Department of Computer Science and Engineering, University of Nevada – Reno

# Gigafactory Systems Machine Learning Project

Team 15: Adam Cassell, Braeden Richards, Ashlee Ladouceur

Project Part 3: Acceptance Criteria and Testing Strategy and Plan

Instructors: Sergiu Dascalu, Devrin Lee

External advisor(s): Dr. Emily Hand (UNR), Gavin Hall (Tesla)

8 March 2019

# Table of Contents

**TABLE OF CONTENTS .......... 2**

**ABSTRACT .......... 3**

**PROJECT UPDATES AND CHANGES .......... 4**

**USER STORIES AND ACCEPTANCE CRITERIA .......... 5**

USER INTERFACE SUBSYSTEM .......... 5
PYTHON BACKEND SUBSYSTEM .......... 6
MYSQL SERVER SUBSYSTEM .......... 7

**TESTING WORKFLOW .......... 8**

HAPPY PATH WORKFLOWS .......... 8
UNHAPPY PATH WORKFLOWS .......... 8

**TESTING STRATEGY .......... 10**

TESTING OUTLINE .......... 10
TESTING RESPONSIBILITIES .......... 10
DEFECT REPORTING .......... 11
PROJECT COMPLETENESS .......... 11
TEST PLAN .......... 11

**TEAM CONTRIBUTIONS .......... 13**

## Abstract

Team 15 is creating a machine learning system that provides real-time and offline analysis of Tesla's Gigafactory systems. The pipeline of the system design includes building an interface for users to connect to Tesla's mySQL database containing both historical and current data of sensors in the factory, building models, training and testing those models, and building a visual analysis of the data received from the training and testing. The team plans to implement a Python backend, a SQL server to hold both Gigafactory and user data, and a simple Interface to interact with the system.

## Project Updates and Changes

Team 15 has made significant developmental progress in order to deliver a working deliverable for their scheduled progress demo on March 14, 2019. The development has been split between the three members equally.  Adam worked on cleaning up the frontend for ease of user interaction. In addition, Adam worked on setting up a working model library and data inspection tab on the frontend that uses model information derived from the Python backend. The model library can now properly load objects from the backend, and once a model from the library is loaded, the data inspection tab is accurate according to data of that model. Braeden developed Python functions for preprocessing data from either csv's or excel files. Braeden also worked with Adam to create the communication between the backend and frontend. Finally, Ash created the functions in the Python script that take preprocessed data and train it with the Keras library. The models trained and tested from those functions are then saved into a folder that is referenced by the frontend for the model library on the frontend.

Communication between the frontend and backend has finally been achieved by using GET and POST methods. This cleared a major roadblock for the GigaML application. GET/POST methods have been chosen for ease of development with Flask. Instead of using forms on the frontend to send data through these methods, GigaML is using Ajax with JQuery to communicate since it is easier to use JavaScript functions in conjunction with Ajax than with forms.

The team did not require any modifications to the design or specification. The development is closely following previously planned development expectations that follow the already documented specifications in regard to expectations of deliverables. The team is writing multiple tests as they develop the backend in order to catch any potential changes needed in design. This will ensure the team does not progress too far into development before a major design overhaul is needed.

# User Stories and Acceptance Criteria

## User Interface Subsystem

- As a user or admin, I want to visualize model data on graphs so that I can better analyze trends over time.
    - AC1. User or admin is logged in with appropriate permissions given by system.
    - AC2. User or admin can select which model they want to visualize on the system from the "Model Library" landing page.
    - AC3. User or admin is immediately directed to the "Performance" tab.
    - AC4. The "Performance" tab displays the correct graph on the screen that plots model outputs over its trained time.
    - AC5. User or admin can select or deselect plotted lines on the graph for visualizing.
    - AC6. User or admin can change between models within the model library and the "Performance" page will update according to the selected model automatically.
- As a user or admin, I want to view the individual model architecture so that I can understand the data more clearly.
    - AC1. User or admin is logged in with appropriate permissions given by system.
    - AC2. User or admin can select which model they want to visualize on the system from the "Model Library" landing page.
    - AC3. User or admin is able to click the "Summary" tab from the "Performance" page of the system.
    - AC4. User or admin can see accurate model architecture on the "Summary" tab, including model parameters, training parameters, and all input and output names, types, and data bounds.
    - AC5. User or admin can change between models within the model library and the "Performance" page will update according to the selected model automatically.
- As a user or admin, I want to view system information so that I can understand where the data I am analyzing is being sourced.
    - AC1. User or admin is logged in with appropriate permissions given by system.
    - AC2. User or admin can select which model they want to visualize on the system from the "Model Library" landing page.
    - AC3. User or admin can select the "View System Information" link at the top-left of any tab.
    - AC4. The system information is displaying in a detailed information box that details accurate information about where the current model data is sourced, including what sensors the model tested, dependencies, parents, and links to factory system details about this sensor data for interpretation.
    - AC5. User or admin can click the "X" button at the top-right of the information box to exit to the original tab that the user or admin was on.
    - AC6. User or admin can change between models within the model library and the "View System Information" link will update according to the selected model automatically.
- As a user or admin, I want to view a system user guide so that I can understand how to use the system.
    - AC1. User or admin is logged in with appropriate permissions given by system.
    - AC2. User or admin can select which model they want to visualize on the system from the "Model Library" landing page.

    - AC3. User or admin can select the “GigaML User Guide” link at the top-right of any tab.
    - AC4. The system displays a PDF of how to use the system.
    - AC5. User or admin can click the “X” button at the top-right of the information box to exit to the original tab that the user or admin was on.
- As an admin, I want to input model information such as the inputs, input/output type, names, data bounds, and activation function so that I can prepare a model for training and testing.
    - AC1. Admin is logged in with appropriate permissions given by system.
    - AC2. Admin can select the “Create Model” button on the “Model Library” landing page.
    - AC3. Admin is directed to the “Create Model” page.
    - AC4. Admin can enter and select model information such as parameters, data source, inputs, and outputs.
    - AC5. Admin can click the “X” button at the top-right of the page to exit to the “Model Library” landing page without saving any information.
    - AC6. Admin can click the “Add to Model Library” button at the bottom-right of the page to save the model to the model library for training and testing. The page will exit to the “Model Library” landing page.
    - AC7. Admin or user can select that model to open within the system.

## Python Backend Subsystem

- As a user or admin, I want to check the accuracy of a model so that I can ensure my analysis from this data is reflecting Gigafactory performance correctly.
    - AC1. User or admin is logged in with appropriate permissions given by system.
    - AC2. User or admin can select which model they want to visualize on the system from the “Model Library” landing page.
    - AC3. User or admin is able to click the “Performance” tab from the “Details” page of the system.
    - AC4. User or admin can see accuracy data calculated by the Python backend for the current model on the tab.
    - AC5. User or admin can change between models within the model library and the “Inspect Data” page will update according to the selected model automatically.
- As an admin, I want to enter a model into training and testing so that I can add more models into the model library for analysis.
    - AC1. Admin is logged in with appropriate permissions given by system.
    - AC2. Admin has entered model data that has been saved in the model library.
    - AC3. Admin can select which model they want to train and test from the “Model Library” landing page.
    - AC4. Admin is able to click the “Train” tab from the “Performance” page of the system.
    - AC5. Admin is able to enter the selected model into training from the “Train” page.
    - AC6. Admin is displayed a progress bar of training and testing progress.
    - AC7. Admin is displayed a real-time plot of training error to monitor.
    - AC8. Admin is able to update the model data once training and testing is complete.
    - AC9. If Admin clicks the “X” button at the top-right of the page during training, the model training and testing data will be discarded.
    - A10. Admin can click the “X” button at the top-right of the page to exit to the “Performance” page without saving any information.
    - A11. Admin can click “Update Model” after training is complete and the page will exit to the “Performance” page with updated information accurately displayed on all pages.

## MySQL Server Subsystem

- As a user or admin, I want to sign into the system so that I can view model information.
    - AC1. User or admin will see a screen to enter their Tesla SID (user ID for employees) upon starting the system application.
    - AC2. User or admin can use keyboard to type in their SID into the input box.
    - AC3. User or admin can select "Login" and the system will check with the server data to determine if the user or admin has user/admin privileges.
    - AC4. User or admin will successfully be given the "Model Library" landing page upon a successful login.
    - AC5. User or admin will be given an error regarding their SID if their data is not in the server data.
    - AC6. User or admin will be given the appropriate user interface experience determined upon their privileges.
    - AC7. User or admin can click "Logout" at the top-right of any tab to be redirected to the "Login" page.
- As an admin, I want to view raw sensor data from Ignition so that I can ensure Ignition is sending correct information to the system.
    - AC1. Admin is logged in with appropriate permissions given by system.
    - AC2. Admin can select which model they want to visualize on the system from the "Model Library" landing page.
    - AC3. Admin can select the "View System Information" link at the top-left of any tab.
    - AC4. Admin has the option to select "Download Raw Data" within the information box.
    - AC5. A CSV file of the raw model information used from the Ignition server will automatically download on the system the Admin is using.
    - AC6. Admin can change between models within the model library and the "Download Raw Data" link will update according to the selected model automatically.

# Testing Workflow

## Happy Path Workflows

### *Happy Path Workflow 1*

1. Choose data file
2. Click Data Inspection Tab
3. View basic data about the dataset (standard deviations, means, etc.)
4. Click dropdown to change to new data file
5. View basic data about new dataset

**Validation**: No file path errors, data files read without errors, data displayed correctly on webapp

### *Happy Path Workflow 2*

1. Choose system to work with
2. Chose which model within the system to load
3. View Graph of accuracy of the model with past data
4. Test the accuracy of the model with newer data
5. Switch to sandbox mode of model
6. Enter numbers to test outputs
7. Close webapp

**Validation**: No file path errors, proper displaying of available models to choose, data sent to frontend without errors, outputs correct based on inputs

### *Happy Path Workflow 3*

1. Choose data file
2. Go to preprocessing tab
3. Normalize some of the columns between 0 and 1
4. Check amount of N/A pieces of data in columns
5. Choose some preprocessing selections to test dependability
6. Remove the columns with many N/A data pieces to improve accuracy
7. Save data for use with training model

**Validation**: No file path errors, preprocessing tab opens, column removal works, saves file successfully

## Unhappy Path Workflows

### *Unhappy Path Workflow 1*

1. When opening data inspection page, the file path passed can't be found
2. Is this error displayed to the user?
3. Does the tab still open?
4. Does the tab allow for retrying file find?

**Validation**: Error is displayed to user, error is displayed on console, tab opens/doesn't crash

### *Unhappy Path Workflow 2*

1. When loading the data inspection page, there are over 50 data attributes for each data object
2. Can the table on the page handle loading that many attributes in the table?
3. Does the table fit on the page?
4. Does the table get broken into multiple pages?
5. When you close and reopen the page does it save the data in memory for quick loading?

**Validation**: Table breaks into smaller tables with pages, table fits on page, data stored in JavaScript variable and therefore reloads quickly

### *Unhappy Path Workflow 3*

1. Choose System and Model from dropdown menu
2. Click on Predict tab
3. Model did not load correctly, does an error show to user?
4. Do the sandbox inputs show an empty default look?
5. Does the page/tab still open without breaking?
6. Does the page allow for refreshing/retrying to gather the data?

**Validation**: All systems and their models show, predict tab shows error with model loading, sandbox inputs show default values to signify errors, tab opens without breaking, reopening tab reattempts loading

# Testing Strategy

## Testing Outline

Team 15 is implementing a robust testing strategy to ensure quality development of GigaML. The Test Automation Pyramid, as defined by Robert C. Martin in chapter 8 of *The Clean Coder*, will be used as a baseline guide for the testing strategy.

At the lowest level, unit tests will be written for the basic functions in both the Python backend codebase as well as the JavaScript frontend codebase. The primary purpose of these tests is to cover the unhappy-path cases that can occur, especially when it comes to data processing and transfer functions. Tests will be written to verify calculations performed by these functions in the backend. Tests will similarly be written for the JavaScript to verify that data is presented to the user accurately once being retrieved from the backend.

The next key level of testing for Team 15 is automated acceptance tests in the form of component testing. GigaML has individual components that must each function nominally as independent pieces. The primary components are (1) Data Retrieval and Preprocessing Module, (2) Keras Model Creation/Training Module, (3) Flask web app hosting Module, and (4) Web frontend GUI module. Each of these components will have their overall outputs tested against automated test inputs to ensure the final result performs as expected. This will also by association double-check that all underlying functions within each module are working together as expected.

Next, team 15 will perform integration testing to ensure all components are working together as expected. For the first 3 modules (all part of the Python backend), this will involve testing whether data is passed from module to module in the correct, expected sequence. The passing of data from the Python backend to the JavaScript frontend is also key to test, to make sure the connection is reliable, efficient, and secure.

Finally, system tests will be performed to test the entire system on a given input and ensure that it provides an expected and reasonable output after being processed by all the modules together. Team 15 will also use any extra time to perform manual exploratory tests of the user interface.

## Testing Responsibilities

Table 1 breaks down testing responsibilities according the main system components. Note that these responsibilities include the associated unit and component testing for that system. Separate overall integration and system testing will be performed as a cooperative simultaneous effort by all developers every time significant changes are made to any of the components.

| Component | Primary Owner/Developer | Assigned Tester | When Testing Should Occur |
|---|---|---|---|
| **Data Retrieval and Preprocessing Module** | Braeden | Ash | Any time a new data processing or retrieval function is added or significantly modified. |
| **Keras Model Creation/Training Module** | Ash | Adam | Any time significant changes have been made to model creation, compilation, training or testing scripts. |

|  |  |  |  |
|---|---|---|---|
| **Flask web app hosting Module** | Braeden | Adam | Any time changes to Flask parameters are made that could influence frontend hosting |
| **Web frontend GUI module** | Adam | Braeden | Any time GUI is altered, or user interaction design is changed for a given feature |

Table 1: Testing responsibilities broken down.

## Defect Reporting

Any time issues are encountered at any level of testing, they will be added to a 'Bug Report' list on the project's GitHub page under that particular component. The owner/primary developer for that component listed in Table 1 will be responsible for ensuring those issues are resolved.

## Project Completeness

Project completeness is defined by all levels of testing successfully being completed as individual pieces, then as integrated components, and finally as an overall system. This means all unit tests, automated acceptance tests (including overall integration/system tests) need to pass. Acceptance testing is defined by stakeholder feature requests, thus the stakeholder (Tesla) will have expressed satisfaction upon the completion of these tests. Additionally, we will ensure the stakeholder is happy with the GUI through a final design review. This will ensure both form and function are to Tesla's liking, and thus considered 'complete' by the developers.

## Test Plan

| Test No. | Test Type | Test Name | Purpose of Test | Outcome and Actions Required |
|---|---|---|---|---|
| **1** | UI | Visualize model data on graphs | System accurately displays models via graphs | Plots are rendered correctly and accurately represent the data. No actions required. |
| **2** | UI | View the individual model architecture | Users can see accurate data of models on system | Model architecture is listed in the expected line-by-line text format and correctly representative of the model structure and parameters. No actions required. |
| **3** | UI | View system information | System displays accurate system details | High-level factory system information is correctly displayed in expected text format. No actions required. |
| **4** | UI | View a system user guide | Users can understand how to operate the system | User is linked to a thorough and educational user-guide page that is maintained on an internal wiki. No action required. |
| **5** | UI | Input model information | Users can modify inputs for models | Text boxes can be populated, dropdown options can be selected, and these pieces are all correctly displayed once entered. Individual example field entry is required for testing. |
| **6** | Backend | Check the accuracy of a model | Users can ensure system models are accurate | Model training and testing accuracy is correctly displayed on the screen in both overall numerical format as well as graphically. No action required. |

|  |  |  |  |  |
|---|---|---|---|---|
| **7** | Backend | Enter a model into training and testing | Users can choose to enter a model into training/testing | Backend script begins the training process and subsequent testing process. Training and testing complete without errors. No action required. |
| **8** | MySQL | Sign into the system | Users can access the system | User is authenticated to the system correctly by referencing a MySQL database of users and loading or not loading admin features depending on the user's permissions. It is required to test using both an admin and a non-admin user id. |
| **9** | MySQL | View raw sensor data | Users can export data into csv or excel file | A csv or excel file is downloaded for the user correctly displaying raw sensor data over the specified time interval. |
| **10** | UI | Successful preprocessing of data | Ensure data integrity for training/testing | Data is correctly updated onscreen to reflect the application of preprocessing on data. No action required. |
| **11** | Backend | File read correctly | Error shown and system doesn't crash | Input file is correctly loaded and is visible and able to be parsed by the preprocessing functions in the backend. No action required. |
| **12** | UI | Tables correctly format | Ensure user can read information | Table on data inspection page is properly rendered and displays the full range of expected statistics in the expected format. Not action required. |

Table 2: Test plan that describes acceptance criteria.

## Team Contributions

Team 15 worked together to create the Acceptance Criteria and Testing Strategy and Plan document with the understanding of group effort. A one-hour long meeting was held between the team members prior to creating the document, so all members understood each section goal and outline project development. Table 3 is a representation of the time worked on each section:

| Team Member | Sections | Time Worked |
|---|---|---|
| **Adam Cassell** | Project Updates and Changes, Testing Strategy | 3 hours |
| **Ashlee Ladouceur** | Abstract, User Stories and Acceptance Criteria, Team Contributions | 3 hours |
| **Braeden Richards** | Project Updates and Changes, Testing Workflow | 3 hours |

Table 3: Detailed breakdown of time per section by each member of Team 15 for the Acceptance Criteria and Testing Strategy and Plan document.