

# **Thryve QA Case Study**

**Task 1 & 2**

**Ngumbah Michael Nyika**

# Task 1: Java Solution

## General Characteristics

**Language Used:** [Java](#)

**Java Testing Framework:** [JUnit 4.8.1](#)

**Java SDK:** [Oracle OpenJDK 22.0.1](#)

**Project Build Tool:** [Maven 4.0.0](#)

**Test Runner Plugin Type:** [Surefire](#)

**Test Runner Plugin Version:** [3.2.5](#)

**Integrated Development Environment (IDE):** [IntelliJ \(JetBrains\)](#)

**Web Client Library Type:** [OKHttp](#)

**Web Client Library Version:** [4.12.0](#)

# Task 1: Haskell Solution

## General Characteristics

**GHC (Glasgow Haskell Compiler) Version:** **9.4.8**

**Haskell Stack Tool Version:** **2.9.3**

**Testing Framework:** **HUnit 1.6.2**

**Integrated Development Environment (IDE):** **Visual Studio Code 1.90.1 Universal**

**Web Client Library Type:** **HTTPConduit 2.3.8.3**

**JSON Unmarshalling Library:** **Aeson 2.2.3.0**

# Task 1

## Artifacts (Java & Haskell)

**Delivery Formats:** Zip

**Project Names:** Thryve\_UploadData\_TestProject and Thryve-UploadData-Haskell

**Java Version Size:** 12KB

**Haskell Version Size:** 2.7MB

## Hardware

Macbook Pro 16-Inch

Apple Silicon

## Tests (Java & Haskell)

**Java Test Class:** PositiveUploadtest.java

**Java Test Method Name:** check\_Uploaded\_HealthDataSample\_Corresponds\_ExactlyTo\_Downloaded\_Sample

**Haskell Test Module:** Main.hs

**Haskell Test Function Name:** test\_Uploaded\_Values\_Matches\_Downloaded\_Values\_Exactly

# Task 1

## General Test Intention

```
"{\"height\": \"175\", \"weight\": \"73.2\", \"birthdate\": \"1974-09-14\", \"gender\": \"female\"}";
```

For Java

Example of User Health Data Upload JSON Content

Check that Health Data **Uploaded** with **these** values, corresponds to the Height and Weight values in the **Downloaded** content, **for that specific Thryve User**

For Haskell

```
"{\"height\": \"167\", \"weight\": \"75.6\", \"birthdate\": \"1974-09-14\", \"gender\": \"female\"}";
```

\*Simple, static numerical values were used for height and weight:

**One way to improve the test** is to generate values within a numerical range that are valid (for both height and weight parameters); It does not make the test any more valuable functionally, but it varies the output interestingly enough to ward off bugs at the edge of the valid ranges, for example, a weight of 0kg or 1kg is technically feasible, but not -1kg

# Task 1

## How to run the Java Implementation with Maven

The screenshot shows a macOS desktop environment. In the top-left corner, there is a terminal window titled "michaelnyika@michaelnyika-T0JM Thryve\_UploadData\_TestProject % mvn clean test". The terminal output is as follows:

```
[INFO] --- resources:3.3.1:testResources (default-testResources) @ Thryve_UploadData_TestProject ---
[INFO] skip non existing resourceDirectory /Users/michaelnyika/IdeaProjects/Thryve_UploadData_TestProject/src/test/resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ Thryve_UploadData_TestProject -- PositiveUploadTest.java
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 1 source file with javac [debug,target 22] to target/test-classes
[INFO] /Users/michaelnyika/IdeaProjects/Thryve_UploadData_TestProject/src/test/java/PositiveUploadTest.java: /Users/michaelnyika/IdeaProjects/Thryve_UploadData_TestProject/src/test/java/PositiveUploadTest.java uses or overrides a deprecated API.
[INFO] /Users/michaelnyika/IdeaProjects/Thryve_UploadData_TestProject/src/test/java/PositiveUploadTest.java: Recompile with -Xlint:deprecation for details.
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ Thryve_UploadData_TestProject ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
[INFO]
[INFO] T E S T S
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0 Time elapsed: 0.617 s -- in PositiveUploadTest
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.608 s
[INFO] Finished at: 2024-06-15T13:00:27+02:00
[INFO]
```

In the bottom-right corner, a file browser window titled "Thryve\_UploadData\_TestProject — zsh — 190x35" is open, showing a list of files and folders in the directory. The list includes:

Name	Date Modified	Size	Kind
dottraceSnapshots	1. Feb 2024 at 14:15		Folder
Downloads	Yesterday at 18:28		Folder
dumps	20. Feb 2023 at 17:23		Folder
Example_Downloaded_Data.rtf	Today at 10:40	1 KB	RTF Doc
IdeaProjects	Today at 12:59		Folder
TDDKata	4. Jun 2024 at 10:23		Folder
.gitignore	3. Jun 2024 at 10:13	344 bytes	Document
idea	Yesterday at 16:12		Folder
src	4. Jun 2024 at 10:23		Folder
test	4. Jun 2024 at 10:46		Folder
Thryve_UploadData_TestProject	4. Jun 2024 at 11:05		Folder
Thryve_UploadData_TestProject.zip	Today at 13:00	12 KB	ZIP arch
Library	3. Jun 2024 at 10:13		Folder
Movies	16. Feb 2024 at 09:08		Folder
Music	12. Feb 2024 at 18:32		Folder
Parallels	17. May 2024 at 09:43		Folder

# Task 1

# How to run the Haskell Implementation with Stack

---

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

- michaelnyika@michaelnyika-T0JM Thryve-UploadData-Haskell % stack build  
The 'package-indices' key is deprecated in favour of 'package-index'.  
Specified source-dir "test" does not exist  
Stack has not been tested with GHC versions above 9.4, and using 9.6.5, this may fail  
Stack has not been tested with Cabal versions above 3.8, but version 3.10.3.0 was found, this may fail
- Warning: Directory listed in `Thryve-UploadData-Haskell.cabal` file does not exist: `test`
- michaelnyika@michaelnyika-T0JM Thryve-UploadData-Haskell % █

After running **stack build**, an executable file is placed in the **.stack-work** hidden directory (this directory lies in the project root folder). Simply access the full path to it to run it.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS > zsh + □ × ... ^ × ● michaelnyika@michaelnyika-T0JM Thryve-UploadData-Haskell % /Users/michaelnyika/Downloads/Thryve-UploadData-Haskell/.stack-work/install/x86_64-osx/8c66de0f31f52daa83e5190383c9636731b10b ac32af5a8d5dee537ef427bc9f/9.6.5/bin/Thryve-UploadData-Haskell-exe Cases: 1 Tried: 0 Errors: 0 Failures: 0 Thryve User Created : Access Token for Session : -> b8406bad90e39cf2162f2ea6c6de2b2 Response Code After Upload: -> 204 Thryve User Health Data Uploaded at TimeStamp: -> 1718625668520 Just (ThryveHealthData {authenticationToken = "b8406bad90e39cf2162f2ea6c6de2b2", dataSources = [DataSource {dataSource = 1001, data' = [Data {startTimestampUnix = 1718625669000, createdAtUnix = 1718625668541, dynamicValueType = 5020, value = "75.6", valueType = "DOUBLE"}, Data {startTimestampUnix = 1718625669000, createdAtUnix = 1718625668541, dynamicValueType = 5030, value = "167", valueType = "LONG"}]}]}) Cases: 1 Tried: 1 Errors: 0 Failures: 0 ○ michaelnyika@michaelnyika-T0JM Thryve-UploadData-Haskell %
```

# Task 1

## Java IDE

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** Thryve\_UploadData\_TestProject
- Project Explorer:** Shows the project structure under "Thryve\_UploadData\_TestProject". It includes a .idea folder, a src folder with main and test subfolders, and a target folder. Inside test/java, there is a PositiveUploadTest.java file.
- Code Editor:** The PositiveUploadTest.java file is open. The code defines a test class for uploading data to a REST API. It includes imports for java.io, java.text.SimpleDateFormat, java.util.AbstractMap, java.util.Base64, and java.util.Date. The class contains constants for height and weight, authentication tokens, and URLs for user creation, upload, and download endpoints. It also includes security credentials for authString and appAuthString. A setup method, @Before, is defined to handle the initial setup of these variables.
- Status Bar:** At the bottom, it shows the file path as Thryve\_UploadData\_TestProject > src > test > java > PositiveUploadTest > upload\_Thryve\_UserData, and the file statistics as 94:138 (111 chars) LF UTF-8 4 spaces.

# Task 1

## Haskell IDE

The screenshot shows a Haskell IDE interface with the following components:

- EXPLORER** sidebar: Lists project files including `.stack-work`, `app` (containing `Main.hs`), `src`, `test`, `ThryveConstants.hs`, `ThryveTypes.hs`, `ThryveUpload.hs`, `ThryveUtils.hs`, `.gitignore`, `CHANGELOG.md`, `LICENSE`, `package.yaml`, `README.md`, `Setup.hs`, `stack.yaml`, `stack.yaml.lock`, and `Thryve-UploadData-Haskell.cabal`.
- EDITOR**: The `Main.hs` file is open, showing code related to ThryveUpload and ThryveConstants imports, a main function, and test cases for uploaded data values.
- TERMINAL**: Shows command-line output from running the application and performing a stack clean operation.
- STATUS BAR**: Displays file statistics (Ln 24, Col 55 (54 selected)), editor settings (Spaces: 4, UTF-8, LF), and Haskell version.

## Task 2

### Functional Cases Brainstorm

# Task 2

## Areas of Coverage (1)

[Required Params] Authentication Credentials: Secured versus Unsecured

[Required Params] Authentication Credentials: Partly Missing Credentials

[Optional Params] Authentication Credentials: Single, Correct Size Values, Correct Content-Type

[Optional Params] Single Param: Correct Size Values, Invalid Content-Type (eg: invalid chars)

[Optional Params] Single Param: Incorrect Size Values (too short, too long)

[Optional Params] Single Param: No content

[Optional Params] Multiple Params: Correct Size Values, Correct Content-Type \*

[Optional Params] Multiple Params: Correct Size Values, Invalid Content-Type (eg: invalid chars) \*

[Optional Params] Multiple Params: Incorrect Size Values (too short, too long) \*

[Optional Params] Multiple Params: No content \*

\*All Optional parameters can/should be manipulated in a consistent manner, across the board; however, exploratory testing would expand that technique to include irregular patterns combinations: for example, one parameter has fewer characters than normal, another has more characters but invalid symbols, another is missing, etc.

# Task 2

## Areas of Coverage (2)

**[Required & Optional] Operational Error Code: Check correct HTTP Response codes (500, 403, 401, 301, etc) correctness**

**[Optional] Valid Standard Language Values work with correct HTTP Response codes (eg: en, de, dk)**

**[Optional] Invalid Language Values work with correct HTTP Response codes (eg: zz, qq, mm)**

} For example: language Option in  
Query Parameter for Thryve User  
Generation

**[Required] Upload: JSON: Proper format, all values filled with correct values within boundary ranges**

**[Required] Upload: JSON: Missing values**

**[Required] Upload: JSON: Missing entire key-value JSON objects**

**[Required] Upload: JSON: Empty lists (JSONArrays)**

**[Required] Upload: Missing Content / No Content at all**

**[Required] Upload: JSON: Very Large valid payloads (gigabytes)**

**[Required] Upload: JSON: Successful upload, check Response Code (204 No-Content code, rather than 200)**

**[Required] Upload: Duplicate Required fields with valid values, for example, multiple **authenticationToken** fields supplied when uploading simple data**

## Task 2

### Areas of Coverage (3) : Special Note on Time/Date Fields

[Optional Params] Test Using a Date representing Epoch (Jan 1 1970), though one cannot retrieve records older than 31 days

[Optional Params] Retrieve records exactly 31 days old, slightly older (literally by seconds or minutes) or slightly under the limit

[Optional Params] Retrieve records for Leap Years

[Optional Params] Retrieve records when request has duplicated fields (for example, multiple `startTimestampUnix` field values)

[Optional Params] Missing half of a pair of dates combination, required to make a valid query (for example, not specifying `endTimestampUnix` even though `startTimestampUnix` was specified

[Outlier Monkey TestCase] Send a valid upload request with valid payload and credentials, **but add headers not accounted for in the API Requirements** (for example, `Cache-Control`, `Keep-Alive`, `Cross-Origin-Embedder-Policy`, etc)



## Task 2

### Non Functional Cases Brainstorm

# Task 2

## Non-Functional Cases Brainstorm

**100,000+ Thryve users created simultaneously: Check Response Times**

**100,000+ Thryve users created simultaneously: Check all responses are HTTP 200 (valid credentials presented)**

**100,000+ Thryve users created simultaneously: At least 50% invalid credentials; Check HTTP Responses**

**10,000+ Thryve users created simultaneously and consistently every second for 1 minute**

**10,000+ Thryve users created simultaneously: Vary the load distribution of User creation requests pseudo-randomly**

**100+ Thryve users created with Gigabytes of Payload Data (JSON), sent intermittently**

**Error Performance Handling: 10,000+ Thryve users attempted creation with bad payloads, bad authentication credentials**

**Manual Test: Error Performance Handling: Upload large payload over a slow internet connection (wired broadband or low-bandwidth wifi); repeat the test with internet connection that breaks and resumes; see if any partial payloads can be successfully downloaded**

# ! Thank You for the Challenge !



Ngumbah Michael Nyika

June 2024