Coding Exercise

As part of the interview process for backend/API engineering at Skyward, we ask you to complete a coding exercise.

Complete this coding exercise ahead of your interview. Please have it ready on the day of your interview.

In case of an in-person interview: This could be a laptop, a thumb drive or an online Git repository. We do not review your solution ahead of time.

In case of a remote interview: Have it available to walk us through it using screen-sharing from the same computer used to join the remote interview.

Please timebox this exercise, and include the things that you consider most valuable in delivering and in showcasing to us. We don't expect you to finish the entire exercise at a production level. If you have questions, we encourage you to put yourself in the position of the Product Owner. Be ready to explain your reasoning.

Drone Logfile Ingest

Create a REST service that addresses the user stories listed below. Feel free to make any additional technical design decisions yourself in order to solve the problem more elegantly. A sample logfile is provided at the end of this document.

Be prepared to explain how you would optimize the design, test, and deploy this system to function well for files ranging in size from 1 kilobyte to 10 megabytes and scale to high request volumes. Upon completion of the practicum, you will participate in a practicum review with the team, including a 5-10 minute teaching session to explain your work to an engineer that may not have experience with the languages and frameworks you used to develop the service.

Most of our existing services are written in Java with Spring Boot as the framework, however these are not requirements for this exercise. If you choose to use a different language or framework, please be prepared to explain why, and be prepared to explain how this would also be accomplished using Spring.

User Stories

As a Drone Pilot, I want to upload a drone logfile so I can store it for later analysis.
Acceptance Criteria:

1. I can make a RESTful request to the server that contains the sample logfile using software such as Postman.
2. My logfile is stored in its entirety on the server.
3. The request returns a unique identifier to later retrieve the logfile.

As a Drone Pilot, I want to know my starting and ending battery voltages so that I know how much of the battery was consumed during a flight.
Acceptance Criteria:

1. I can make a restful GET request to the server to obtain the starting and ending voltages in JSON using software such as Postman.
2. If I use an identifier that doesn't exist, I receive an error that tells me the file doesn't exist.

The starting and ending battery voltages can be found by looking at the values in the oldest and newest event, such as **battery_charge**, **battery_voltage**, or even **remaining_life** (your choice).

As a Drone Pilot, I want to retrieve my logfile so that I can import it to other third party services.
Acceptance Criteria:

1. I can make a restful GET request to the server to obtain my original logfile. 2. If I use an identifier that doesn't exist, I receive an error that tells me the file doesn't exist.

Logfile

```
{
 "exchange_type" : "flight_logging",
 "exchanger" : "Skyward GCS v1.0.3",
 "uploadingOrgUuid" : null,
 "uploadingPilotUuid" : "",
 "flight_session_id" :
"G650131FrTJmN2mMxnPX2019-01-03T10:21:59.148-0800",  "message" : {
 "file" : {
 "creation_dtg" : "2019-01-03T10:21:59.149-0800",
 "logging_type" : "SKYWARD_GCS",
 "filename" : "G650131FrTJmN2mMxnPX2019-01-03T10:21:59.148-0800"
},
 "message_type" : "flight_logging_submission",
 "flight_logging" : {
 "event" : [
 {
 "event_info" : "TAKE_OFF",
 "event_timestamp" : "2019-01-03T10:21:45.945-0800",
 "event_type" : "CONTROLER_EVENT"
 }
 ],
 "altitude_system" : "WGS84",
 "logging_start_dtg" : "2019-01-03T10:21:43.648-0800",
 "flight_logging_items" : [
 [
 "2019-01-03T10:21:48.649-0800",
 -122.6704425835603,
 45.523050743642223,
 1.7000000000000002,
 0,
 -157.60000000000002,
 0,
 0,
 0,
 0,
 0,
 -157.60000000000002,
 15,
 1,
 1,
 "GPS",
 0,
 0,
 1,
```

1.7000000000000002,

```
0,
0,
0,
0,
0,
0,
5988,
-1480,
15977,
46.950000000000045,
4071,
"2019-01-03T10:21:48.248-0800",
3998,
4001,
3997,
3997,
0,
0,
"2019-01-03T10:21:48.251-0800",
37.787358900000001,
-122.408227,
0,
0,
0,
-61.799999237060547,
2
],
[
"2019-01-03T10:21:53.650-0800",
-122.6704425835603,
45.523050743642223,
1.2000000000000002,
0.5,
-157.60000000000002,
0,
0,
0.5,
0,
0,
-157.60000000000002,
15,
1,
1,
"GPS",
0,
0,
1,
1.2000000000000002,
```

```
0,
0,
0,
0,
0,
0,
5988,
-1482,
15977,
46.950000000000045,
4071,
"2019-01-03T10:21:53.248-0800",
3997,
4001,
3996,
3996,
0,
0,
"2019-01-03T10:21:53.252-0800",
37.787358900000001,
-122.408227,
0,
0,
0,
-61.799999237060547,
2
],
[
"2019-01-03T10:21:58.648-0800",
-122.6704425835603,
45.523050743642223,
0,
0,
-158.60000000000002,
0,
0,
0,
0,
0,
-158.60000000000002,
15,
1,
1,
"GPS",
0,
0,
1,
0,
0,
```

```
0,
0,
2339,
0,
0,
5988,
-1480,
15977,
46.950000000000045,
4071,
"2019-01-03T10:21:58.248-0800",
3998,
4000,
3996,
3996,
0,
0,
"2019-01-03T10:21:58.251-0800",
37.787358900000001,
-122.408227,
0,
0,
0,
-62.799999237060547,
2
]
],
"flight_logging_keys" : [
"timestamp",
"aircraft_lon",
"aircraft_lat",
"aircraft_altitude",
"aircraft_speed",
"aircraft_heading",
"aircraft_vel_x",
"aircraft_vel_y",
"aircraft_vel_z",
"aircraft_pitch",
"aircraft_roll",
"aircraft_yaw",
"aircraft_satellites",
"aircraft_motors_on",
"aircraft_flying",
"aircraft_flight_mode",
"aircraft_flight_mode_value",
"aircraft_imu_preheating",
"aircraft_ultrasonic_on",
"aircraft_ultrasonic_altitude",
"aircraft_vision_on",
```

```
"aircraft_gps_signal",
"aircraft_gps_signal_value",
"aircraft_smart_gohome_flight_time_remaining",
"aircraft_smart_gohome_requesting",
"aircraft_last_updated",
"battery_power",
"battery_current",
"battery_voltage",
"battery_temperature",
"battery_charge",
"battery_last_updated",
"battery_cell_one",
"battery_cell_two",
"battery_cell_three",
"battery_cell_four",
"battery_cell_five",
"battery_cell_six",
"battery_cell_last_updated",
"gcs_lat",
"gcs_lon",
"gcs_location_last_updated",
"gimbal_pitch",
"gimbal_roll",
"gimbal_yaw",
"gimbal_mode"
]
},
"flight_data" : {
"payload" : {
"camera" : {
"serial_number" : "BAD6E9W001008B",
"model" : "Phantom 4 Pro Camera",
"firmware_version" : "01.07.06.105"
},
"gimbal" : {
"serial_number" : null,
"firmware_version" : "01.50.13.17"
}
},
"remote_controller" : {
"serial_number" : null,
"firmware_version" : null
},
"aircraft" : {
"manufacturer" : "DJI",
"serial_number" : "90DFE9F002002Y",
"name" : "Charlie PHANTOM 4 PRO",
"model" : "Phantom 4 Pro",
"firmware_version" : "01.05.0600"
```

Skyward
A Verizon company

```
  },
  "summary" : {
  "home_location_lat" : "45.52305073504952",
  "home_location_lon" : "-122.67044258416709",
  "aircraft_smart_gohome" : {
  "flight_return_time" : null,
  "landing_power" : null,
  "return_power" : "15",
  "landing_radius" : "9122.101",
  "landing_time" : null
  }
  },
  "gcs" : {
  "manufacturer" : "SkywardIO",
  "model" : "Skyward GCS",
  "version" : "1.0.3"
  },
  "flight_session_end" : "2019-01-03T10:21:59.148-0800",
  "flight_controller" : {
  "serial_number" : "90DFE9F002002Y",
  "firmware_version" : "03.02.44.07"
  },
  "battery" : {
  "serial_number" : "FT3DE8H03203JZ",
  "remaining_life" : "0",
  "discharges" : "22",
  "full_charge_volume" : "5988",
  "cell_number" : "4",
  "firmware_version" : "02.00.07.30"
  },
  "flight_session_start" : "2019-01-03T10:21:43.648-0800",
  "logfile_device_origin" : {
  "user_interface_idiom" : "iPad",
  "operating_system" : "iOS12.1",
  "model" : "iPad",
  "device_ssid" : ""
  }
  }
  }
}
```