Regression Test Case Report For Hummingbot

Table of Contents

Hummingbot gateway

a. Issues Found During Integration of cardano, minswap and sundaeswap

Issue ID	Description	Component Affected
001	Required a library to integrate Cardano; selected lucid-cardano	Cardano Integration
002	Gateway uses CommonJS; Lucid and Minswap SDK are ESM-only. Required Babel transpilation and custom module copy	Build/Module Compatibility
003	cardano.yml lacked token list and schema updates. Caused schema validation errors	API Configuration
004	No Cardano chain support; created chain/cardano.ts and secure key storage logic	Blockchain Configuration
005	Required token balance from UTXOs; needed to understand asset ID composition	Balance API

Issue ID	Description	Component Affected
006	Aggregated UTXO quantities for a specific asset to calculate token balance	Balance Aggregation
007	Needed transaction polling; used Blockfrost API to fetch transaction details by hash	Transaction Polling
008	Required Minswap liquidity pool details; referred to SDK GitHub documentation	Liquidity Pool Query
009	Calculated ADA/Min and Min/ADA price using Minswap SDK functions	Pool Pricing
010	Implemented buy/sell logic for trades based on side input (base/quote logic)	Swap API
011	Calculated LP tokens for liquidity add using reserve ratio formulas	Liquidity Add Logic
012	Implemented liquidity removal logic using LP token burn formula	Liquidity Remove Logic
013	Refactored Lucid instance to support multiple DEXes; added SundaeSwap tokens	Multi-DEX Support
014	SBERRY used instead of SUNDAE on preview network; clarified with SundaeSwap team	Network Token Mapping
015	Ensured wallet is connected before creating txBuilderLucid for swaps	Wallet Session Management
016	Calculated required ADA amount when user specified only base asset amount in a buy	ADA Quote Calculation
017	JavaScript Math.pow doesn’t support BigInt; monkey patched to allow it	BigInt Math Handling

b. Fixes Implemented

Fix ID	Description	Linked Issue(s)	Commit/Branch
F001	Integrated lucid-cardano and set up transaction/query pipeline	001	feature/cardano-lucid
F002	Transpiled ESM modules using Babel and copied to node_modules	002	fix/esm-cjs-compat
F003	Added token list and updated cardano.schema.json to fix schema errors	003	fix/cardano-schema
F004	Created full Cardano chain config in chain/cardano.ts; implemented secure key storage	004	feature/cardano-chain-support
F005	Parsed UTXO assets and calculated token balances using policyId+assetName logic	005,006	feature/balance-api
F006	Implemented transaction polling using Blockfrost API	007	feature/tx-status
F007	Queried Minswap pool data using SDK; exposed pricing via API	008,009	feature/minswap-pricing
F008	Enabled swap trading via side-aware API	010	feature/swap-trade

Fix ID	Description	Linked Issue(s)	Commit/Branch
	endpoints		
F009	Added liquidity logic with correct LP token calculations	011	feature/add-liquidity
F010	Enabled liquidity removal using burn formulas provided by minswap sdk	012	feature/remove-liquidity
F011	Supported both Minswap and SundaeSwap via dynamic Lucid instances	013	refactor/multi-dex
F012	Mapped SBERRY to testnet; SUNDAE to mainnet for correct behavior	014	config/token-aliases
F013	Added wallet checks before transaction build to prevent failures	015	fix/lucid-wallet-check
F014	Implemented ADA calculation logic for swap estimation	016	feature/buy-price-quote
F015	Monkey patched Math.pow to support BigInt token decimal math	017	fix/bigint-math

Hummingbot client

a. Issues Found During Hummingbot Client Development

Issue ID	Description	Component Affected
C001	Didn’t know which file invokes all Gateway API calls	gateway_http_client.py discovery
C002	Native Cardano token not specified in client config	gateway_config_utils.py
C003	Gateway connection configs location unknown	conf/gateway-connections.json
C004	Custom scripts didn’t show up under create –script-config	Needed to inherit BaseClientModel & ScriptStrategyBase
C005	“Market doesn’t exist” error when running scripts	Missing gateway_cardano_amm.py connector file
C006	No Cardano AMM connector class to handle order/balance/API	Needed GatewayCardanoAMM extending ConnectorBase
C007	No script to fetch pool price	Add price-fetch scripts under conf/scripts
C008	No script to perform pool trades (buy/sell)	Add trade scripts under conf/scripts
C009	No liquidity-management scripts	Add add/remove-liquidity bots under conf/scripts
C010	Cardano doesn’t use allowance/approval because tokens are linked to utxos	Needed override to set allowance = Decimal("0") for Cardano

b. Fixes Implemented

Fix ID	Description	Linked Issue(s)	Commit/Branch
CF001	Mapped out and documented gateway_http_client.py entry points	C001	fix/client-entrypoints
CF002	Updated gateway_config_utils.py to include Cardano native-token field	C002	feature/cardano-config
CF003	Confirmed and documented conf/gateway-connections.json usage	C003	docs/config-paths
CF004	Enforced script models to inherit BaseClientModel & ScriptStrategyBase	C004	fix/script-registration
CF005	Created gateway_cardano_amm.py with full AMM connector implementation	C005, C006	feature/cardano-amm
CF006	Added price-fetch, trade, and liquidity scripts under conf/scripts	C007, C008, C009	feature/client-scripts
CF007	Overrode allowance logic to return zero for Cardano chains	C010	fix/cardano-allowance

Testing Results

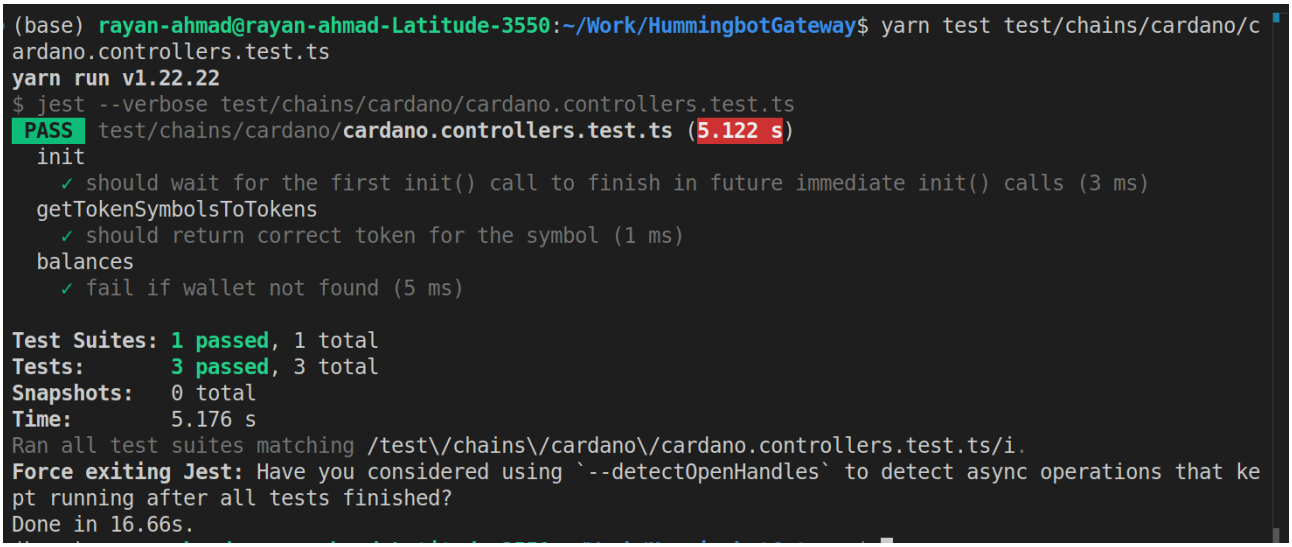
cardano.controller.test.ts

Illustration

Test No.	Test Name / User Story	Conditions	Steps	Expected Output	Accepted Output	Result
1	init: ensure subsequent init() calls wait on the first call	First init() call pending completion	1. Call cardano.init() and attach .then() to set a flag when fulfilled.2. await the first call.3. Call cardano.init() a second time.4. Check that the flag is true.	Flag should be set to true (first call fulfilled before second)	firstCallFulfilled === true	✔️
2	getTokenSymbolsToTokens: map “MIN”	Patched cardano.getToken	1. Use patch(cardano, 'getTokenForSymbol',	Returns the exact min: CardanoTokenInfo	cardano.getTokenForSymbol('MIN') strictly equals min	✔️

	symbol to token info	ForSymbol to return the min object	() => min);2. Call cardano.getTokenForSymbol('MIN').3. Compare the returned value to the min constant.	object		
3	balances: error thrown when wallet not found	Patched cardano.getWalletFromAddress to throw an HttpException with code/message from constants	1. Patch getWalletFromAddress to throw new HttpException(500, LOAD_WALLET_ERROR_MESSAGE + 'Error: wallet does not exist', LOAD_WALLET_ERROR_CODE).2. Call CardanoController.balances(...) with that address.3. await expect(...).rejects.toThrow(...).	The promise rejects with the same HttpException instance and message	The call to CardanoController.balances(...) rejects with an HttpException(500, LOAD_WALLET_ERROR_MESSAGE + 'Error: wallet does not exist', LOAD_WALLET_ERROR_CODE)	✔️

Screenshots

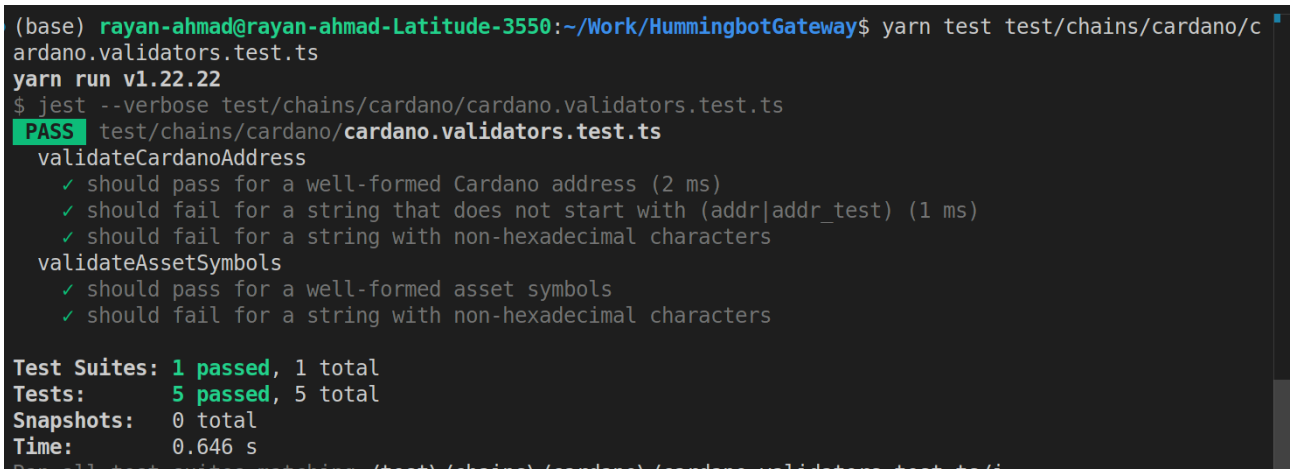


cardano.validators.test.ts

Illustration

Test No.	Test Case	Conditions	Steps	Expected Output	Accepted Output	Result
1	validateCardanoAddress: pass for a well-formed Cardano address	req.address = 'addr_test1vznd34ydghfh2aw8cnn5lgw90vpvlg82ngj30wzue0rw5jgct5m7d'	Call validateCardanoAddress(req)	[]	[]	✔
2	validateCardanoAddress: fail for string not starting with addr or addr_test	req.address = '0xFaA12FD102FE8623C9299c72'	Call validateCardanoAddress(req)	['Invalid Cardano address.']	['Invalid Cardano address.']	✔
3	validateCardanoAddress: fail for string with non-hexadecimal characters	req.address = 'addr_pqrst'	Call validateCardanoAddress(req)	['Invalid Cardano address.']	['Invalid Cardano address.']	✔
4	validateAssetSymbols: pass for a well-formed asset symbols	req.symbols = ['ADA', 'MIN']	Call validateAssetSymbols(req)	[]	[]	✔
5	validateAssetSymbols: fail for string with non-hexadecimal characters in symbols list	req.symbols = ['ADA', '']	Call validateAssetSymbols(req)	['Invalid asset symbol.'] (or [])	['Invalid asset symbol.'] (or [])	✔

Screenshots



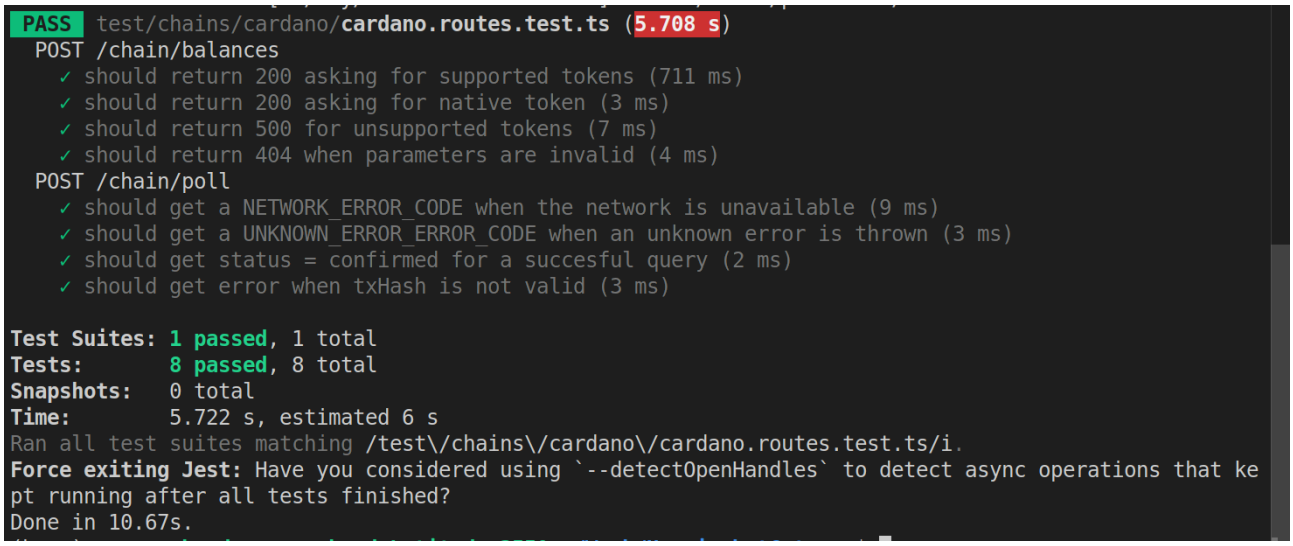
cardano.routes.test.ts

Illustration

Test No.	Endpoint	Test Case	Conditions & Mocks	Steps	Expected Output	Accepted Output	Result
1	POST /chain/balances	should return 200 asking for supported tokens	Wallet, token-for-symbol (ADA, MIN), and native-balance patched	• POST body with token Symbols: ['ADA','MIN']• Expect JSON, status 200	res.body.balances.ADA and .MIN are defined	res.body.balances.ADA and .MIN are defined	✔️
2	POST /chain/balances	should return 200 asking for native token	Wallet, token-for-symbol (ADA), and native-balance patched	• POST body with token Symbols: ['ADA']• Expect JSON, status 200	res.body.balances.ADA is defined	res.body.balances.ADA is defined	✔️

3	POST /chain/balances	should return 500 for unsupported tokens	Wallet, token-for-symbol patched (XXX, YYY fall through to unsupported)	• POST body with token Symbols: ['XXX','YYY']• Expect JSON, status 500	HTTP 500 error (unsupported tokens)	HTTP 500 error	✔️
4	POST /chain/balances	should return 404 when parameters are invalid	No tokenSymbols in request	• POST body missing token Symbols• Expect status 404	HTTP 404 error (invalid parameters)	HTTP 404 error	✔️
5	POST /chain/poll	should get a NETWORK_ERROR_CODE when the network is unavailable	No patches; Cardano.getTransaction calls network endpoint and fails	• POST body { chain: 'cardano', network: 'testnet', txHash }• Expect status 503	res.body.errorCode === NETWORK_ERROR_CODEres.body.message === NETWORK_ERROR_MESSAGE	As above	✔️
6	POST /chain/poll	should get an UNKNOWN_ERROR_ERROR_CODE on unknown exception	Patched cardano.getTransaction to throw generic Error()	• POST body with a txHash (no chain/network)	status 503res.body.errorCode === UNKNOWN_ERROR_ERROR_CODE	status 503res.body.errorCode === UNKNOWN_ERROR_ERROR_CODE	✔️
7	POST /chain/poll	should get status = confirmed for a successful query	Patched cardano.getTransaction to return transactionSuccesful.json	• POST body { chain: 'cardano', network: 'preprod', txHash }• Expect status 200	res.body.block and res.body.blockHeight are defined	res.body.block and res.body.blockHeight are defined	✔️
8	POST /chain/poll	should get error when txHash is not valid	No patches; invalid txHash: 'abcd'	• POST body { chain: 'cardano', network: 'preprod', txHash: 'abcd' }• Expect status 500	res.body.errorCode === NETWORK_ERROR_CODE	res.body.errorCode === NETWORK_ERROR_CODE	✔️

Screenshot



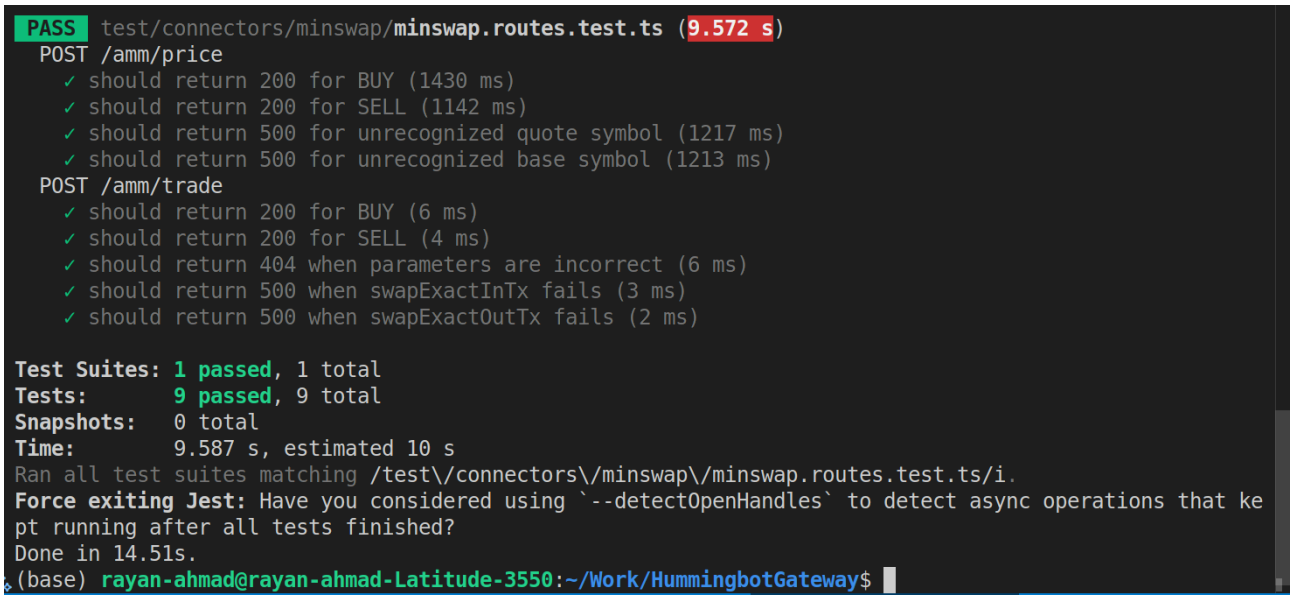
minswap.routes.test.ts

Illustration

Test No.	Endpoint	Test Case	Conditions & Mocks	Steps	Expected Output	Accepted Output	Result
1	POST /amm/price	should return 200 for BUY	patchGetWallet(), patchInit(), patchStoredTokenList(), patchGetTokenBySymbol()	• POST /amm/price with {..., side: 'BUY', amount: '100'}	HTTP 200; response has amount & rawAmount	HTTP 200; res.body.amount & res.body.rawAmount defined	✔️
2	POST /amm/price	should return 200 for SELL	same as Test 1	• POST /amm/price with {..., side: 'SELL', amount: '100'}	HTTP 200; response has amount & rawAmount	HTTP 200; res.body.amount & res.body.rawAmount defined	✔️
3	POST /amm/price	should return 500 for unrecognized quote symbol	same as Test 1	• POST /amm/price with { quote: 'DOGE', ... }	HTTP 500 error (quote symbol not in token list)	HTTP 500	✔️
4	POST	should	same as Test	• POST	HTTP 500	HTTP 500	✔️

	/amm/price	return 500 for unrecognized base symbol	1	/amm/price with { base: 'SHIBA', … }	error (base symbol not in token list)		
5	POST /amm/trade	should return 200 for BUY	patchTrade()	• POST /amm/trade with { side: 'BUY', … }	HTTP 200; response has txHash = 'FAKE_HASH'	HTTP 200; res.body.txHash === 'FAKE_HASH'	✔️
6	POST /amm/trade	should return 200 for SELL	patchTrade()	• POST /amm/trade with { side: 'SELL', … }	HTTP 200; response has txHash = 'FAKE_HASH'	HTTP 200; res.body.txHash === 'FAKE_HASH'	✔️
7	POST /amm/trade	should return 404 when parameters are incorrect	no controller stub	• POST /amm/trade with malformed params (side: 'comprar', bad address)	HTTP 404 error (invalid parameters)	HTTP 404	✔️
8	POST /amm/trade	should return 500 when swapExactInTx fails	patch(MinswapController, 'trade', async () => { throw new Error('boom') })	• POST /amm/trade with { side: 'SELL', … } (trade stub throws)	HTTP 500 error	HTTP 500	✔️
9	POST /amm/trade	should return 500 when swapExactOutTx fails	same stub as Test 8 (throws)	• POST /amm/trade with { side: 'BUY', … } (trade stub throws)	HTTP 500 error	HTTP 500	✔️

Screenshot:



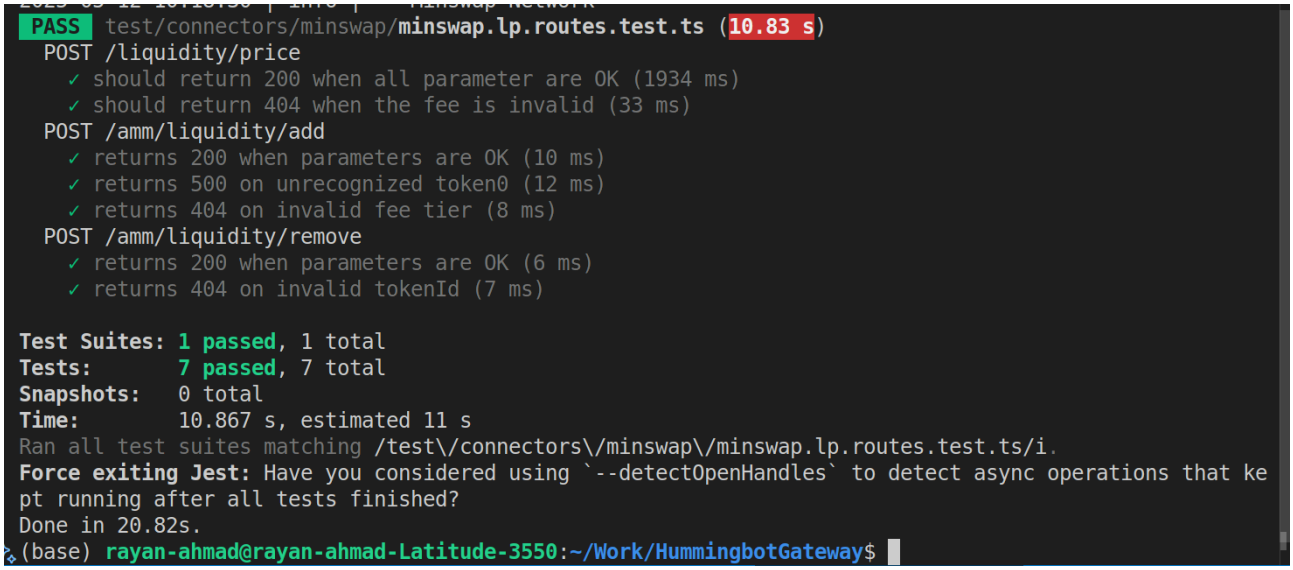
minswap.lp.routes.test.ts

Illustration

Test No.	Endpoint	Test Case	Conditions & Mocks	Steps	Expected Output	Accepted Output	Result
1	POST /amm/liquidity/price	should return 200 when all parameters are OK	patchInit(), patchStoredTokenList(), patchGetTokenBySymbol(), patch(minswap, 'poolPrice', () => ['100','105'])	• POST /amm/liquidity/price with valid token0, token1, fee: 'LOW', period, interval	HTTP 200	HTTP 200	✔️
2	POST /amm/liquidity/price	should return 404 when the fee is invalid	patchGetWallet(), patchInit(), patchStoredTokenList(), patchGetTokenBySymbol()	• POST /amm/liquidity/price with fee: 11 (invalid fee type)	HTTP 404	HTTP 404	✔️
3	POST /amm/liquidity/add	returns 200 when parameters are OK	Stubbed MinswapController.addLiquidity to return fake	• POST /amm/liquidity/add with valid address, token0,	HTTP 200; res.body.txHash === 'FAKE_ADD_	HTTP 200; res.body.txHash === 'FAKE_ADD_	✔️

			TX (FAKE_ADD_HASH)	token1, amount0, amount1, fee: 'LOW'	HASH'	HASH'	
4	POST /amm/liquidity/add	returns 500 on unrecognized token0	Stubbed addLiquidity to throw new Error('bad token')	• POST /amm/liquidity/add with token0: 'DOGE'	HTTP 500	HTTP 500	✔️
5	POST /amm/liquidity/add	returns 404 on invalid fee tier	Default stubbed addLiquidity; fee-validation in controller active	• POST /amm/liquidity/add with fee: 'INVALID_FEE'	HTTP 404	HTTP 404	✔️
6	POST /amm/liquidity/remove	returns 200 when parameters are OK	Stubbed MinswapController.removeLiquidity to return fake TX (FAKE_REMOVE_HASH)	• POST /amm/liquidity/remove with valid tokenId: 0, decreasePercent: 50	HTTP 200; res.body.txHash === 'FAKE_REMOVE_HASH'	HTTP 200; res.body.txHash === 'FAKE_REMOVE_HASH'	✔️
7	POST /amm/liquidity/remove	returns 404 on invalid tokenId	Default stubbed removeLiquidity; param-validation in controller active	• POST /amm/liquidity/remove with tokenId: 'INVALID', decreasePercent: 50	HTTP 404	HTTP 404	✔️

Screenshot



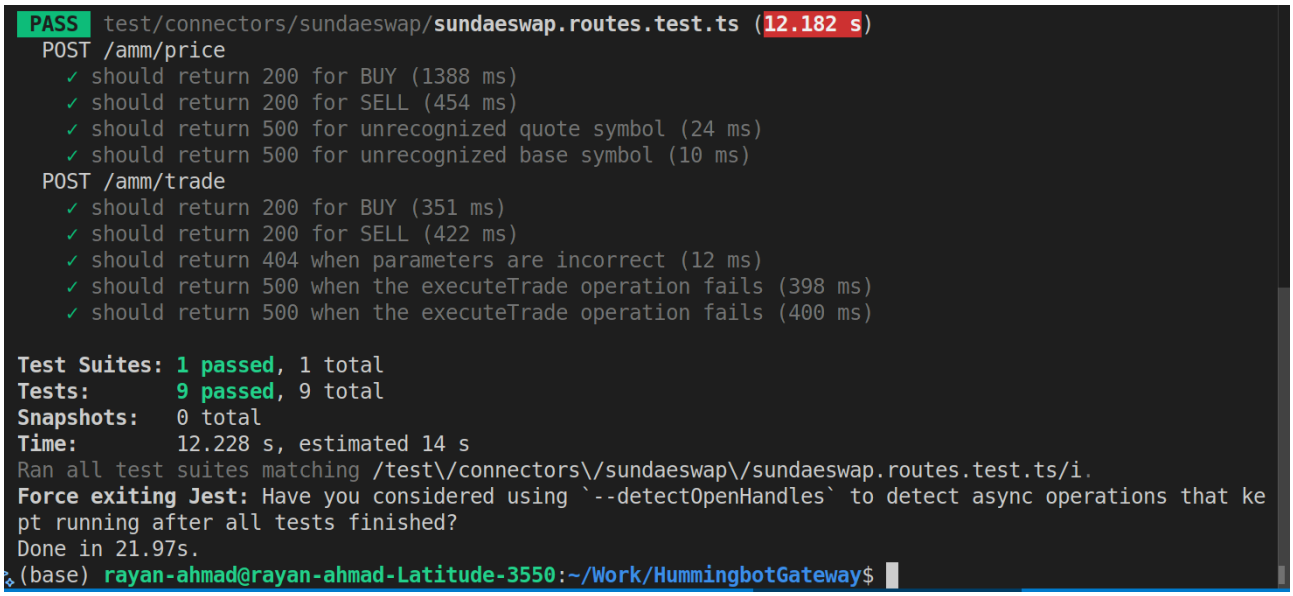
sundaeswap.routes.test.ts

Illustration

Test No.	Endpoint	Test Case	Conditions & Mocks	Steps	Expected Output	Accepted Output	Result
1	POST /amm/price	should return 200 for BUY	patchGetWallet(), patchInit(), patchStoredTokenList(), patchGetTokenBySymbol(), patchExecuteTrade()	• POST /amm/price with { side: 'BUY', base:'SBERRY', quote:'ADA', amount:'10000' }	HTTP 200; res.body.amount & res.body.rawAmount defined	HTTP 200; res.body.amount & res.body.rawAmount defined	✔️
2	POST /amm/price	should return 200 for SELL	same as Test 1	• POST /amm/price with { side: 'SELL', ... }	HTTP 200; amount & rawAmount defined	HTTP 200; amount & rawAmount defined	✔️
3	POST /amm/price	should return 500 for unrecognized quote symbol	same as Test 1 (no patchExecuteTrade() needed)	• POST /amm/price with { quote: 'DOGE', ... }	HTTP 500 error	HTTP 500 error	✔️
4	POST /amm/price	should return 500 for unrecogniz	same as Test 1	• POST /amm/price with { base: 'SHIBA', ... }	HTTP 500 error	HTTP 500 error	✔️

		ed base symbol					
5	POST /amm/trade	should return 200 for BUY	patchForBuy() (includes wallet, init, tokenList, getToken, executeTrade patches)	• POST /amm/trade with { side: 'BUY', amount:'10000', … }	HTTP 200; res.body.txHash defined	HTTP 200; res.body.txHash defined	✔️
6	POST /amm/trade	should return 200 for SELL	patchForSell() (same as above)	• POST /amm/trade with { side: 'SELL', amount:'1000', … }	HTTP 200; res.body.txHash defined	HTTP 200; res.body.txHash defined	✔️
7	POST /amm/trade	should return 404 when parameters are incorrect	only patchInit()	• POST /amm/trade with malformed params (address: 'da8', side: 'comprar')	HTTP 404 error	HTTP 404 error	✔️
8	POST /amm/trade	should return 500 when executeTrade fails (1)	patch(sundaeswap, 'executeTrade', () => 'error')	• POST /amm/trade with { side:'SELL', … } (patched to return invalid)	HTTP 500 error	HTTP 500 error	✔️
9	POST /amm/trade	should return 500 when executeTrade fails (2)	same as Test 8	• POST /amm/trade with { side:'SELL', … } (duplicate failure scenario)	HTTP 500 error	HTTP 500 error	✔️

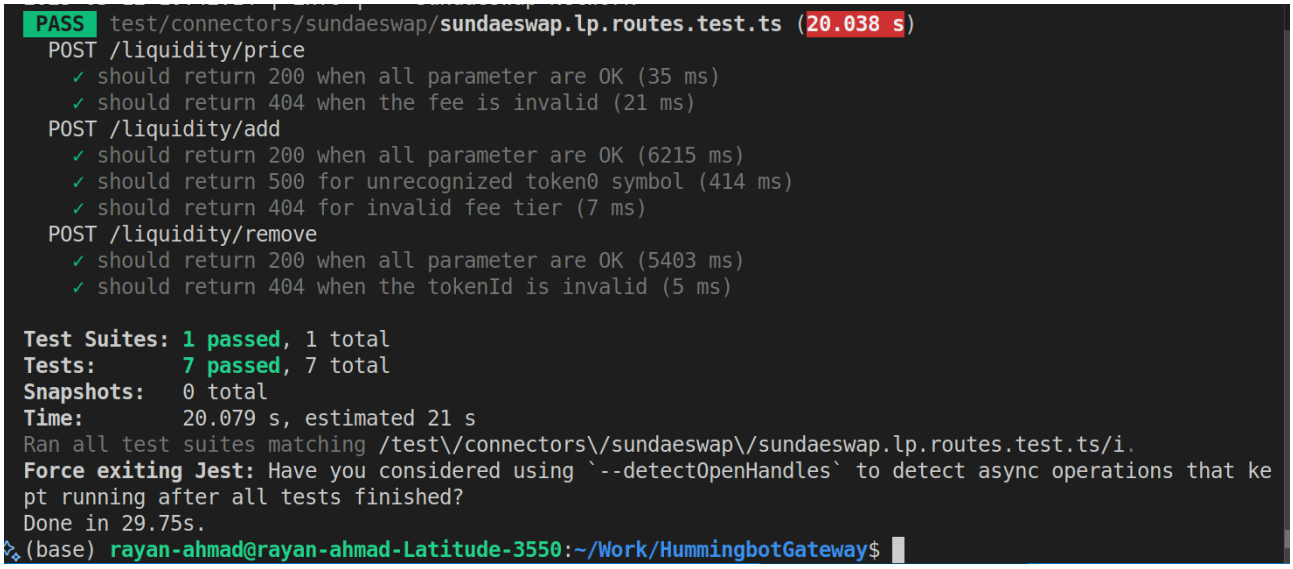
Screenshot:



Test No.	Endpoint	Test Case	Conditions & Mocks	Steps	Expected Output	Accepted Output	Result
1	POST /amm/liquidity/price	should return 200 when all parameters are OK	patchInit(), patchStoredTokenList(), patchGetTokenBySymbol(), patch(sundaeswap, 'poolPrice', () => ['100','105'])	• POST /amm/liquidity/price with valid token0: 'SBERRY', token1: 'ADA', fee: 'LOW', period, interval	HTTP 200	HTTP 200	✔️
2	POST /amm/liquidity/price	should return 404 when the fee is invalid	patchGetWallet(), patchInit(), patchStoredTokenList(), patchGetTokenBySymbol()	• POST /amm/liquidity/price with fee: 11 (non-string/invalid fee)	HTTP 404	HTTP 404	✔️
3	POST /amm/liqu	should return 200	patchGetWallet(), patchInit(),	• POST /amm/liquidity/	HTTP 200; res.body.t	HTTP 200; res.body.txH	✔️

	idity/add	when all parameters are OK	patchStoredTokenList(), patchGetTokenBySymbol(), stubbed addLiquidity in controller returning txHash	add with valid address, token0: 'SBERRY', token1: 'ADA', amount0, amount1, fee: 'LOW'	xHash defined	ash defined	
4	POST /amm/liquidity/add	should return 500 for unrecognized token0 symbol	same mocks as Test 3	• POST /amm/liquidity/add with token0: 'DOGE' (not in tokenList)	HTTP 500	HTTP 500	✔️
5	POST /amm/liquidity/add	should return 404 for invalid fee tier	same mocks as Test 3 (no change to addLiquidity)	• POST /amm/liquidity/add with fee: 300 (invalid fee tier)	HTTP 404	HTTP 404	✔️
6	POST /amm/liquidity/remove	should return 200 when all parameters are OK	patchGetWallet(), patchInit(), patchStoredTokenList(), patchGetTokenBySymbol(), stubbed removeLiquidity returning txHash	• POST /amm/liquidity/remove with valid address, tokenId: 0, decreasePercent: 50	HTTP 200; res.body.txHash defined	HTTP 200; res.body.txHash defined	✔️
7	POST /amm/liquidity/remove	should return 404 when the tokenId is invalid	same mocks as Test 6	• POST /amm/liquidity/remove with tokenId: 'Invalid' (non-numeric)	HTTP 404	HTTP 404	✔️

Screenshot


PASS test/connectors/sundaeswap/sundaeswap.lp.routes.test.ts (20.038 s)
POST /liquidity/price
✓ should return 200 when all parameter are OK (35 ms)
✓ should return 404 when the fee is invalid (21 ms)
POST /liquidity/add
✓ should return 200 when all parameter are OK (6215 ms)
✓ should return 500 for unrecognized token0 symbol (414 ms)
✓ should return 404 for invalid fee tier (7 ms)
POST /liquidity/remove
✓ should return 200 when all parameter are OK (5403 ms)
✓ should return 404 when the tokenId is invalid (5 ms)
Test Suites: 1 passed, 1 total
Tests: 7 passed, 7 total
Snapshots: 0 total
Time: 20.079 s, estimated 21 s
Ran all test suites matching /test\/connectors\/sundaeswap\/sundaeswap.lp.routes.test.ts/i.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?
Done in 29.75s.
(base) rayan-ahmad@rayan-ahmad-Latitude-3550:~/Work/HummingbotGateway$