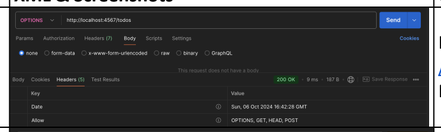
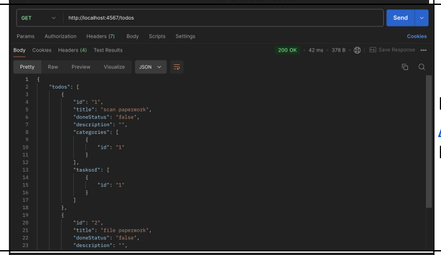
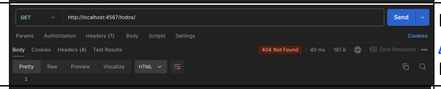
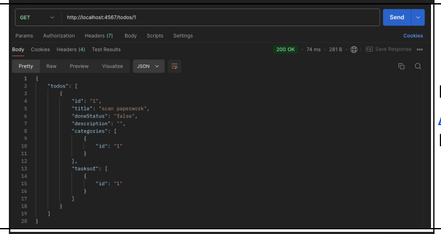
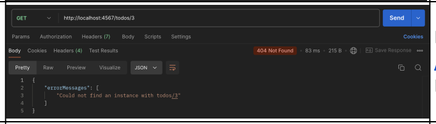
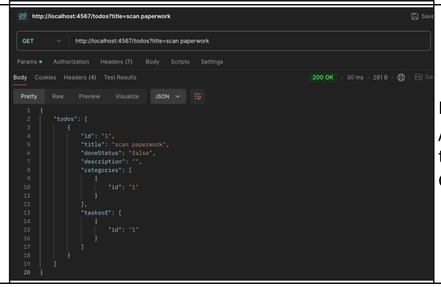
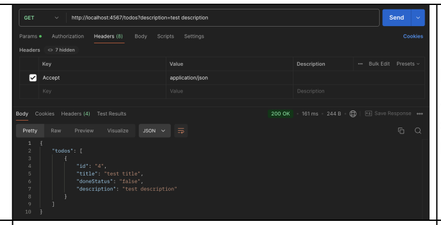
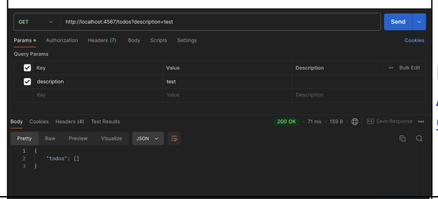
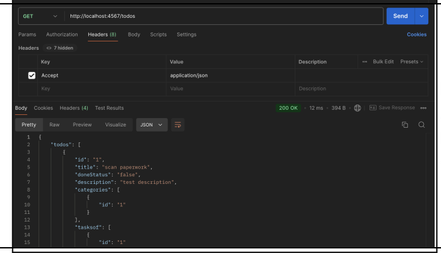
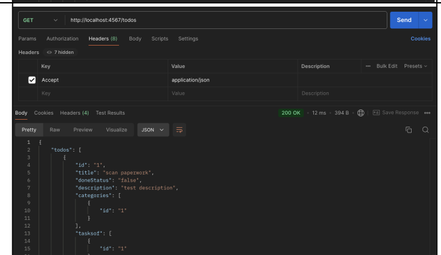
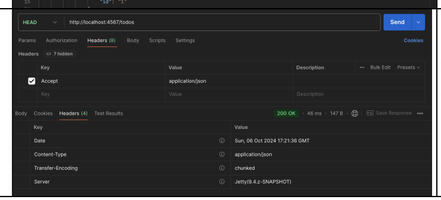
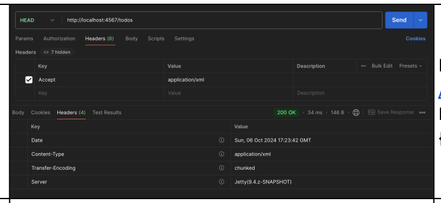
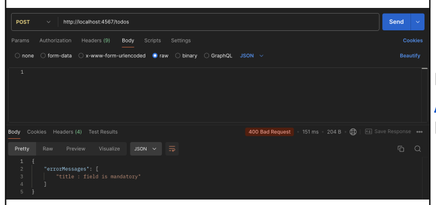
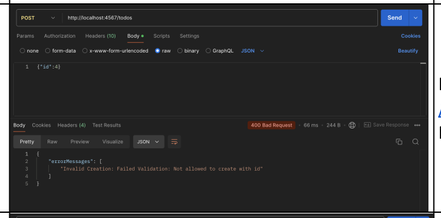
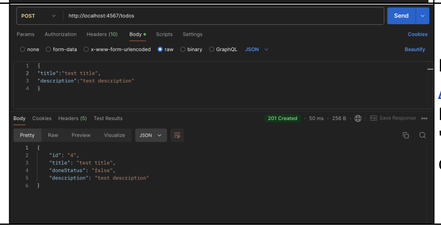
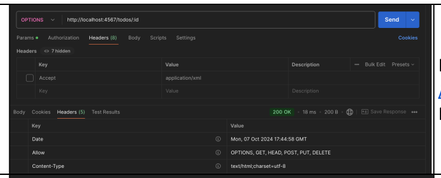
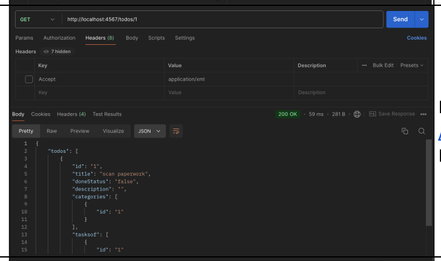
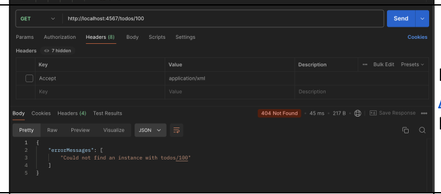
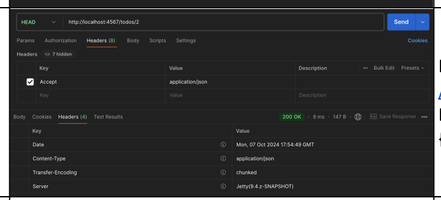
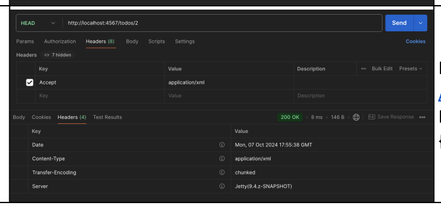
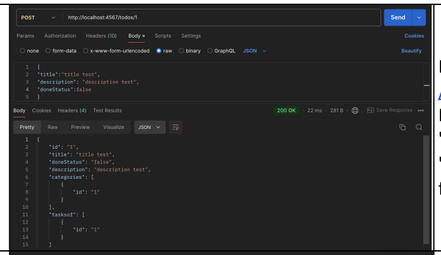
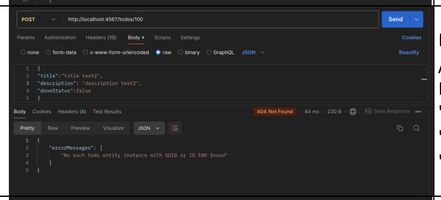
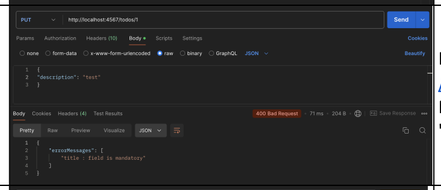
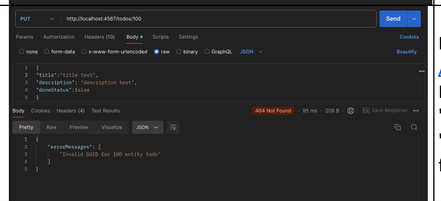
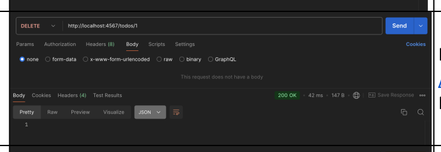
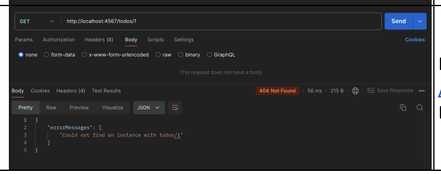
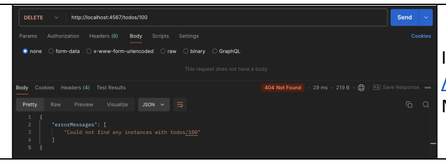
Method	Method Type	is documented?	Expected Behavior	expected == observed	Observed Behavior	Side Notes	XML & Screenshots	Script (Windows PowerShell)
/todos	OPTIONS	UNDOCUMENTED	Return the list of options (OPTIONS, GET, HEAD, POST)	YES	Received the list of options (OPTIONS, GET, HEAD, POST)	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method OPTIONS
/todos	GET	DOCUMENTED	Return all todos	YES	Received all todos instances	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method GET
/todos/	GET	UNDOCUMENTED	not allowed	YES	404 Not Found Error	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/" -Method GET
/todos/1	GET	DOCUMENTED	Return the todo based on id	YES	Received the todo with ID 1	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method GET
/todos/3	GET	UNDOCUMENTED	not allowed	YES	404 Not Found Error	Tried getting a todo with a nonexistent id		Invoke-WebRequest -Uri "http://localhost:4567/todos/3" -Method GET
/todo?title=scan paperwork	GET	DOCUMENTED	Return the todo based on title	YES	Received the todo based on title	-		Invoke-WebRequest -Uri "http://localhost:4567/todos?title=scan paperwork" -Method GET

/todos?description=test description	GET	DOCUMENTED	Return the todo based on description	YES	Received the todo based on description	-		Invoke-WebRequest -Uri "http://localhost:4567/todo?description=test description" -Method GET
/todos?description=test	GET	UNDOCUMENTED	Return the todo based on partial description	NO	Received no todo	Would make more sense to filter description based on inclusion as opposed to complete correspondence of the description field		Invoke-WebRequest -Uri "http://localhost:4567/todo?description=test" -Method GET
/todos	GET	DOCUMENTED	Return all todos in JSON	YES	Received the todos in JSON	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method GET ` -Headers @{"Accept"="application/json"}
/todos	GET	DOCUMENTED	Return all todos in XML	YES	Received the todos in XML	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method GET ` -Headers @{"Accept"="application/xml"}
/todos	HEAD	DOCUMENTED	Return the todos headers in JSON	YES	Received the headers in JSON	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method HEAD ` -Headers @{"Accept"="application/json"}

/todos	HEAD	DOCUMENTED	Return the todos headers in XML	YES	Received the headers in XML	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method HEAD ` -Headers @{"Accept"="application/xml"}
/todos	POST	DOCUMENTED	Post an empty todo	YES	Did not post an empty todo. Gave the following error message: "title: field is mandatory"	The error message, however, indicates that the empty todo was not created because the title field is mandatory and missing. It seems that if the title field was not mandatory, the empty todo would have been created which would have been a bug. Also, ID increases even though post fails (newly created ID will be 2 higher than previous object		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method POST ` -Body '{}'
/todos	POST	DOCUMENTED	Fail to post a todo with an id specified	YES	Failed to post a todo	Tried posting a todo with an id		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method POST ` -Body '{"id": 4}'
/todos	POST	DOCUMENTED	Post a todo instance with title and description	YES	Posted a todo instance with title and description with JSON	-		Invoke-WebRequest -Uri "http://localhost:4567/todos" -Method POST ` -Body '{"title": "test title", "description": "test description"}'

/todos/:id	OPTIONS	UNDOCUMENTED	Return the list of options (OPTIONS, GET, HEAD, POST, PUT, DELETE)	YES	Received the list of options (OPTIONS, GET, HEAD, POST, PUT, DELETE)	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/:id" -Method OPTIONS
/todos/1	GET	DOCUMENTED	Get a todo with a specific id	YES	Got a todo with a specific id	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method GET
/todos/100	GET	DOCUMENTED	Fail to get a todo with a nonexisting id	YES	Failed to get a todo with a nonexisting id (404 NOT FOUND)	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/100" -Method GET
/todos/2	HEAD	DOCUMENTED	Receive the headers in JSON	YES	Received the headers in JSON	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/2" -Method HEAD ` -Headers @ {"Accept"="application/json"}
/todos/2	HEAD	DOCUMENTED	Receive the headers in XML	YES	Received the headers in XML	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/2" -Method HEAD ` -Headers @ {"Accept"="application/xml"}

/todos/1	POST	DOCUMENTED	Modify a todo with an existing id	YES	Modified a todo with an existing id	PATCH should be used for amending instances instead		Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method POST ` -Body '{"title": "title test", "description": "description test", "doneStatus": false}'
/todos/100	POST	DOCUMENTED	Fail to modify a todo with a nonexistent id	YES	Failed to modify a todo with a nonexistent id (404 NOT FOUND)	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/100" -Method POST ` -Body '{"title": "title test2", "description": "description test2", "doneStatus": false}'
/todos/1	PUT	DOCUMENTED	Modify a todo with an existing id and modified description	NO	Returned error message "title field is mandatory			Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method PUT ` -Body '{ "description": "test"}'
/todos/100	PUT	DOCUMENTED	Fail to modify a todo with a nonexistent id	YES	Failed to modify a todo with a nonexistent id (404 NOT FOUND)	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/100" -Method PUT ` -Body '{"title": "title test", "description": "description test", "doneStatus": false}'
/todos/1	DELETE	DOCUMENTED	Delete a todo with a specific id	YES	Deleted a todo with a specific id	- No confirmation message, just returns 200 status code		Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method DELETE
/todos/1	GET	DOCUMENTED	Fail to get a deleted todo	YES	Failed to get a deleted todo	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/1" -Method GET

/todos/100	DELETE	DOCUMENTED	Fail to delete a todo with a nonexistent id	YES	Failed to delete a todo with a nonexistent id	-		Invoke-WebRequest -Uri "http://localhost:4567/todos/100" -Method GET

Name	Mohammed Elsayed
ID	261053266

Charter	*Objective:* Identify the capabilities and areas of potential instability of the REST API endpoints related to todo management (/todos and /todos/:id). Focus on both documented and undocumented capabilities, and demonstrate each capability through small scripts or programs. Ensure that all capabilities are tested with data typical to the intended use of the application. *Scope:* Endpoints to be tested: /todos and /todos/:id *Testing will cover the following aspects:* - Create Todos: Valid and invalid payloads for todos creation. - Retrieve Todos: Fetch all todos and specific todos using their IDs. - Update Todos: Modify existing todos, checking the impact of valid and invalid inputs. - Delete Todos: Remove todos and observe system responses to edge cases (e.g., deleting nonexistent todos). - Error Handling: Verify proper error messages for common failure scenarios such as invalid todo IDs, missing fields, or unauthorized access. *Deliverables:* - For each identified capability, create scripts or small programs to: - Demonstrate the Capability: Show how each feature functions, including successful and failed operations. - Test with Typical Data: Use typical todo data to test each capability. - Capture Instabilities: Identify and log any instabilities, inconsistencies, or undocumented behavior. Testing Considerations: - Test with valid and invalid data inputs. - Explore edge cases, such as sending unexpected payloads or using invalid identifiers. - Document any undocumented API behaviors.
	OPTIONS: Successfully validated documented options for both /todos and /todos/:id. The provided options appear to be exhaustive. GET: - Successfully retrieved todos by ID, title, description, and doneStatus. - Filtering todos by a word from the description does not work as expected. The full description is required for filtering (raises a question about filtering by inclusion). - Successfully retrieved todo data in both JSON and XML formats. HEAD: Received expected headers in both JSON and XML formats for all tested requests.

Summary of session findings	POST: - Handled valid input. It also correctly handled invalid input (missing title field, by raising an error and sending a 400 status. It also successfully created an instance of todo when a title field was provided. - Posted the object with the existing id, while modifying the fields that were passed. It also handled invalid input correctly. The API also correctly raised an error and sent a 404 status when given an id with no associated todos. PUT: Replaced entire todo objects by specifying partial fields; fields not provided defaulted to initial values. This behavior is not documented. Posting with nonexistent IDs correctly failed, as expected. DELETE: - Successfully deleted todos by ID. - Attempted retrieval after deletion correctly failed.
Summary of concerns	- Filtering by Description: Filtering todos by partial description is not supported. This raises the question of whether inclusion-based filtering should be implemented, as it would provide a more user-friendly experience. - POST/PUT Behavior: While attempting to POST or PUT with an existing or nonexistent ID behaves as expected, the behavior of PUT (completely replacing the todo) is not documented, especially the resetting of unspecified fields to default values. This may cause unexpected data loss and should be documented or handled differently. - Posting Empty Todos: The error message "title: field mandatory", indicates that the empty todo was not created because the title field is mandatory and missing. It seems that if the title field was not mandatory, the empty todo would have been created which would have been a bug. - ID increases even though post fails (newly created ID will be 2 higher than previous object instead of 1). This may lead to data loss and loss of data integrity. App can be shutdown by anyone as it is a simple get request without any authentication or validation

Test ideas	- Inclusion-Based Filtering: Implement tests to explore more flexible filtering methods, such as filtering by partial descriptions, to confirm if this feature can be supported or if it requires future development. - PUT/POST Validation: Write test cases to explore different use cases for PUT and POST, focusing on documenting and clarifying the default behavior of unspecified fields. Consider testing PATCH as a more suitable alternative for partial updates. - Test with IDs that are negative