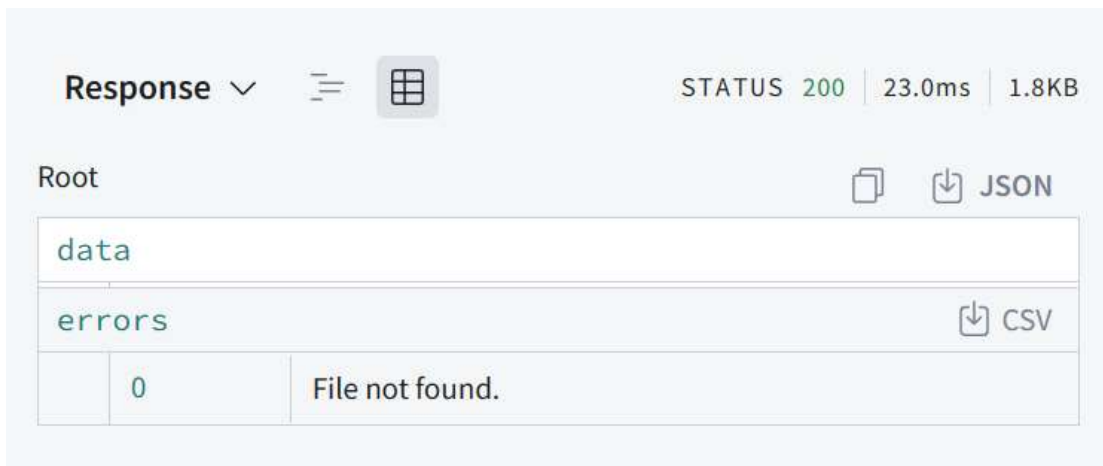
Testcase Description:

When the input for the file ID does not exist in the database, the function throws an error.

Inputs:

```
{
  "prId": "1",
  "content": "this is a nEWWW inline comment",
  "author": "2",
  "fileId": "100",
  "lineNum": 20
}
```

Output:



The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS** 200 | **23.0ms** | **1.8KB**
- Root** (copy icon, download icon, **JSON**)
- data** (copy icon)
- errors** (download icon, **CSV**)
- | | |
|---|-----------------|
| 0 | File not found. |
|---|-----------------|

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: addReaction

1.

Testcase Identifier: testAddReactionToGeneralComment

Testcase Description:

When inputting existing comment ID of a general comment, a string for the type of reaction, and an existing author (user), the function creates a new reaction with the fields for type and creator (user), and adds it to the specified comment, and returns the current reaction list in the specified general comment.

Inputs:

```
{
  "commentId": "10001",
  "type": "+1",
  "author": "1"
}
```

Output:



The screenshot shows a REST client interface with the following details:

- Response** (expanded) with icons for expand, collapse, and table view.
- STATUS** 200 | **25.0ms** | **133B**
- Path: **Root > data > addReaction**
- Actions: Copy, Download, CSV, JSON
- Table:**

	id	type	user.username
0	2	smiley	user2
1	5	+1	user1

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testAddReactionToInlineComment

Testcase Description:

When inputting existing comment ID of an inline comment, a string for the type of reaction, and an existing author (user), the function creates a new reaction with the fields for type and creator (user), and adds it to the specified comment, and returns the current reaction list in the specified inline comment.

Inputs:

```
{
  "commentId": "20001",
  "type": "+1",
  "author": "2"
}
```

Output:

Response

STATUS 200 | 54.0ms | 192B

Root > data > addReaction

CSV

JSON

	id	type	user.username
0	3	thumbsUp	user3
1	4	thumbsUp	user4
2	5	+1	user2

Remarks: The output is correct. The happy path is tested to be successful.

3.

Testcase Identifier: testAddReactionForInvalidUser

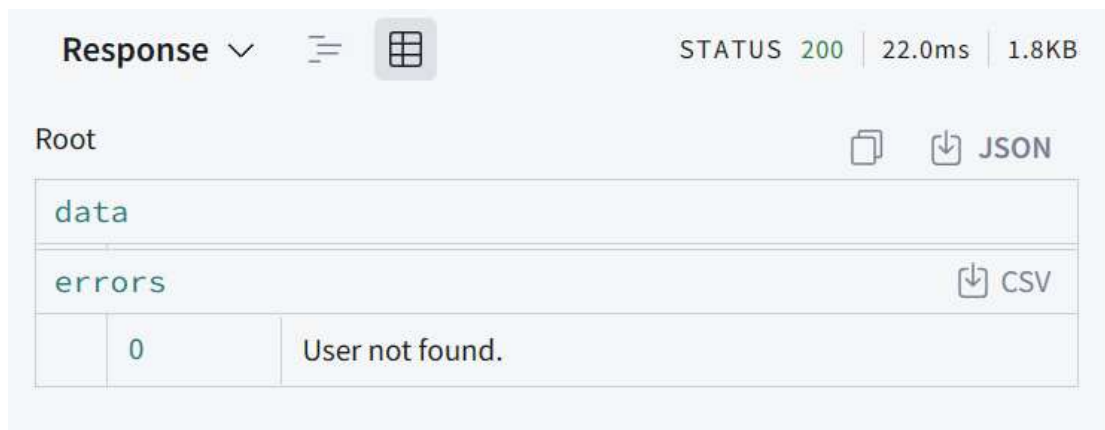
Testcase Description:

When inputting a user ID that does not exist in the database, the function throws an error.

Inputs:

```
{  
  "commentId": "20001",  
  "type": "+1",  
  "author": "2000"  
}
```

Output:



The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS** 200 | **22.0ms** | **1.8KB**
- Root** (with copy and download icons)
- JSON** (download icon)
- data** (empty object)
- errors** (array of 1 error, with CSV download icon)
- | | |
|---|-----------------|
| 0 | User not found. |
|---|-----------------|

Remarks: The output is correct. The negative path is handled with an error thrown.

4.

Testcase Identifier: testAddReactionForSameUser

Testcase Description:

When inputting a user ID that has already added a reaction for the comment, the function throws an error.

Inputs:

```
{
  "commentId": "10001",
  "type": "+1",
  "author": "2"
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS 200** | **19.0ms** | **1.8KB**
- Root** (copy icon) **JSON** (download icon)
- data** (copy icon)
- errors** (download icon) **CSV**
- | | |
|---|--|
| 0 | User has already reacted to this comm... |
|---|--|

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: removeReaction

1.

Testcase Identifier: testRemoveReactionWithCorrectInput

Testcase Description:

When inputting an existing reaction ID and the ID of the user that creates the reaction, the function removes the reaction from the reaction database and the comment's reaction list.

Inputs:

```
{  
  "reactionId": 2,  
  "user": "2"  
}
```

Output:

Response		STATUS 200 20.0ms 201B	
Root > data > removeReaction		CSV	JSON
	user.username	type	id
0	user1	thumbsUp	1
1	user3	thumbsUp	3
2	user4	thumbsUp	4

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testRemoveReactionWithInvalidReactionID

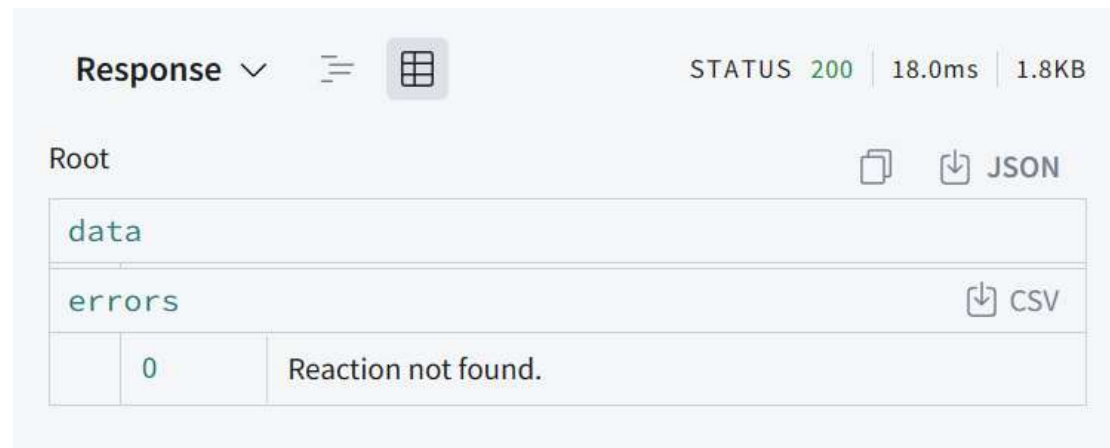
Testcase Description:

When inputting a reaction ID that does not exist in the database, the function throws an error.

Inputs:

```
{  
  "reactionId": "100",  
  "user": "2"  
}
```

Output:



The screenshot shows a REST client interface with a 'Response' tab selected. The status is 200, with a response time of 18.0ms and a size of 1.8KB. The response body is displayed in a table format with two main sections: 'data' and 'errors'. The 'errors' section contains a single entry with the index '0' and the message 'Reaction not found.'.

Root	
data	
errors	
0	Reaction not found.

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testRemoveReactionWithWrongUser

Testcase Description:

When inputting a user ID that does not create the inputted reaction, the function throws an error.

Inputs:

```
{  
  "reactionId": "1",  
  "user": "3"  
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS 200** | **15.0ms** | **1.8KB**
- Root** (with copy and download icons)
- data** (empty object)
- errors** (array of 1 error, with CSV download icon)
- | | |
|---|----------------------------------|
| 0 | User does not own this reaction. |
|---|----------------------------------|

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: mergePullRequest

1.

Testcase Identifier: testMergePullRequestWithCorrectCondition

Testcase Description:

When inputting an id of a pull request which is not merged, not rejected or has no conflict (commit hash starting with '0'), the function changes the pull request's status to be MERGED and returns the pull request.

Inputs:

```
{  
  "prId": "1"  
}
```

Output:

Response

STATUS 200 | 16.0ms | 137B

Root > data > mergePullRequest JSON

id	1
status	MERGED
sourceCommit	12H3J4
targetBranch	main
description	Fix some issues

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testMergePullRequestWithMergedPR

Testcase Description:

When inputting an id of a pull request which is merged, the function throws an error.

Inputs:

```
{  
  "prId": "4"  
}
```

Output:

The screenshot shows a REST client interface with a 'Response' tab selected. The status is 200, with a response time of 35.0ms and a size of 1.8KB. The response body is displayed in a table format with two main sections: 'data' and 'errors'. The 'errors' section contains a single entry with an index of 0 and the message 'Pull request already merged.'.

Root	
data	
errors	
0	Pull request already merged.

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testMergePullRequestWithMergeConflict

Testcase Description:

When inputting an id of a pull request which has a merge conflict (its commit has a hash starting with '0'), the function changes the status of the pull request to be MERGE_CONFLICT, and throws an error.

Inputs:

```
{  
  "prId": "2"  
}
```

Output:

The screenshot shows a REST client interface. At the top, it displays 'Response' with a dropdown arrow, a list icon, and a table icon. To the right, it shows 'STATUS 200', '20.0ms', and '1.8KB'. Below this, the 'Root' of the response is shown with a copy icon and a 'JSON' label. The response body is a JSON object with two main properties: 'data' and 'errors'. The 'data' property is expanded, showing a table with one row containing the value '0'. The 'errors' property is also expanded, showing a table with one row containing the message 'Merge conflict detected. Merge failed.'.

data	
0	Merge conflict detected. Merge failed.

Remarks: The output is correct. The negative path is handled with an error thrown.

4.

Testcase Identifier: testMergePullRequestWithInvalidPRID

Testcase Description:

When inputting an id of a pull request which does not exist in database, the function throws an error.

Inputs:

```
{  
  "prId": "100"  
}
```

Output:

The screenshot shows a REST client interface. At the top, it displays 'Response' with a dropdown arrow, a hamburger menu icon, and a grid icon. To the right, it shows 'STATUS 200', '40.0ms', and '1.8KB'. Below this, the 'Root' of the response is shown with a copy icon and a 'JSON' label. The response body is a JSON object with two fields: 'data' and 'errors'. The 'errors' field is expanded, showing a table with one row containing the index '0' and the message 'Pull request not found.'.

Root	
data	
errors	
0	Pull request not found.

Remarks: The output is correct. The negative path is handled with an error thrown.

5.

Testcase Identifier: testMergePullRequestWithRejectedPR

Testcase Description:

When inputting an id of a pull request which has been rejected (its status is REJECTED), the function throws an error.

Inputs:

```
{  
  "prId": "3"  
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS** 200 | 22.0ms | 1.8KB
- Root** (with copy and download icons)
- data** (expandable section)
- errors** (expandable section with a download icon and label **CSV**)
- | | |
|---|--------------------------------|
| 0 | Pull request already rejected. |
|---|--------------------------------|

Remarks: The output is correct. The negative path is handled with an error thrown.

Mutation: rejectPullRequest

1.

Testcase Identifier: testRejectPullRequestWithCorrectInput

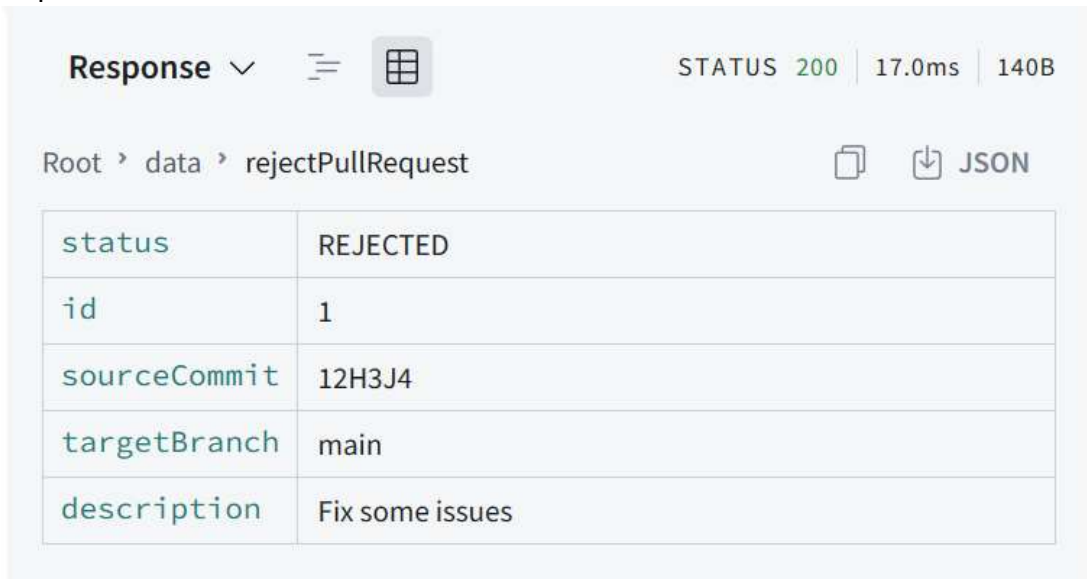
Testcase Description:




When inputting an id of a pull request which has not been rejected or merged, the function changes the status of the pull request to be REJECTED and returns the pull request.



Inputs:

```
{  
  "prId": "1"  
}
```

Output:



Response    STATUS 200 | 17.0ms | 140B

Root > data > rejectPullRequest   JSON

status	REJECTED
id	1
sourceCommit	12H3J4
targetBranch	main
description	Fix some issues

Remarks: The output is correct. The happy path is tested to be successful.

2.

Testcase Identifier: testRejectPullRequestWithInvalidPRID

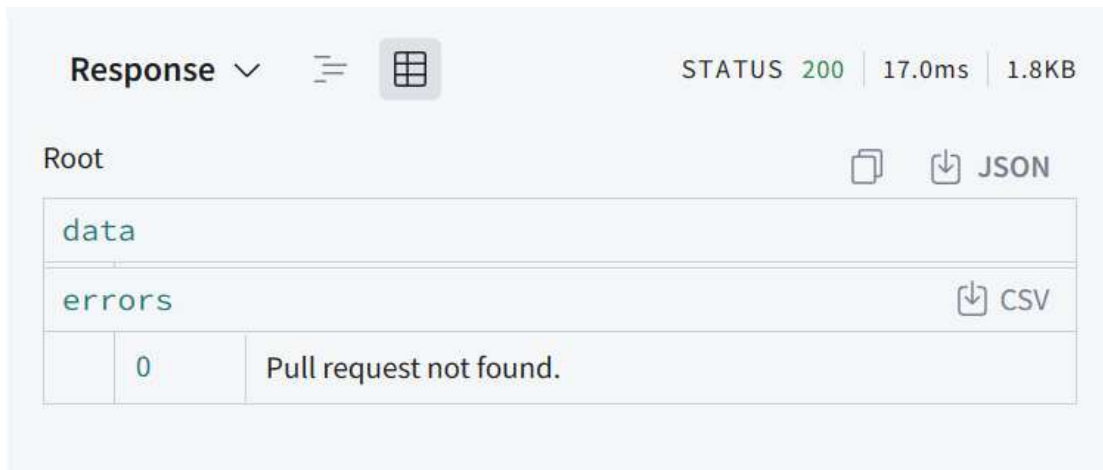
Testcase Description:

When inputting an id of a pull request which does not exist in database, the function throws an error.

Inputs:

```
{  
  "prId": "100"  
}
```

Output:



The screenshot shows a REST client interface with a 'Response' tab selected. The status is 200, with a response time of 17.0ms and a size of 1.8KB. The response body is displayed in a table format with two main sections: 'data' and 'errors'. The 'errors' section contains a single entry with an index of 0 and the message 'Pull request not found.'.

Root		
data		
errors		
	0	Pull request not found.

Remarks: The output is correct. The negative path is handled with an error thrown.

3.

Testcase Identifier: testRejectPullRequestWithMergedPR

Testcase Description:

When inputting an id of a pull request which has been merged (its status is MERGED), the function throws an error.

Inputs:

```
{  
  "prId": "4"  
}
```

Output:

Response ▾

≡

📊

STATUS 200 | 23.0ms | 1.8KB

Root

📄

📄 JSON

data		
errors <div>📄 CSV</div>		
	0	Pull request already merged.

Remarks: The output is correct. The negative path is handled with an error thrown.

4.

Testcase Identifier: testRejectPullRequestWithRejectedPR

Testcase Description:

When inputting an id of a pull request which has been rejected (its status is REJECTED), the function throws an error.

Inputs:

```
{  
  "prId": "3"  
}
```

Output:

The screenshot shows a REST client interface with the following details:

- Response** (dropdown menu)
- STATUS 200** (green text)
- 12.0ms** (response time)
- 1.8KB** (response size)
- Root** (breadcrumb)
- data** (JSON key)
- errors** (JSON key)
- 0** (error count)
- Pull request already rejected.** (error message)
- JSON** (download button)
- CSV** (download button)

errors
0
Pull request already rejected.

Remarks: The output is correct. The negative path is handled with an error thrown.