Backend User API Test Cases

Test Case #	Action	Expected Result	Actual Result	Result (Pass/Fail)
1.	Create a user using valid input: Open Postman and send a POST request to http://localhost:8080/api/users with the body set to raw and formatted as JSON: { "name": "John Snow", "email": "john@example.com"} Then click Send	Verify the response includes an auto-generated userID, correct name and email, and an auto-generated createdAt	All fields returned: userID, name, email, and createdAt	Pass
2.	Create a second user: Send another POST request to the same URL with a different user	Verify the new user is returned with a different userID and a different createdAt	Second user is created with a different userID and createdAt	Pass
3.	Allow users with the same name: Send two POST requests with identical name but different email addresses	Verify both users are created with unique userIDs and different createdAt values	Both users are created with the same name, different email, userIDs and createdAt	Pass
4.	Check and confirm userID and createdAt progression: Create multiple users and compare their returned values.	Verify all users have different userIDs and unique createdAt	Each user has a different userID and createdAt	Pass
5.	Fetch all users: Send a GET request to the same URL	Verify the response contains a list of users, each with a unique userID, name, email, and createdAt	A list of users with the correct fields is returned	Pass
6.	Fetch a single user by ID: Use a valid userID from a previous response and	Verify the response shows the correct user data including userID,	Correct user details are returned by ID	Pass

	send a GET request to http://localhost:8080/api/users/{id}	name, email, and createdAt		
7.	Send a PUT request to http://localhost:8080/api/users with any body	Verify the response returns the string “Tranactions”	“Tranactions” returned	Pass
8.	Send a DELETE request to http://localhost:8080/api/users	Verify the response returns the string “Add Transaction”	“Add Transaction” returned.	Pass
9.	Create a transaction using valid input: Open Postman and send a POST request to http://localhost:8080/api/transactions with the body set to raw and formatted as JSON: { "userID": "5", "amount": "400","transactionType": "expense","dateHelper": "2025-04-10","category": "bill","description": "water bill"} Then click Send	Verify the response includes an auto-generated transactionID, correct userID, amount, dateHelper,category, and description, and an auto-generated createdAt	All fields returned: transactionID, userID, amount, dateHelper, category, description, and createdAt	Pass
10.	Create a transaction with missing or invalid required fields(“amount” or “userID”)	Verify the response returns 400 bad request error	400 bad request error	Pass
11.	Fetch all transactions: Open Postman and send a GET request to http://localhost:8080/api/transactions	Verify the response contains a list of all transactions each with a unique transactionID, userID, amount, dateHelper, category, description, and createdAt.	A list of transactions with the correct fields is returned	Pass
12.	Get a transaction by single transactionID: http://localhost:8080/api/transaction?id=4 (transactionID)	Verify the response shows the correct transaction data including all the details.	Correct transaction details are returned by transactionID	Pass
13.	Get a transaction by	Verify the response	500 Internal	Pass

	nonexistent transactionID: http://localhost:8080/api/transaction?id={non existent transactionID}	returns 500 Internal Server Error	Server Error	
14.	Delete an existing transaction: use a valid transactionID from a previous response and send a DELETE request to http://localhost:8080/api/transaction?id={transactionID)	Confirm the transaction is deleted by sending a followup GET request for the same ID.	The transaction is deleted and the follow up GET request returns 500 Internal Server Error	Pass