Test case documentation for backend-testing with Postman, later to be implemented in cypress

Test case	Description	Step	Test step	Test data	Expected result	Automated
Prerequisite	*Login and get token*	1	Start Tester Hotel application			N/A
Prerequisite	*Login and get token*	2	Open Postman			N/A
Prerequisite	*Login and get token*	3	POST towards /api/login	{"username":"tester01","password":"GteteqbQQgSr88SwNExUQv2ydb7xuf8c"}	- Status: 2xx - Token created	As Custom Command function ('authenticateSession()')
Prerequisite	*Login and get token*	4	Copy token value and go to postman collection, Paste token value as session_token value	Token created in previous step	No need for individual headers ('X-User-Auth') for new requests during active login session	
Prerequisite	*Login and get token*	5	Save			N/A
TC1	View all rooms	1	Send a GET-request towards /api/rooms		- Status: 2xx - There are two rooms and their data match constant variables in data.js	Yes
TC2	View all clients	1	Send a GET-request towards /api/clients		- Status: 2xx - There are two clients and their data match constant variables in data.js	Yes
TC3	View all bills	1	Send a GET-request towards /api/bills		- Status: 2xx - There is one bill and data match constant variables in data.js	Yes
TC4	View all reservations	1	Send a GET-request towards /api/reservations		- Status: 2xx - There is one reservation and data match constant variables in data.js	Yes

TC5	Add feature to room no.1	1	Send a PUT-request towards /api/room/1	{
				"id": 1,
				"created": "2020-01-03T12:00:00.000Z",
				"category": "double",
				"floor": 1,
				"number": 101,
				"available": true,
				"price": 1500,
				"features": [
				"balcony", "ensuite", "sea_view"
				]
				}
					- Status: 2xx
					- Room properties are updated
						Yes, with some adjustments (creates a new room first to safely change, then deletes it)

TC5	Add feature to room no.1	2	Send a GET-request towards /api/rooms		- Status: 2xx - Room no.1 now has sea_view	

TC6	Create a new room	1	Send a POST-request towards /api/room/new	{
				"category": "single",
				"floor": 2,
				"number": 201,
				"available": true,
				"price": 1500,
				"features": [
				"balcony", "ensuite", "sea_view"
				]
				}
					- Status: 2xx
					- Room properties shown with keys id and created
						Yes, with the adjustment that it deletes the room in the end to make the TC independent

TC6	Create a new room	2	Send a GET-request towards /api/rooms		- Status: 2xx - The room created in previous step is there	
TC7	Update first reservation	1	Send a GET-request towards /api/reservation/:id	:id=1	- Status: 2xx - Reservation object shown	Yes, with some adjustments (changes the object back to original state)
TC7	Update first reservation	2	Copy object data and paste as body in a PUT-request towards /api/reservation/:id Change end-date before sending	:id=1	- Status: 2xx - Reservation object shown with new end-date	
TC8	Delete first reservation	1	Send a DELETE-request towards /api/reservation/:id	:id=1	- Status: 2xx	Yes, with some adjustments (creates a new object first to delete safely)
TC8	Delete first reservation	2	Send a GET-request towards /api/reservations		- Status: 2xx - Reservation no. 1 is removed	