# Vodafone Java Test

This is the initial test for a Java developer role at Vodafone. For this take-home exam, we have defined REST APIs for an application you will need to build. Data can be stored in memory, so you do not need to worry about connecting to a database or using an advanced caching solution. You are encouraged to use a data structure from the Java Collections library.

You may include additional third-party libraries as you see fit. We ideally expect 80% unit test coverage. Writing additional automated tests such as integration tests is optional, but you are welcome to do so if you wish.

Please write "production-quality" code and add comments when the code's behavior is not clear.

**Every solution must:**

* Be written in Java 17.
* Be compiled and packaged using Maven.
* Use open-source Spring Boot libraries.
* Include a "README.md" file explaining how to compile, run, and test the application.

**Important Note:** These exercises simulate real-world scenarios and may contain incorrect information. If you encounter inaccuracies, feel free to modify or rectify them and document your changes in the "README.md" file.

## Java Exam - Vodafone Consumer IoT (version 2)

Write a Java Spring Boot microservice to expose a simple REST API that reports the location of IoT tracking devices loaded in batches from a CSV file.

Here is an initial example:



Here is the contenct if you can’t open the file:

|  |
| --- |
| DateTime,EventId,ProductId,Latitude,Longitude,Battery,Light,AirplaneMode  1582605077000,10001,WG11155638,51.5185,-0.1736,0.99,OFF,OFF  1582605137000,10002,WG11155638,51.5185,-0.1736,0.99,OFF,OFF  1582605197000,10003,WG11155638,51.5185,-0.1736,0.98,OFF,OFF  1582605257000,10004,WG11155638,51.5185,-0.1736,0.98,OFF,OFF  1582605257000,10005,6900001001,40.73061,-73.935242,0.11,N/A,OFF  1582605258000,10006,6900001001,40.73071,-73.935242,0.1,N/A,OFF  1582605259000,10007,6900001001,40.73081,-73.935242,0.1,N/A,OFF  1582605317000,10008,WG11155800,45.5185,-12.52029,0.11,ON,OFF  1582605377000,10009,WG11155800,45.5186,-12.52027,0.1,ON,OFF  1582605437000,10010,WG11155800,45.5187,-12.52025,0.09,ON,OFF  1582605497000,10011,WG11155638,51.5185,-0.17538,0.95,OFF,OFF  1582605557000,10012,6900001001,40.73081,-73.935242,0.1,N/A,OFF  1582605615000,10013,6900233111,,,0.1,N/A,ON  1582612875000,10014,6900233111,,,0.1,N/A,OFF |

There are three parts to this assignment. Each part should be considered a minimally viable product (MVP), providing a fully functional solution that is well-tested, clearly documented, and consists of production-quality code.

Please use Java 8 or higher, with lambda expressions and streams where appropriate. Modify the data file to include more scenarios as necessary; we will be testing the controller with a larger file to ensure all boundary conditions and edge cases are handled appropriately.

### Part 1 MVP: IoT batch data loading service

**Assumptions:** IoT location tracking devices generate data that is sent from the device to the backend platform in a specific location. It is not yet loaded in memory.

Please write a service and REST controller capable of batch loading the data provided in the attached CSV file (see data.csv) and matching the following endpoint:

|  |  |
| --- | --- |
| URL Path | iot**/**event**/**v1**/** |
| HTTP Method | POST |
| Body | **{**  filepath**:** "C:/path/to/data.csv"  **}** |

SUCCESSFUL RESPONSE:

|  |  |
| --- | --- |
| HTTP Status | 200 OK |
| Body | **{**  description**:** "data refreshed"  **}** |

The data must be loaded into memory without using a database or cache. Each time this endpoint is called, the data should completely replace the existing data with the contents of the CSV file.

Use Swagger to document and test this endpoint.

Unhappy Paths

1. If the path to the data.csv is incorrect or the file cannot be found please return the following message:

**Unsuccessful Response**

|  |  |
| --- | --- |
| HTTP Status | 404 Not Found |
| Body | **{**  description**:** "ERROR: no data file found"  **}** |

1. For any other exceptions, return 500 with the message "ERROR: A technical exception occurred." You may append additional details for debugging.

**Unsuccessful Response**

**(Misc technical exception)**

|  |  |
| --- | --- |
| HTTP Status | 500 |
| Body | **{**  "description"**:** "ERROR: A technical exception occurred"  **}** |

### Part 2 MVP+1: Report device details and location

Extend your microservice from Part 1 to include an additional REST endpoint that reports a device's location given the productId and an optional timestamp parameter.

The timestamp lookup should return a single set of data (i.e., a row from the CSV file) matching the DateTime and ProductId fields in the CSV file. If the provided timestamp is not an exact match, return the data closest to it in the past for the given productId. If no timestamp parameter is provided, default to the current UTC time. All datetimes should relate to UTC, but you may allow this to be configurable, provided the default configuration is UTC with milliseconds.

Please build this API to match the following endpoint definition:

|  |  |
| --- | --- |
| URL Path | iot**/**event**/**v1**?**ProductId=WG11155638**?**tstmp=1582605137000 |
| HTTP Method | GET |

SUCCESSFUL RESPONSE:

|  |  |
| --- | --- |
| HTTP Status | 200 OK |
| Body | **{**  "id"**:**"WG11155638"**,**  "name"**:**"CyclePlusTracker"**,**  "datetime"**:**"25/02/2020 04:31:17"**,**  "long"**:**"51.5185"**,**  "lat"**:**"-0.1736"**,**  "status"**:**"Active"**,**  "battery"**:**"Full"**,**  "description"**:**"SUCCESS: Location identified."  **}** |

As this is a customer-facing product, you need to format the data to be user-friendly.

1. **Differentiate between products:** The two supported products are:
   * CyclePlusTracker (product ID starts with 'WG') is used on bicycles.
   * GeneralTracker (product ID starts with '69').
2. **Handle airplane mode:**
   * If airplane mode is on, the GPS data will not be available. The response should still be 200 OK and include the description "SUCCESS: Location not available: Please turn off airplane mode." Provide other details about the product, with latitude and longitude left blank.
   * If airplane mode is off, provide the GPS data and a 200 OK response with a success message.
3. **Set status flag:**
   * The status flag should be 'Active' when latitude and longitude data are present.
   * When no GPS data is available, such as when airplane mode is on, the status should be 'Inactive.'
4. **Report battery life:** Report battery life as:
   * 'Full' if it is ≥98%.
   * 'High' if it is ≥60%.
   * 'Medium' if it is ≥40%.
   * 'Low' if it is ≥10%.
   * 'Critical' if it is <10%.

Unhappy Paths

1. If the AirplaneMode is off and there is no GPS data available in the csv file the status should be 400:

**(GPS not reported)**

|  |  |
| --- | --- |
| HTTP Status | 400 |
| Body | **{**  "description"**:** "ERROR: Device could not be located"  **}** |

1. If the user provides an Id that is not present in the csv file then a 404 should result:

**(ID not found)**

|  |  |
| --- | --- |
| HTTP Status | 404 |
| Body | **{**  "description"**:** "ERROR: Id <insert productId> not found"  **}** |

### Part 3 MVP+2: Dynamic activity-tracking

Following the release of 1.1.2, users of the CyclePlusTracker have requested a way to determine if they are actively cycling, resting, or stopped at a traffic light.

Design teams and architects suggest modifying the functionality of the status field to achieve this. The status should remain 'Inactive' as long as three consecutive sets of GPS coordinates are identical. If the device is just starting and there are not enough readings to compute three sets of coordinates, the status should be 'N/A.'

These changes should only apply to the CyclePlusTracker. The existing functionality for the GeneralTracker should remain unchanged from Part 1.1.2.