Back End Test Report

By: Aaron Huggins and Bryar Frank

Overview

The back end testing suites created offer comprehensive testing of the business logic, security, and reliability of the application hosted at

http://ahuggins-warehousemanager-frontend.s3-website.us-east-2.amazonaws.com/

Providing component tests for all available classes and methods, as well as comprehensive testing of each API endpoint. It does this by utilizing TestNG, SonarQube, and Jacoco to achieve 83% line coverage and 61% branch coverage of the application's backend, as well as JMeter to create requests that test the valid and invalid use of the API.

Tests

The testing suite consists of 142 total tests, that achieve a success rate of 87.2%

Component Tests

Making full use of TestNG's capabilities, there are **124** tests that achieve 83% line coverage and 61% branch coverage of the backend code. A full breakdown of the tests created and their success/failure rate is as follows.

Test Group	Number of Tests	Success Rate
Aspect Tests	7	85.7% - (6/7)
Controller Tests	18	100% - (18/18)
Service Tests	26	73.1% - (19/26)
Repository Tests	16	100.0% - (13/13)
Model Tests	4	100% - (4/4)

Aspect Tests

The aspect layer tests created a test class for each of the following backend classes:

- GlobalExceptionHandler
- LoggingAspect
- SecurityFilter

With the following results:

ID	Test Class Name	Test Method Name	Success Rate
AT-1	GlobalExceptionHandler	handleStatusExceptio n	PASS - 1/1 - 100%
AT-2	GlobalExceptionHandler	handleIllegalAccess	PASS - 1/1 - 100%

AT-3	LoggingAspect	testRequest	PASS - 1/1 - 100%
AT-4	SecurityFilter	doFilterInternal	FAIL - 6/7 - 85.7%

Controller Tests

The controller layer tests created a test class for each of the following backend classes:

- AdminController
- ItemController
- WarehouseController

With the following results:

ID	Test Class Name	Test Method Name	Success Rate
CT-1	AdminController	getAdministrators	PASS - 2/2 - 100%
CT-2	AdminController	login	PASS - 2/2 - 100%
CT-3	AdminController	signup	PASS - 2/2 - 100%
CT-4	AdminController	updateAccount	PASS - 2/2 - 100%
CT-5	AdminController	deleteAccount	PASS - 1/1 - 100%
CT-6	ItemController	getWarehouseItems	PASS - 2/2 - 100%
CT-7	ItemController	getItemById	PASS - 2/2 - 100%
CT-8	ItemController	getItemByName	PASS - 2/2 - 100%
CT-9	ItemController	addItemToWarehouse	PASS - 2/2 - 100%
CT-10	ItemController	updateWarehouseItem	PASS - 2/2 - 100%
CT-11	ItemController	removeItemFromWarehouse	PASS - 1/1 - 100%
CT-12	WarehouseController	getAllWarehouses	PASS - 4/4 - 100%
CT-13	WarehouseController	getWarehouseById	PASS - 2/2 - 100%
CT-14	WarehouseController	getWarehouseByName	PASS - 2/2 - 100%

CT-15	WarehouseController	getWarehouseByLocation	PASS - 4/4 - 100%
CT-16	WarehouseController	createWarehouse	PASS - 2/2 - 100%
CT-17	WarehouseController	updateWarehouse	PASS - 2/2 - 100%
CT-18	WarehouseController	deleteWarehouse	PASS - 1/1 - 100%

Model Tests

The model layer tests created a test class for each of the following backend classes:

- Administrator
- Item
- StoredItem
- Warehouse

With the following results:

ID	Test Class Name	Success Rate
MT-1	Administrator	PASS - 1/1 - 100%
MT-2	Item	PASS - 1/1 - 100%
MT-3	StoredItem	PASS - 1/1 - 100%
MT-4	Warehouse	PASS - 1/1 - 100%

Repository Tests

The repository layer tests created a test class for each of the following backend interfaces:

- AdministratorRepository
- WarehouseRepository

And Combined the following two categories into one class:

- ItemRepository
- StoredItemRepository

With the following results:

ID	Test Class Name	Test Method Name	Success Rate
RT-0	AdministratorRepository	generalTest	PASS - 1/1 - 100%
RT-1	AdministratorRepository	findByCompanyNameAndPassword	PASS - 1/1 - 100%
RT-2	AdministratorRepository	findByCompanyName	PASS - 1/1 - 100%
RT-3	ItemRepository	generalTest	PASS - 1/1 - 100%
RT-4	ItemRepository	findByName	PASS - 1/1 - 100%
RT-5	StoredItemRepository	findByWarehouse	PASS - 1/1 - 100%
RT-6	StoredItemRepository	findByItem	PASS - 1/1 - 100%
RT-7	StoredItemRepository	findByItemAndWarehouse	PASS - 1/1 - 100%
RT-8	WarehouseRepository	generalTest	PASS - 1/1 - 100%
RT-9	WarehouseRepository	findByIdAndAdministrator	PASS - 1/1 - 100%
RT-10	WarehouseRepository	findByIdOrNameAndAdministrator	PASS - 1/1 - 100%
RT-11	WarehouseRepository	findByAdministrator	PASS - 1/1 - 100%
RT-12	WarehouseRepository	findByName	PASS - 1/1 - 100%
RT-13	WarehouseRepository	findByNameAndAdministrator	PASS - 1/1 - 100%
RT-14	WarehouseRepository	findByLocation	PASS - 1/1 - 100%
RT-15	WarehouseRepository	findByLocationAndAdministrator	PASS - 1/1 - 100%

Service Tests

The service layer tests created a test class for each of the following backend classes:

- AdminService
- ItemService

- SecurityService
- WarehouseService

With the following results:

ID	Test Class Name	Test Method Name	Success Rate
ST-1	AdminService	getAllAdministrators	PASS - 2/2 - 100%
ST-2	AdminService	getAdministratorById	PASS - 2/2 - 100%
ST-3	AdminService	login	PASS - 2/2 - 100%
ST-4	AdminService	createAdministrator	PASS - 2/2 - 100%
ST-5	AdminService	updateAdministrator	FAIL - 0/2 - 0%
ST-6	AdminService	deleteAdministrator	PASS - 2/2 - 100%
ST-7	ItemService	getWarehouseItems	PASS - 2/2 - 100%
ST-8	ItemService	getItemById	PASS - 2/2 - 100%
ST-9	ItemService	getItemByName	PASS - 2/2 - 100%
ST-10	ItemService	addItemToWarehouse	PASS - 2/2 - 100%
ST-11	ItemService	updateWarehouseItem	PASS - 2/2 - 100%
ST-12	ItemService	removeItemFromWarehouse	PASS - 2/2 - 100%
ST-13	SecurityService	getJwt	FAIL - 2/3 - 66.6%
ST-14	SecurityService	hashString	FAIL - 4/5 - 80%
ST-15	SecurityService	validateAdmin	PASS - 2/2 - 100%
ST-16	SecurityService	validate	PASS - 1/1 - 100%
ST-17	SecurityService	getClaim	FAIL - 0/1 - 0%
ST-18	WarehouseService	getAllWarehouses	PASS - 4/4 - 100%
ST-19	WarehouseService	getWarehouseById	FAIL - 0/2
ST-20	WarehouseService	getWarehouseByNam	PASS - 2/2 - 100%

		e	
ST-21	WarehouseService	getWarehouseByLocation	PASS - 4/4 - 100%
ST-22	WarehouseService	createWarehouse	PASS - 2/2 - 100%
ST-23	WarehouseService	updateWarehouse	PASS - 2/2 - 100%
ST-24	WarehouseService	deleteWarehouse	PASS - 2/2 - 100%

Component Integration Tests

Utilizing JMeter's powerful automated API test software and Postman's manual API test software, each of the 18 API endpoints now has test coverage with a 83.3% pass rate.

A full listing of the HTTP endpoints and their pass rate are as follows:

ID	Method	Path	Success Rate
API-1	GET	Get Administrator's Warehouses /${adminId}	PASS
API-2	GET	Get Warehouse By ID /${adminId}/${warehouseId}	FAIL
API-3	GET	Get Warehouse By Name /${adminId}/warehouseByName	PASS
API-4	GET	Get Warehouse By Location /${adminId}/warehouseByLocation	PASS
API-5	POST	Create Warehouse /${adminId}	PASS - 1/1 - 100%
API-6	PUT	Update Warehouse /${adminId}/${warehouseId}	PASS - 1/1 - 100%
API-7	DELETE	Delete Warehouse /${adminId}/${warehouseId}	PASS 1/1 - 100%
API-8	GET	Get All Administrators /	FAIL
API-9	POST	Login /login	PASS - 1/1 - 100%
API-10	POST	Sign up	PASS - 1/1 - 100%

		/signup	
API-11	PUT	Update Administrator /	PASS - 1/1 - 100%
API-12	DELETE	Delete Administrator /	PASS - 1/1 - 100%
API-13	GET	Get Item By Warehouse /${adminId}/${warehouseId}/items	PASS
API-14	GET	Get Item By ID /${adminId}/${warehouseId}/items/${itemId}	PASS
API-15	GET	Get Item By Name /${adminId}/${warehouseId}/items/item	FAIL
API-16	POST	Add Item To Warehouse /${adminId}/${warehouseId}/items	PASS - 1/1 - 100%
API-17	PUT	Update Warehouse Item /${adminId}/${warehouseId}/items	PASS - 1/1 - 100%
API-18	DELETE	Remove Item From Warehouse /${adminId}/${warehouseId}/items/${itemId}	PASS - 1/1 - 100%

Notes on Failed Tests

ID	Failure Note
AT-4	The SecurityFilter fails to account for requests that contain an authorization token created with the correct secret that do not contain an administrator ID. These requests are incorrectly forwarded to the controller layer.
ST-5	The updateAdministrator method fails to prevent empty data from overwriting existing data.
ST-13	The SecurityService returns a valid JWT when an empty Administrator object is passed in. JWTs should only be issued with a valid administrator ID and company name.
ST-14	The hashString method fails to ensure that the string to hash is not null or empty.
ST-17	The getClaim method, when using the class specifier, generates claims that do not exist; an example of this is when a request for the "random" claim is submitted with the Integer.class specifier, the Integer returned is populated with the base 0 value instead of a null reference.
ST-19	The incorrect parameter is used when creating the lookup administrator.
API-2	Defect found in ST-19 causing a failure in the endpoint.
API-8	This endpoint is intended to be deprecated, but can still be successfully called and lists all administrator accounts, indicating a failure in LoggingAspect::request.
API-15	Attempts to find items in a warehouse by name result in failure. After verifying that the "banana" item exists in a "Really Cool Warehouse" warehouse owned by the "Amazing Company LLC 4" administrator. Reason for this failure is unknown, and warrants further investigation.

Acceptance Criteria

Acceptance Criteria ID	Corresponding Test ID	Pass/Fail
A10	API-10, CT-3, MT-1, ST-4	PASS
A11	API-10, CT-3, MT-1, ST-4	PASS
A12	ST-1, ST-2, RT-1, RT-2, MT-1	PASS
A13	ST-1, ST-2, RT-1, RT-2, MT-1	PASS

A14	API-11, CT-4, ST-5, MT-1	FAIL
A15	API-11, CT-4, ST-5, MT-1	FAIL
A16	API-12, ST-6, MT-1, CT-4	PASS
A17	API-12, ST-6, MT-1, CT-4	PASS
A18	API-8, ST-1, MT-1, CT-1	FAIL
W7	API-5, ST-22, MT-4, CT-16	PASS
W8	API-5, ST-22, MT-4, CT-16	PASS
W9	API-1, API-2, API-3, API-4	FAIL
W10	API-1, API-2, API-3, API-4	FAIL
W11	API-6, ST-23, MT-4, CT-17	PASS
W12	API-6, ST-23, MT-4, CT-17	PASS
W13	API-7, ST-24, MT-4, CT-18	PASS
W14	API-7, ST-24, MT-4, CT-18	PASS
I11	API-13, API-14, API-15	FAIL
I12	API-13, API-14, API-15	FAIL
I13	API-16, ST-10, CT-9, MT-2, MT-3	PASS
I14	API-16, ST-10, CT-9, MT-2, MT-3	PASS
I15	API-17, ST-11, MT-2, MT-3, CT-10	PASS
I16	API-17, ST-11, MT-2, MT-3, CT-10	PASS
I17	API-18, ST-12, MT-2, MT-3, CT-11	PASS
I18	API-18, ST-12, MT-2, MT-3, CT-11	PASS

Other Important Notes

Note Identifier	Description
NB-1	All GET request endpoints return with a 302 FOUND status, a status code that is intended for resources that have been moved. All GET requests should be updated to return 200 OK status due to lack of relocated resources.

Future Testing

While the testing suite is extensive, it does not yet have 100% line or branch coverage of the backend code, and extensive automated integration testing became blocked due to the incorrect status code returns. Due to this, future testing will include:

- More component tests to achieve higher coverage.
- More integration tests that not only test the base functionality of the API, but also the security of the API.