Table of Contents

[Concept 2](#_Toc71213149)

[UI Mockups 3](#_Toc71213150)

[Feature List 7](#_Toc71213151)

[Data Structure Definition 9](#_Toc71213152)

[API 10](#_Toc71213153)

[Branding Guidelines 21](#_Toc71213154)

# Concept

Below is the flow chart for Lab mode. This includes the three use cases 1) Technician and Reader online, 2) Technician online but Reader offline and 3) Technician and Reader offline

Diagram

Description automatically generated

This workflow is easily adaptable for the at home mode where the technician is replaced by the patient. The assumption there is that the Patient and the Reader are both online and the user has an account already so there is not patient information collected.

# UI Mockups

Based on this, the following shows the screen mockups (mobile version) **(**Interactive desktop version at <https://invis.io/M410VKN3CNXF#/450908554_Desktop-Login>**)**

**Login Screen**

Graphical user interface, application, chat or text message

Description automatically generated

**Home Screen + User Menu**

Chart, funnel chart

Description automatically generated Chart, funnel chart

Description automatically generated

**Register Patient (Search)**

Graphical user interface, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

**New Patient Entry**

Graphical user interface

Description automatically generated with medium confidence

**Associate Patient to Collector**

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

**Test Result Search**

Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, application

Description automatically generated

**Test Result Details (Test Complete, Test Incomplete)**

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

# Feature List

Galen will create two an application and configure the Galen Cloud as described below for a Phase 1 prototype effort.

1. Aptitude Web Application (AWA) – Implementing the workflow as described in the diagram above.
   1. The AWA shall be designed to execute on an internet-based server.
   2. The AWA shall allow a lab technician to login to the Galen Cloud
      1. The AWA shall allow the lab technician to reset a forgotten password
   3. The AWA shall allow the lab technician to enter patient information for a new test
      1. First Name
      2. Middle Name
      3. Last Name
      4. Date of Birth
      5. Gender (Male, Female, Other, Declined to identify)
      6. Race (American Indian or Alaskan Native, Asian, Black or African American, Native Hawaiian or Pacific Islander, White, Other, Declined to identify)
      7. Ethnicity (Hispanic or Latino, Not Hispanic or Latino, Declined to identify)
      8. Email Address
      9. Phone Number
      10. Street Address 1
      11. Street Address 2
      12. City
      13. State
      14. Zip code
      15. Password
      16. Confirm Password
   4. The AWA shall allow the lab technician to associate a patient to a collector via entering or scanning a Collector Code (entered or scanned via a QR code)
   5. The AWA shall create a patient account (unless the account already exists) and save the patient + collector mapping in the Galen Cloud.
   6. The AWA shall allow the lab technician to enter or scan the reader and collector codes before the test run begins
      1. The AWA shall create the test entry (patient + collector + reader mapping) with status = Pending in the Galen Cloud
      2. The AWA shall create a notification when the test is started. Note: Notification are kept for that session only, once the user logs out or closes the browser, old notifications are no longer accessible
      3. When the notification is clicked, the AWA shall show the details of that test run (see item h below)
   7. The AWA shall allow the lab technician to search for test results associated with that lab/site
   8. The AWA shall allow the lab technician to view details on each test result
   9. The AWA shall allow the lab technician to manually update the test results if the result is not already set (test not completed).
      1. The AWA shall allow the lab technician to upload a photo (png or jpeg) associated with the test during manual upload
      2. The AWA shall save the test results and mark the test as Completed
   10. The AWA shall allow the lab technician to bulk insert test information from a CSV file
       1. The AWA shall require a compatible CSV file to be uploaded
       2. For each test information in the CSV file the AWA shall
          1. Create patient account (if needed)
          2. Create the test record (patient + collector + reader + test results)
   11. The AWA shall allow the lab technician to update their own profile and password

Galen Cloud functions (Included as part of the effort) as described in the diagram above.

1. Notification (Email, SMS) to patients upon completion of test

Reader function (blue boxes in the diagram above) are **NOT** included in this effort and will be responsibility of Aptitude) including but not limited to

1. The Reader software shall poll the Galen Cloud for a pending test record using its reader identifier
2. The Reader software shall use the test record identifier to send results
3. The Reader software shall indicate an error if the no such record exists
4. The Reader software shall indicate an error if the reader is offline
5. Creation and management of the Wi-Fi network functionality for the Reader

# Data Structure Definition

**Data Model: Test Information** - Scope: Patient (Patient info is auto captured by Galen Cloud)

DeviceModelId: edfcab65-18d6-419d-abdc-86d413298b1a

DevicePropertySetId: 13867531-fcb9-4ccf-bc12-3a53f4385117

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Prop Code** | **Description** |
| Collector Id (Unique) | Text | CollectorId | Identification of the Collector |
| Reader Id | Text | ReaderId | Identification of the Reader |
| Sample Collected On | Date and Time | SampleCollectedOn | The date and time the sample was collected (the time the patient was associated with the collector) |
| Sample Collected By | Text | SampleCollectedBy | The identifier of the user collecting the sample |
| Test Run By | Text | TestRunBy | The identifier of the user doing the test |
| Test Start Date | Date and Time | TestStartDate | The date and time the test was started. This is going to be the same as "Assigned To Reader", but could diverge in future |
| Test End Date | Date and Time | TestEndDate | The date and time the test was ended. |
| Test Complete | Yes/No | TestComplete | A flag to indicate if the test is complete or not |
| Test Result | Text | TestResult | The result (positive, negative, invalid) of the test |
| Test Photo | Image | TestPhoto | Image of the test results (LED lights e.g.) |

**Data Model: Reader List –** Scope: Practice(Site (Practice) Info is auto captured by Galen Cloud)

DeviceModelId: de4769bd-3226-40a2-91ab-4d585c1a307c

DevicePropertySetId: 00301ca3-8272-4d5d-b216-17c675a9f2a0

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Prop Code** | **Description** |
| Reader Id (Unique) | Text | ReaderId | Identification of the Reader |
| Created on | Date and Time | CreatedOn | The date and time the reader was created in the system |
| Heartbeat Received On | Date and Time | HeartbeatReceivedOn | The date and time the last heartbeat was received on |
| Status | Text | Status | Error or other status |

**Data Model: Reader Raw Data** – Scope: Manufacturer

DeviceModelId: 5eabb5f6-da51-49eb-9a23-8cd22ad417c5

DevicePropertySetId: 4f111387-0681-45ed-b02a-2685edc819be

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Prop Code** | **Description** |
| Reader Id | Text | ReaderId | Identification of the Reader |
| Collector Id | Text | CollectorId | Identification of the Collector on which this test was run. |
| Test Date | Date and Time | TestDate | The date and time the test was run on reader |
| Measurement | JSON | Measurement | The raw measurements from the reader |
| Temperature | JSON | Temperature | The temperature values from the reader |
| Fluid Fill | JSON | FluidFill | The fluid fill values from the reader |
| Curve Fit | JSON | CurveFit | The curve fit values from the reader |
| Result QC | Text | ResultQc | The quality control text |
| Firmware Version | Text | FirmwareVersion | The firmware version of the reader |

# API

**Common info across all API endpoints**

Base URL: <https://api.galencloud.com>

Headers:

X-TENANT-DOMAIN: aptitudedev.galencloud.com (needs to be configurable for production)

X-API-VERSION: 3

X-APP-TYPE: Device

**APIs for Reader Software**

1. Login (Using master token, one for each reader. Master token can be generated using Galen Web Portal under Manufacturer Settings > Security)
   1. End point: POST /auth/refresh-token
   2. Additional Request Headers: None
   3. Query Params: None
   4. Request Body: master token string
   5. Response Header: auth token in header under header name Authorization. The auth token is in Bearer format
   6. Response Body: None
   7. Example (Python – upper case variables are constants/configuration values):

def authenticate():

url = BASE\_URL + "/auth/refresh-token"

headers = {

"X-APP-TYPE": APP\_TYPE,

"X-TENANT-DOMAIN": TENANT\_DOMAIN,

"X-API-VERSION": API\_VERSION

}

payload = MASTER\_TOKEN

response = requests.post(url, headers=headers, data=payload)

response.encoding = response.apparent\_encoding

if response.status\_code == 200:

token = response.headers["Authorization"]

return token

else:

raise Exception("Invalid Authentication - " + str(response.text))

1. Retrieve Pending Test for the Reader
   1. End point: POST /data/devicedata-advanced
   2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
   3. Query Params: pageNumber=0&pageSize=1
   4. Request Body: Json (see below) with deviceDataModelId, deviceCriteria, devicePropertyCode values

**{**

**"deviceDataModelId": "{TestInfoDeviceModelId}",**

**"deviceCriteria": [**

**{**

**"key": "ReaderId",**

**"operator": "Equal",**

**"value": "string"**

**},**

**{**

**"key": "TestComplete",**

**"operator": "Equal",**

**"value": "true"**

**}**

**],**

**"devicePropertyCodes": ["CollectorId", "ReaderId", "TestStartDate"]**

**}**

* 1. Response Header: None
  2. Response Body: Json (simplified beow), with page of data. Each item in the data array corresponds to a single record

**{**

**"empty": true,**

**"first": true,**

**"last": true,**

**"number": 0,**

**"numberOfElements": 1,**

**"size": 1,**

**"totalElements": 1,**

**"totalPages": 1,**

**"content": [**

**{**

**"deviceDataId": "{TestInfoDeviceDataId}",**

**"deviceDataModelId": "{TestInfoDeviceModelId}",**

**"ownerId": "{patientUserId}",**

**"data": {**

**"CollectorId": {**

**"value": "string",**

**},**

**"ReaderId": {**

**"value": "string"**

**},**

**"TestStartDate": {**

**"value": "2021-05-06T14:22:38.395Z"**

**}**

**}**

**}**

**]**

**}**

* 1. Example (Python – upper case variables are constants/configuration values):

def getPendingTest(token, readerId):

url = BASE\_URL + "/data/devicedata-advanced";

payload = {

"deviceDataModelId" : TEST\_INFO\_DEVICE\_MODEL\_ID,

"deviceCriteria" : [{

"key": "ReaderId",

"operator": "Equal",

"value": readerId

},{

"key": "TestComplete",

"operator": "Equal",

"value": false

}]

}

headers = {

"X-APP-TYPE": APP\_TYPE,

"X-TENANT-DOMAIN": TENANT\_DOMAIN,

"X-API-VERSION": API\_VERSION,

"Content-Type": "application/json",

"Authorization": token

}

response = requests.post(url, headers=headers, data = json.dumps(payload))

response.encoding = response.apparent\_encoding

if (response.status\_code != 200):

raise Exception("Could not retrieve device data - reported error - " + str(response.text))

responseDict = json.loads(response.text)

contentArr = responseDict["content"];

if (len(contentArr) == 0):

raise Exception("Could not retrieve device data - none found")

dataDict = contentArr[0]["data"];

simpleDict = {}

for key in dataDict:

simpleDict[key] = dataDict[key]["value"]

return simpleDict

1. Update Test with Results
   1. End point: POST /data/devicedata
   2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
   3. Query Params: None
   4. Request Body: JSON of the updated data

**{**

**"deviceDataId": "{TestInfoDeviceDataId}",**

**"deviceDataModelId": "{TestInfoDeviceModelId}",**

**"devicePropertySetId": "{TestInfoDevicePropSetId}",**

**"data": {**

**"TestResult": "string",**

**"TestEndDate": "2021-05-06T14:42:38.395Z",**

**"TestComplete": true,**

**}**

**}**

* 1. Response Header: None
  2. Response Body: The deviceDataId of the updated record (the same that was passed in)
  3. Example (Python – upper case variables are constants/configuration values):

def uploadTestResults(token, deviceDataId, testResult, testDate):

url = BASE\_URL + "/data/devicedata"

headers = {

"X-APP-TYPE": APP\_TYPE,

"X-TENANT-DOMAIN": TENANT\_DOMAIN,

"X-API-VERSION": API\_VERSION,

"Content-Type": "application/json",

"Authorization": token

}

deviceData = {}

deviceData["TestResult"] = testResult

deviceData["TestEndDate"] = testDate.strftime("%Y-%m-%dT%H:%M:%S")

deviceData["TestComplete"] = true

payload = {

"deviceDataId": deviceDataId,

"deviceDataModelId": TEST\_INFO\_DEVICE\_MODEL\_ID,

"devicePropertySetId": TEST\_INFO\_DEVICE\_PROP\_SET\_ID,

"data": deviceData

}

response = requests.post(url, headers=headers, data=json.dumps(payload))

if response.status\_code != 201:

raise Exception("Could not post test result - " + str(response.text))

1. Send Heart-Beat
   1. End point: POST /data/devicedata
   2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
   3. Query Params: None
   4. Request Body: JSON of the updated data

**{**

**"deviceDataModelId": "{ReaderInfoDeviceModelId}",**

**"devicePropertySetId": "{ReaderInfoPropertySetId}",**

**"data": {**

**"ReaderId": "string",**

**"HeartbeatReceivedOn": "2021-05-06T14:22:38.395Z",**

**"Status": "string"**

**}**

**}**

* 1. Response Header: None
  2. Response Body: The deviceDataId of the updated record
  3. Example (Python – upper case variables are constants/configuration values):

def sendHeartBeat(token, readerId, status):

url = BASE\_URL + "/data/devicedata"

headers = {

"X-APP-TYPE": APP\_TYPE,

"X-TENANT-DOMAIN": TENANT\_DOMAIN,

"X-API-VERSION": API\_VERSION,

"Content-Type": "application/json",

"Authorization": token

}

deviceData = {}

deviceData["ReaderId"] = readerId

deviceData["HeartbeatReceivedOn"] = datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

deviceData["Status"] = status

payload = {

"deviceDataModelId": READER\_INFO\_DEVICE\_MODEL\_ID,

"devicePropertySetId": READER\_INFO\_DEVICE\_PROP\_SET\_ID,

"data": deviceData

}

response = requests.post(url, headers=headers, data=json.dumps(payload))

if response.status\_code != 201:

raise Exception("Could not post heart beat - " + str(response.text))

1. Upload Raw Data for Test
   1. End point: POST /data/devicedata
   2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
   3. Query Params: None
   4. Request Body: JSON of the updated data

**{**

**"deviceDataModelId": "{ReaderRawDeviceModelId}",**

**"devicePropertySetId": "{ReaderRawPropertySetId}",**

**"data": {**

**"ReaderId": "string",**

**"CollectorId": "string",**

**"TestDate": "2021-05-06T14:22:38.395Z"**

**"Measurement": {},**

**"Temperature": {},**

**"FluidFill": {},**

**"CurveFit": {},**

**"ResultQc": "string",**

**"FirmwareVersion": "string"**

**}**

**}**

* 1. Response Header: None
  2. Response Body: The deviceDataId of the updated record
  3. Example (Python – upper case variables are constants/configuration values):

def sendRawData(token, readerId, collectorId, testDate, measurement, temperature, fluidFill, curveFit, resutQc, firmwareVersion):

url = BASE\_URL + "/data/devicedata"

headers = {

"X-APP-TYPE": APP\_TYPE,

"X-TENANT-DOMAIN": TENANT\_DOMAIN,

"X-API-VERSION": API\_VERSION,

"Content-Type": "application/json",

"Authorization": token

}

deviceData = {}

deviceData["ReaderId"] = readerId

deviceData["CollectorId"] = collectorId

deviceData["TestDate"] = testDate.strftime("%Y-%m-%dT%H:%M:%S")

deviceData["Measurement"] = measurement

deviceData["Temperature"] = temperature

deviceData["FluidFill"] = fluidFill

deviceData["CurveFit"] = curveFit

deviceData["ResultQc"] = resutQc

deviceData["FirmwareVersion"] = firmwareVersion

payload = {

"deviceDataModelId": READER\_RAW\_DEVICE\_MODEL\_ID,

"devicePropertySetId": READER\_RAW\_DEVICE\_PROP\_SET\_ID,

"data": deviceData

}

response = requests.post(url, headers=headers, data=json.dumps(payload))

if response.status\_code != 201:

raise Exception("Could not post raw data - " + str(response.text))

**APIs for Web Portal**

* + - 1. Login

1. End point: POST /auth/login
2. Additional Request Headers: None (check common info section for common headers)
3. Query/Form Params: emailAddress, password (both form parameters)
4. Request Body: None
5. Response Header: auth token in header under header name Authorization. The auth token is in Bearer format
6. Response Body: JSON User object

{

"userId": "abcdef",

"firstName": "Abbas",

"middleName": null,

"lastName": "Dhilawala",

"emailAddress": "abbas@galendata.com",

"contactInfo": {

"streetAddress1": null,

"streetAddress2": null,

"city": null,

"state": null,

"country": null,

"zipcode": null,

"primaryPhone": null,

"secondaryPhone": null,

"primaryPhoneVerified": false

}

"gender": "Male",

"dateOfBirth": "01/01/1900",

"height": 0.0,

"weight": 0.0,

"patientId": null,

"status": "Active",

"fullName": "Abbas Dhilawala"

}

1. Example (Curl):

curl -X POST \

https://api.galencloud.com/auth/login \

-H 'Content-Type: application/x-www-form-urlencoded' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d 'emailAdderess=abbas%40galendata.com&password=MyPassword'

* + - 1. Search for Patient

1. End point: GET /user/user
2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
3. Query/Form Params: emailAddress
4. Request Body: None
5. Response Header: None
6. Response Body: JSON Page of matching User objects

{

"content": [

{

"userId": "abcdef",

"firstName": "Abbas",

"middleName": null,

"lastName": "Dhilawala",

"emailAddress": "abbas@galendata.com",

"contactInfo": {

"streetAddress1": null,

"streetAddress2": null,

"city": null,

"state": null,

"country": null,

"zipcode": null,

"primaryPhone": null,

"secondaryPhone": null,

"primaryPhoneVerified": false

}

"gender": "Male",

"dateOfBirth": "01/01/1900",

"height": 0.0,

"weight": 0.0,

"patientId": null,

"status": "Active",

"fullName": "Abbas Dhilawala"

}

],

"totalPages": 1,

"totalElements": 1,

"last": true,

"size": 2147483647,

"number": 0,

"numberOfElements": 1,

"first": true,

"empty": false

}

1. Example (Curl):

curl -X GET \

'https://api.galencloud.com/user/user?emailAddress=abbas.dhilawala@galendata.com' \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com'

* + - 1. Create Patient

1. End point: POST /user/user
2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
3. Query/Form Params: None
4. Request Body: JSON of the user information (for userId use a uuid construct)

{

"user":

{

"userId": "abcdef",

"firstName": "Abbas",

"middleName": null,

"lastName": "Dhilawala",

"emailAddress": "abbas@galendata.com",

"contactInfo": {

"streetAddress1": null,

"streetAddress2": null,

"city": null,

"state": null,

"country": null,

"zipcode": null,

"primaryPhone": null,

"secondaryPhone": null,

"primaryPhoneVerified": false

},

"gender": "Male",

"dateOfBirth": "01/01/1900",

"roles": [

{

"role": "Patient",

"practice": {

"practiceId": "mypractice-1"

},

"defaultRole": true

}

]

},

"password": "MyPassword1",

"customData": [

{

"field": {

"fieldId":"0bd60ff9-cb92-4b75-8ae1-036cc4991d35"

},

"fieldData": "Asian"

},

{

"field": {

"fieldId":"75b54a97-9f94-42f7-a18d-4233c45332b0"

},

"fieldData": "Not Hispanic or Latino"

}

]

}

1. Response Header: None
2. Response Body: None
3. Example (Curl):

curl -X POST \

https://api.galencloud.com/user/user \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'Content-Type: application/json' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d '{

"user": {

"userId": "abcdef",

"firstName": "Abbas",

"middleName": null,

"lastName": "Dhilawala",

"emailAddress": "abbas@galendata.com",

"contactInfo": {

"streetAddress1": null,

"streetAddress2": null,

"city": null,

"state": null,

"country": null,

"zipcode": null,

"primaryPhone": null,

"secondaryPhone": null,

"primaryPhoneVerified": false

},

"gender": "Male",

"dateOfBirth": "01/01/1900",

"roles": [{

"role": "Patient",

"practice": {

"practiceId": "mypractice-1"

},

"defaultRole": true

}]

},

"password": "MyPassword1",

"customData": [{

"field": {

"fieldId": "0bd60ff9-cb92-4b75-8ae1-036cc4991d35"

},

"fieldData": "Asian"

},

{

"field": {

"fieldId": "75b54a97-9f94-42f7-a18d-4233c45332b0"

},

"fieldData": "Not Hispanic or Latino"

}

]

}'

* + - 1. Associate Patient to Collector

1. End point: POST /data/devicedata
2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
3. Query/Form Params: None
4. Request Body: Device data with userId and collectorId and some meta data (values in {{}} are to be replaced by actual values)

{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertySetId": "13867531-fcb9-4ccf-bc12-3a53f4385117",

"ownerId": "{{USERID OF PATIENT}}",

"data": {

"CollectorId": "{{ENTERED COLLECTOR ID}}",

"SampleCollectedOn": "{{CURRENT DATE IN UTC IN FORMAT yyyy-MM-ddTHH:mm:ss.SSSZ}}",

"SampleCollectedBy": "{{USERID OF CURRENT USER}}"

}

}

1. Response Header: None
2. Response Body: identifier of the record
3. Example (Curl):

curl -X POST \

https://api.galencloud.com/data/devicedata \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'Content-Type: application/json' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d '{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertySetId": "13867531-fcb9-4ccf-bc12-3a53f4385117",

"ownerId": "{{USERID OF PATIENT}}",

"data": {

"CollectorId": "{{ENTERED COLLECTOR ID}}",

"SampleCollectedOn": "{{CURRENT DATE IN UTC IN FORMAT yyyy-MM-ddTHH:mm:ss.SSSZ}}",

"SampleCollectedBy": "{{USERID OF CURRENT USER}}"

}

}'

* + - 1. Associate Reader to Collector (Start Test)

1. End point: POST /data/devicedata
2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
3. Query/Form Params: None
4. Request Body: Device data with collectorId, readerId and some meta data (values in {{}} are to be replaced by actual values)

{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertySetId": "13867531-fcb9-4ccf-bc12-3a53f4385117",

"data": {

"CollectorId": "{{ENTERED COLLECTOR ID}}",

"ReaderId": "{{ENTERED READER ID}}",

"TestStartDate": "{{CURRENT DATE IN UTC IN FORMAT yyyy-MM-ddTHH:mm:ss.SSSZ}}",

"TestRunBy": "{{USERID OF CURRENT USER}}",

"TestComplete": false

}

}

1. Response Header: None
2. Response Body: identifier of the record
3. Example (Curl):

curl -X POST \

https://api.galencloud.com/data/devicedata \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'Content-Type: application/json' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d '{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertySetId": "13867531-fcb9-4ccf-bc12-3a53f4385117",

"ownerId": "{{USERID OF PATIENT}}",

"data": {

"CollectorId": "{{ENTERED COLLECTOR ID}}",

"ReaderId": "{{ENTERED READER ID}}",

"TestStartDate": "{{CURRENT DATE IN UTC IN FORMAT yyyy-MM-ddTHH:mm:ss.SSSZ}}",

"TestRunBy": "{{USERID OF CURRENT USER}}",

"TestComplete": false

}

}'

* + - 1. Search for Patient given a collector Id

1. End point: POST /data/devicedata-advanced/owner
2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
3. Query/Form Params: None
4. Request Body: Json with deviceDataModelId, deviceCriteria, devicePropertyCode values

{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertyCodes": ["CollectorId"],

"deviceCriteria": [{

"key": "CollectorId",

"operator": "Equal",

"value": "{{ENTERED COLLECTOR ID}}"

}]

}

1. Response Header: None
2. Response Body: Page of device data (should only be one record), with the owner information

{

"empty": true,

"first": true,

"last": true,

"number": 0,

"numberOfElements": 1,

"size": 1,

"totalElements": 1,

"totalPages": 1,

"content": [

{

"deviceDataId": "1234

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"ownerId": "user123",

"owner": {

"userId": "user123",

"firstName": "Abbas",

"middleName": "null,

"lastName": "Dhilawala",

"emailAddress": [abbas@galendata.com](mailto:abbas@galendata.com),

"gender": "Male",

"dateOfBirth": "01/01/1900"

},

"data": {

"CollectorId": {

"value": "string"

}

}

}

]

}

1. Example (Curl):

curl -X POST \

https://api.galencloud.com/data/devicedata-advanced \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'Content-Type: application/json' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d '{

"deviceDataModelId": "edfcab65-18d6-419d-abdc-86d413298b1a",

"devicePropertyCodes": ["ReaderId", "CollectorId"],

"deviceCriteria": [{

"key": "CollectorId",

"operator": "Equal",

"value": "{{ENTERED COLLECTOR ID}}"

}]

}

'

* + - 1. Search for Reader status given a collector Id
  1. End point: POST /data/devicedata-advanced
  2. Additional Request Headers: Auth token in Bearer format (Auth token available as a result of login)
  3. Query/Form Params: None
  4. Request Body: Json with deviceDataModelId, deviceCriteria, devicePropertyCode values

{

"deviceDataModelId": "de4769bd-3226-40a2-91ab-4d585c1a307c",

"devicePropertyCodes": ["ReaderId", "Status", "HeartbeatReceivedOn"],

"deviceCriteria": [{

"key": "ReaderId",

"operator": "Equal",

"value": "{{ENTERED READER ID}}"

}]

}

* 1. Response Header: None
  2. Response Body: Page of device data (should only be one record), with the owner information

{

"empty": true,

"first": true,

"last": true,

"number": 0,

"numberOfElements": 1,

"size": 1,

"totalElements": 1,

"totalPages": 1,

"content": [

{

"deviceDataId": "1234

"deviceDataModelId": "de4769bd-3226-40a2-91ab-4d585c1a307c",

"ownerId": "practice123",

"data": {

"ReaderId": {

"value": "string"

},

"Status": {

"value": "string"

},

"HeartbeatRecivedOn": {

"value": "2021-05-06T14:22:38.395Z"

},

}

}

]

}

* 1. Example (Curl):

curl -X POST \

https://api.galencloud.com/data/devicedata-advanced \

-H 'Authorization: Bearer {{TOKEN}}' \

-H 'Content-Type: application/json' \

-H 'X-API-VERSION: 3' \

-H 'X-APP-TYPE: DEVICE' \

-H 'X-TENANT-DOMAIN: aptitudedev.galencloud.com' \

-d '{

"deviceDataModelId": "de4769bd-3226-40a2-91ab-4d585c1a307c",

"devicePropertyCodes": ["ReaderId", "Status", "HeartbeatReceivedOn"],

"deviceCriteria": [{

"key": "ReaderId",

"operator": "Equal",

"value": "{{ENTERED READER ID}}"

}]

}

'

* + - 1. Search for Test Results
         1. TBD
      2. Update Test Results
         1. TBD
      3. Bulk Upload Results
         1. TBD

# Branding Guidelines

1. Fonts: Open Sans for body, and Lato for Headings
2. Colors
   1. Primary: Blue (#368AB6)
   2. Accent 1: Light Gray (#D1D3D4)
   3. Accent 2: Dark Gray (#2A2A2A)