ScoreCard Automation Project Documentation

In this document, you'll find Automation process details, troubleshooting and repair processes, as well as how to replicate the results by hand. The components of the process, how they are connected, and setup instructions will also be discussed. Moreover, you'll find the pre/post conditions, and examples, for all of the code used in the project.

- I. Built-In Functions
  - 1.1 ConfluenceAPI
  - 1.2 DCRumAPI
  - 1.3 DCRumHelpers
  - 1.4 NewRelicAPI
  - 1.5 GoogleSheetsAPI
  - 1.6 Helpers
- II. Interactivity & Design Methodology
  - Interactivity
  - Design Methodology
- III. Components & Setup
  - A. EC2 - Component Setup
  - B. RDS Component - Setup
  - C. Google Sheets Component - Setup
    - 1) Enable API Calls
    - 2) Create User Credentials
- IV. Creating a New Scorecard
- V. First Run Conditions
- VI. Troubleshooting, Known Issues, and Error Messages
  - PUT Status Code: 400
  - ValueError
  - Invalid Images
  - Next
- VII. RDS Database Schema
  - 1) APPLICATION_FrontPage
  - 2) APPLICATION_FrontPageGraph
- VIII. Developer Account Dependencies

I. Built-In Functions

1.1 ConfluenceAPI

- Included global variables in the module (you'll need to set these up)
  - url :: Path to the root directory of the Confluence server (Ex: 'http://13.59.88.253:8090')
  - user :: Confluence developer account username (you'll be modifying the server on this user's behalf)
  - pass :: The password associated with *user* in plaintext
  - SPACE_ID :: The ID of the space for the page set you're modifying with API calls
- Python 2.7.12 $*post(title)*
  - Pre-Con:
    - title :: The title that should be posted for the page you're trying to create a page for
  - Post-Con:
    - Creates a page under *SPAC_ID* named *title* and returns the JSON response
  - Example Usage:

```
id_ = ConfluenceAPI.post(name);
```

- Python 2.7.12 $*geta(id, file)*
  - Pre-Con:
    - id :: Page ID containing the attachment you're trying to query
    - file :: The name of the attachment you're requesting information about
  - Post-Con:
    - Returns a JSON response containing information about *file* on page with ID, *id*. See ConfluenceAPI Documentation.
  - Example Usage:

```
id_ = ConfluenceAPI.post(name);
modificationDate = ConfluenceAPI.geta(id_, "banner.png");
modificationDate = Helpers.unwrap(modificationDate,
"LastModeified");
```

- Python 2.7.12 $*put(id, path, title, version)*
  - Pre-Con:
    - id :: The ID of the page to modify
    - path :: The relative path to the file containing your payload for the PUT command
      - **Remark:** Functions such as `Helpers.imageTag(id, file)` use './update.txt' for writing.
    - title :: The title to give the attachment
    - Version :: The current version of the page being updated.
      - **Remark:** This must be managed manually. Versions must be incremented by one, as per Confluence Version Control requirements. Failure to increment this value by exactly one will result in a JSON transaction error.
  - Post-Con:
    - Returns the status code of the PUT request.
      - **Remark:** This does not append to what is already on the page, but rather it replaces it with the contents of *upda te.txt* located at *path*
  - Remarks:

      - *update.txt* should be deleted directly after this request as to not submit a duplicate of the request to the page on the next *put* request.
  - Example Usage:

```
ConfluenceAPI.put(id_, "update.txt" , "Scorecard Example",
version)
version += 1 #increment version control
```

- Python 2.7.12 $*dpost(id, path)*
  - Pre-Con:
    - id :: The ID of the page you want to upload an attachment to
    - path :: The relative path of your attachment file
  - Post-Con:
    - Uploads and attaches *path* to page with *id*
  - *Remarks:*
    - You'll still need to add the image to the page. This function only attaches the image to the page.
  - Example usage:

```
ConfluenceAPI.dpost(id_, "banner.png"); #Attach image to the page
Helpers.constructPayload(Helpers.imageTag(id_, "banner.png"));
#Add image to the page
```

- Python 2.7.12 $*get(title)*
  - Pre-Con:
    - title :: The name of the page you want to get information about
  - Post-Con:
    - Returns a JSON response with information about the page named *title* in pagespace *SPAC_ID*. See ConfluenceAPI Documentation.
  - Example Usage:
    - N/a at this time
- Python 2.7.12 $*getURL()*
  - Pre-Con:
    - N/a
  - Post-Con:
    - Returns **ConfluenceAPI::*url***
- Python 2.7.12 $delete*(id)*
  - Pre-Con:
    - id :: The ID of the page you wish to delete
  - Post-Con:

            - Deletes the page with ID *id* in pagespace *SPAC_ID* and returns the status code

1.2 DCRumAPI

- Included global variables in the module (you'll need to set these up)
    - user_ :: DCRum developer account username (you'll be querying the server on this user's behalf)
    - pass_ :: Password for *user_*
- Python 2.7.12 $GET_Data_to_Export_Ban()
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. This call returns the data traditionally stored in a file called "AK - ScoreCard Data to Export" for the Banner application.
    - Example Usage:
        - <u>This syntax is used for all functions in this module if no example is given</u>

```
DCRumAPI.GET_Data_to_Export_Ban();
```

- Python 2.7.12 $*GET_FontPage_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing all of the data used in the front page data table for the Banner application.
- Python 2.7.12 $*GET_Top_Ten_Slow_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing all of the data used in the Top Ten Slow URLs table for the Banner application.
- Python 2.7.12 $*GET_Bucket_02_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing information about all URLs that took between 0 and 2 seconds, with more than 10 operations, to load for the Banner application.
- Python 2.7.12 $*GET_Bucket_23_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing information about all URLs that took between 2 and 3 seconds, with more than 10 operations, to load for the Banner application.
- Python 2.7.12 $*GET_Bucket_35_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing information about all URLs that took between 3 and 5 seconds, with more than 10 operations, to load for the Banner application.
- Python 2.7.12 $*GET_Bucket_58_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing information about all URLs that took between 5 and 8 seconds, with more than 10 operations, to load for the Banner application.
- Python 2.7.12 $*GET_Bucket_8_Ban()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response with raw data to be handled. It returns a string containing information about all URLs that took more than 8 seconds, with more than 10 operations, to load for the Banner application.

1.3 DCRumHelpers

- Python 2.7.12 $*handle(source_)*
    - Pre-Con:
        - source_ :: A JSON response from the **DCRumAPI**
    - Post-Con:
        - Removes the header from *source_*. That is, it removes characters up to and including, '"formattedData":['
    - Example Usage:

```
DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_Ban());
```

- Python 2.7.12 $*parseDataSet(source_, iteration)*
    - Pre-Con:
        - source_ :: A JSON response from the **DCRumAPI** that has been passed to **DCRumHelpers::handle(source_)**
        - iteration :: The index of the list you wish to isolate
    - Post-Con:
        - Remark that *source_* is a list of lists. This function isolates the list at indexiteration, and returns its contents as a single string.
    - Example Usage:

```
dataContainer = [];
for i in range(0, 10):
 dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.parse
DataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_Ban()),
i))); #Adds single elements from the JSON response into
dataContainer

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">")
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weight:bo
ld;\\\">" + str(float(dataContainer[0].pop(0))) +
"</td>");#Remove and return the first element in the list
```

- Python 2.7.12 $Tokenize(source_)
    - Pre-Con:
        - source_ :: A JSON response from **DCRumAPI::GET_Data_to_Export_Ban()** that has been passed to **DCRumHelpers::handle(source_)**
    - Post-Con:
        - Converts *source_* into a *list* object where each index holds a piece of data correlating to the columns of the AWS RDS table mentioned at **VII**. If *source_* is empty, an empty *list* is returned.
    - Example Usage:

```
query = DCRumHelpers.handle(DCRumAPI.GET_Data_to_Export_Ban())
dataContainer =
DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query, 0));
```

- Python 2.7.12 $*Tokenize_7(source_)*
    - Pre-Con:
        - source_ :: A JSON response from **DCRumAPI::GET_FrontPage_Ban()** that has been passed to **DCRumHelpers:parseDataSet(source_, iteration)**, and **DCRumHelpers::handle(source_)**.
    - Post-Con:
        - Converts *source_* into a *list* object where each index holds a piece of data correlating to the following list of attributes: Availability, Application Performance, Average Operation Time, Total Operations, Slow Operations, and Unique Users.
    - Example Usage:

```
dataContainer =
DCRumHelpers.Tokenize_7(DCRumHelpers.parseDataSet(DCRumHelpers.h
andle(DCRumAPI.GET_FrontPage_Ban()), 0));
aval = str(round(float(dataContainer[0]), 2));
```

- Python 2.7.12 $*Tokenize_3(source_)*
  - Pre-Con:
    - source_ :: A JSON response from **DCRumAPI::GET_Top_Ten_Slow_Ban()** that has been passed to **DCRumHelpers::parseDataSet(source_, iteration)**, and **DCRumHelpers::handle(source_)**.
  - Post-Con:
    - Converts *source_* into a *list* object where each index holds a piece of data correlating to the following list of attributes: Operation, Count, Time. For use in constructing the Top Ten Slow Performing URLs Table.
  - Example Usage:

```
dataContainer = [];

for i in range(0, 10):
 dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.parse
DataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_Ban()),
i)));

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\\\">" +
str(dataContainer[0].pop(0)) + "</td>");#Remove and return the
first element in the list
```

- Python 2.7.12 $*execute_BucketList_Query(response_)*
  - Pre-Con:
    - response_ :: A JSON response from an application Bucketlist request found in **DCRumAPI**
  - Post-Con:
    - Returns a list containing the data to be entered into the Bucketlist for the passed query, that is, URLs in index zero, and count in index one.
  - Example Usage:

```
query =
DCRumHelpers.execute_BucketList_Query(DCRumAPI.GET_Bucket_02_Ban
());
url_02 = GoogleSheetsAPI.GS_Put('1', str(len(query)), 'A', 'A',
'USER_ENTERED', query, 'ROWS', 0); #Add data to Google Sheets
Document (Discussed in II::3.D)
```

1.4 NewRelicAPI

- Included global variables in the module (you'll need to set these up)
  - queryKey :: Key generated from New Relic that allows you to use the Insight API calls with your instance. See this page for key generation.
- Python 2.7.12 $*Application_Performance()*
  - Pre-Con:

        - N/a
    - Post-Con:
        - Returns a JSON response containing Flashline Application Performance
    - Example Usage:

```
app_perf = Helpers.unwrap(NewRelicAPI.Application_Performance(),
':[{"result');
cur.execute(
 "INSERT INTO Flashline_FrontPage(
  ID,
  dataDescription,
  dataValue,
  dataDate
 )
 VALUES(
  0,
  'Application Performance',
  '" + app_perf + "'," +
(datetime.datetime.today()-datetime.timedelta(days=7)).strftime(
'%Y%m%d')+
 ");"
);#upload to RDS
```

- Python 2.7.12 $*Average_Operation_Time()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response containing Flashline Average Operation Time
    - Example Usage:

```
Helpers.unwrap(NewRelicAPI.Average_Operation_Time(), 'average');
```

- Python 2.7.12 $*Total_Page_Views()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response containing Flashline Total Number of Operations
    - Example Usage:

```
total_ops = Helpers.unwrap(NewRelicAPI.Total_Page_Views(),
'count');
```

- Python 2.7.12 $*SlowOps()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response containing Flashline Total Page Views
    - Example Usage:

```
slow_ops = Helpers.unwrap(NewRelicAPI.SlowOps(), 'count');
```

- Python 2.7.12 $*Unique_Users()*
    - Pre-Con:
        - N/a
    - Post-Con:

        - Returns a JSON response containing Flashline Unique Users
    - Example Usage:

```
uniq_usr = Helpers.unwrap(NewRelicAPI.Unique_Users(),
'uniqueCount');
```

- Python 2.7.12 $*Slow_URLs()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response containing the information needed to construct the Flashline Top Ten Slow URLs list
    - Example Usage:

```
dataContainer = Helpers.buildStack_3(NewRelicAPI.Slow_URLs(),
10);
dataContainer.pop(0); #Remove sentenial

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");

Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\\\">" +
dataContainer.pop(0) + "</td>");

Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weight:bo
ld;\\\">" + str(round(float(dataContainer.pop(0)), 2)) +
"</td>");#Remove and return the first element in the list

#etc
```

- Python 2.7.12 $Transaction_Perf()
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a JSON response containing the information needed to construct the Flashline Transaction Performance table
    - Example Usage:

```
dataContainer = Helpers.buildStack_7(NewRelicAPI.Top_Slow(), 25);
dataContainer.pop(0); #Remove sentinel value

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");

Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\\\">" +
str(dataContainer.pop(0)) + "</td>");#Remove and return the first
element in the list
```

- <u>The *_Plus() functions found in this file are not used. They return extraneous information in their JSON response that is not needed for the scorecard. They were included for completeness.</u>

1.5 GoogleSheetsAPI

- Included global variables in the module (you'll need to set these up)
  - The *.credentials* directory
    - You will need to ensure all of the proper permissions have been set up for the code to access Google Sheets on your behalf. This is done in the Google API Console. Documentation for this procedure can be found here. Additionally, you will have to enter the python executable on your environment and execute a function from the module, and follow the instructions there to generate a credential file. More specific instructions are discussed in **III::C**.
  - client_secret.json
    - Follow this guide to acquire the *client_secret.json*, and place it in the same directory as the executing python script. In this case it will be both the *Create_*_Bucketlist.py* at */home/ec2-user/*\ Scorecard/Bucketlist/Drive* and *Build_*_Bucketlist.py* at */home/ec2-user/*\ Scorecard/Bucketlist*.
- Python 2.7.12 $GS_Put(rrangeL_, rrangeU_, crangeL_, crangeR_, value_input_option_, payload_, majorDimension_, payload_Selector_)
  - Pre-Con:
    - rrangeL_ :: The row number to start data entry at
    - rrangeU_ :: The row number to end data entry at
    - crangeL_ :: The column letter to start data entry at
    - crangeR_ :: The column letter to end data entry at
    - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
    - payload_ :: A list of lists where the data for each cell is contained in the nested list. It is not required that the nested list contain only one element. *payload_Selector_*tells the function which column you wish to use. It is required that all of the nested lists are the same length
    - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
    - payload_Selector_ :: The index of the table, *payload_* you wish to pull data from. See the figure below.
  - Post-Con:
    - Places the data in the column *payload_Selector_* of *payload_* and places it into the range *crangeL_ rrangeL_ : crangeR_ rrangeU_* on the active Google Sheet.
  - Example Usage:

```
url_02 = GoogleSheetsAPI.GS_Put('1', str(len(query)), 'A', 'A',
'USER_ENTERED', query, 'ROWS', 0);
```

- Python 2.7.12 $GS_Put_Unique(rrangeL_, rrangeU_, crangeL_, crangeR_, value_input_option_, payload_, majorDimension_)
  - Pre-Con:
    - rrangeL_ :: The row number to start data entry at
    - rrangeU_ :: The row number to end data entry at
    - crangeL_ :: The column letter to start data entry at
    - crangeR_ :: The column letter to end data entry at
    - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
    - payload_ :: A string that contains a formula to be entered in the range defined by *rrangeL_, rrangeU_, crangeL_,* and *crangeR_*. Note that parameters of the formula cannot be changed relative to new cells if your range is larger than one cell. This function is designed for formulas that return ranges such as the *UNIQUE* formula.
    - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
  - Post-Con:
    - Places the formula into the range *crangeL_ rrangeL_ : crangeR_ rrangeU_* on the active Google Sheet.
  - Example Usage:

```
GoogleSheetsAPI.GS_Put_Unique('1', '1', 'C', 'C', 'USER_ENTERED',
'=UNIQUE(A:A)', 'ROWS');
```

- Python 2.7.12 $GS_Get(rrangeL_, rrangeU_, crangeL_, crangeR_, value_input_option_, majorDimension_)
  - Pre-Con:
    - rrangeL_ :: The row number to start data entry at
    - rrangeU_ :: The row number to end data entry at
    - crangeL_ :: The column letter to start data entry at
    - crangeR_ :: The column letter to end data entry at
    - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the

user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
        - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
    - Post-Con:
        - Returns a list where each indicy contains a list containing the value of the corresponding cell that was found to contain any value in the range *crangeL_ rrangeL_ : crangeR_ rrangeU_* on the active Google Sheet in unicode formatting.
    - Example Usage:

```
upper_02 = len(GoogleSheetsAPI.GS_Get('', '', 'C', 'C',
'USER_ENTERED', 'ROWS')); # get number of unique URLs
GoogleSheetsAPI.GS_Put_Sumif(1, upper_02, 'E', 'E', '1', url_02,
'USER_ENTERED', 'ROWS'); #Use the number of unique URLs to give
the next PUT request its range
```

- Python 2.7.12 $GS_Put_Sumif(rrangeL_, rrangeU_, crangeL_, crangeR_, sumRange_L, sumRange_U, value_input_option_, majorDimension)
    - Pre-Con:
        - rrangeL_ :: The row number to start data entry at
        - rrangeU_ :: The row number to end data entry at
        - crangeL_ :: The column letter to start data entry at
        - crangeR_ :: The column letter to end data entry at
        - sumRange_L :: The lower bound of the *sum_range*
        - sumRange_U :: the upper bound of the*sum_range*
        - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
        - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
    - Post-Con:
        - Places a *SUMIF* formula in range *crangeL_ rrangeL_ : crangeR_ rrangeU_* on the active Google Sheet where the *ifrang e* is *A sumRange_L : AsumRange_U*, the *criterion* is *C (row index relative to the current column)*, and the *sum_range* is *B sumRange_L : B sumRange_U* in accordance with the SUMIF documentation found here.
    - Remarks:
        - See **II::3.D.i** for more details
    - Example Usage:

```
GoogleSheetsAPI.GS_Put_Sumif(1, upper_02, 'E', 'E', '1', url_02,
'USER_ENTERED', 'ROWS');
```

- Python 2.7.12 $GS_Put_Sum(rrangeL_, rrangeU_, crangeL_, crangeR_, value_input_option_, majorDimension_)
    - Pre-Con:
        - rrangeL_ :: The row number to start data entry at
        - rrangeU_ :: The row number to end data entry at
        - crangeL_ :: The column letter to start data entry at
        - crangeR_ :: The column letter to end data entry at
        - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
        - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
    - Post-Con:
        - Places the following formula into the range *crangeL_ rrangeL_ : crangeR_ rrangeU_*, *=SUM(E*, F*, G*, H*, I*)* where * is the row index relative to the current column.
    - Remarks:
        - See **II::3.D.i** for more details
    - Example Usage:

```
GoogleSheetsAPI.GS_Put_Sum(1, upper_8, 'D', 'D', 'USER_ENTERED',
'ROWS');
```

- Python 2.7.12 $*GS_Put_Percent(rrangeL_, rrangeU_, crangeL_, crangeR_, num_Col, value_input_option_, majorDimension)*

- Pre-Con:
    - rrangeL_ :: The row number to start data entry at
    - rrangeU_ :: The row number to end data entry at
    - crangeL_ :: The column letter to start data entry at
    - crangeR_ :: The column letter to end data entry at
    - num_Col :: The column letter for the numerator desired when dividing by the total number of operations for the operation time grouping.
    - value_input_option_ :: Must be *'USER_ENTERED'* or *'RAW'*. 'USER_ENTERED' will allow the input to behave as if the user is directly typing the data into the Google Sheets GUI (For example, formulas will be accepted). *'RAW'* allows what is entered to be placed into the cells without further interpretation by Google Sheets
    - majorDimension_ :: Must be *'ROWS'* or *'COLUMNS'*. This indicates which direction to iterate as you place the data. Choose *'ROWS'* to enter data vertically
- Post-Con:
    - Places the following formula into the range *crangeL_ rrangeL_ : crangeR_ rrangeU_*, *=(num_Col*/D*)*100*. Remark, column D should be set up to house the total number of operations for all time groupings. See **II::3.D.i** for more details.
- Example Usage:

```
GoogleSheetsAPI.GS_Put_Percent(1, upper_8, 'J', 'J', 'E',
'USER_ENTERED', 'ROWS');
```

1.6 Helpers

- Python 2.7.12 $*handle(response)*
    - Pre-Con:
        - response :: The JSON response of a Confluence request
    - Post-Con:
        - Returns the page ID of the newly created page if the creation was successful. If not, it returns the JSON response in its entirety.
    - Remarks:
        - The validity of the response is determined by string parsing meaning that if the server changes the format of the response, this function will need to be modified accordingly.
    - Example Usage:

```
#Create page and get ID
id_ = ConfluenceAPI.post(name);
id_ = Helpers.handle(id_);

#Assert page has been created
if not Helpers.valid(id_):
 print(id_);
    exit(2);
```

- Python 2.7.12 $*valid(id)*
    - Pre-Con:
        - id :: The ID of a Confluence page
    - Post-Con:
        - Returns true if the ID is of the right format for a Confluence Page ID. **Remark:** It does not neccesarily gaurentee that the ID points to a page that exists.
    - Remarks:
        - The validity of the response is determined by string parsing meaning that if the server changes the format of the response, this function will need to be modified accordingly.
    - Example Usage:

```
id_ = Helpers.handle(ConfluenceAPI.post(name);
if not Helpers.valid(id_):
    #code
```

- Python 2.7.12 $*constructPayload(value)*
    - Pre-Con:
        - value :: String to be saved into a text file containing the payload value for a Confluence PUT command.
    - Post-Con:
        - Writes *value* into './update.txt'
    - Remarks:
        - Mixing HTML and Non-HTML entries in one PUT request will result in a PUT Status Code 500 (Invalid value).
    - Example Usage:

```
Helpers.constructPayload("<h1
style=\\\"text-align:center;text-decoration:underline;\\\">Banner
4 Week KPI Review</h1>");
```

- Python 2.7.12 $*imageTag(id, file)*
    - Pre-Con:
        - id :: The ID of a page you wish to attach an image to
        - file :: The file name and type of your image file
            - **Remark:** File must be of type *.png*
            - **Remark:** The image must already be attached to the Confluence page whose body you wish to add the image to. See **1.2 - ConfluenceAPI ::`Python2.7.12 $dpost(id, path)`**
    - Post-Con:
        - Returns a string that contains the correct formatting for placing an attached image into the body of a Confluence page
    - Example Usage:

```
ConfluenceAPI.dpost(id_, "banner.png");
Helpers.constructPayload(Helpers.imageTag(id_, "banner.png"));
```

- Python 2.7.12 $*getDates()*
    - Pre-Con:
        - N/a
    - Post-Con:
        - Returns a list of dates that are incremented by seven days for each index in **range(0,4)**, most recent in **list[0]**.
    - Example usage:

```
weeks = Helpers.getDates();

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
weeks[0] + "</td>");
```

- Python 2.7.12 $*unwrap(source_, target_)*
    - Pre-Con:
        - source_ :: JSON response
        - target_ :: The key of the value you wish to get from the response
    - Post-Con:
        - Returns the value associated with *target_* if it exists. Otherwise, an exception is thrown.
    - Example usage:

```
Helpers.unwrap(NewRelicAPI.Average_Operation_Time(), 'average');
```

- Python 2.7.12 $*buildStack_3(jsonPack, lim)*
    - Pre-Con:

        - jsonPack :: JSON response returned by `NewRelicAPI.Slow_URLs()`
        - lim :: The number of items to return
            - **Remark:** This number must be less than or equal to the number of items returned by the dependent query. See **1.3 - NewRelicAPI :: `Python2.7.12 $Slow_URLs()`**
    - Post-Con:
        - Returns a list that contains each JSON item relevant to the ScoreCard in its own indice in FILO order.
        - **Remark:**_*3* specifies the number of columns the table uses besides the name column
    - Example usage:

```
slow = NewRelicAPI.buildStack_3(NewRelicAPI.Slow_URLs(), 10);

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
slow.pop() + "</td>");#Remove and return the last element in the
list
```

- Python 2.7.12 $*buildStack_8(jsonPack, lim)*
    - Pre-Con:
        - jsonPack :: JSON response returned by **`NewRelicAPI.Top_Slow()`**
        - lim :: The number of items to return
            - **Remark:** This number must be less than or equal to the number of items returned by the dependent query. See **1.3 - NewRelicAPI :: `Python3.6.1 $Top_Slow()`**
    - Post-Con:
        - Returns a list that contains each JSON item relevant to the ScoreCard in its own indice in FILO order
    - Example usage:

```
dataContainer = Helpers.buildStack_8(NewRelicAPI.Top_Slow(), 25);
#Flashline Bucketlist
dataContainer.pop(0); #Remove sentinal

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataContainer.pop(0); + "</td>");#Remove and return the first
element in the list
```

- Python 2.7.12 $*buildIndefiniteStack(jsonPack)*
    - Pre-Con:
        - jsonPack :: JSON Response
    - Post-Con:
        - Returns a list that contains each JSON item relevant to the ScoreCard in its own indice in FILO order.
            - **Remark:** This return has a variable rate of return items meaning it can take a query that returns conditional data
    - Example usage:

```
dataContainer =
Helpers.buildIndefiniteStack(NewRelicAPI.Transaction_Perf());
dataContainer.pop(0);
```

- Python 2.7.12 $*handle_SQL(source_)*
    - Pre-Con:
        - source_ :: SQL response returned by a SQL query made with the **MySQLdb** Python library containing only a single

result.
- Post-Con:
    - Removes the wrappers from the SQL response and returns the data as a string
- Example Usage:

```
import MySQLdb
db = MySQLdb.connect(

host="emonscorecards.cqn2vtdstij1.us-east-2.rds.amazonaws.com",
                     user="scorecarduser",

                     passwd="cheeseburger",

                     db="scorecards"
);

cur = db.cursor();

cur.execute(
 "SELECT dataValue FROM Banner_FrontPage WHERE dataDate LIKE '" +
  (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y-%m-%d') + "' AND
  dataDescription LIKE 'Number of Critical Events';");


dataSet[0] = Helpers.handle_SQL(str(cur.fetchall())); #Array of
SQL results (one per index)

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[0] + "</td>");
```

- Python 2.7.12 $*stackHelper(source_, target_, iterator)*
    - Pre-Con:
        - source_ :: A NewRelic API response (I.E. **NewRelicAPI::Slow_URLs()**)
        - target_ :: Name of the item you wish to fetch from the API response
        - iterator :: The number of times *target_* appears (minus one) before the value you wish to retrieve
    - Post-Con:
        - Returns the value correlating with *target_* after *iterator* occurrences of *target_* in *source_*
    - Example Usage:

```
for iteration in range(0, int(lim)): #Code snipet from
buildStack_3 - lim is upper bound of iteration
 stack.append(stackHelper(jsonPack, 'name', iteration))
    stack.append(stackHelper(jsonPack, 'average', iteration))
    stack.append(stackHelper(jsonPack, 'count', iteration))
```

- Python 2.7.12 $*stackHelper2(source_, target_, iterator)*
    - Pre-Con:
        - source_ :: A NewRelic API response (I.E. **NewRelicAPI::Slow_URLs()**)
        - target_ :: Name of the item you wish to fetch from the API response
        - iterator :: The number of times *target_* appears (minus one) before the value you wish to retrieve

        - Post-Con:
            - Returns the value correlating with *target_* after *iterator* occurrences of *target_* in *source_*
        - Remarks:
            - For use with **NewRelicAPI::Top_Slow()**
- Python 2.7.12 $*resolve_Statistical_Inequality(new_, old_, domination_Implication_)*
    - Pre-Con:
        - new_ :: LHS of inequality *float(new_) > float(old_)*
        - old_ :: RHS of inequality float(new_) > float(old_)
        - dominitation_Implication_ :: The answer to the question, "Is it 'Good', 'Bad', or 'Indifferent', if *new_ > old_* TRUE?" This parameter must be one of the following strings (case sensitive): 'Good', 'Bad', or 'Indifferent'.
    - Post-Con:
        - Returns the proper character to indicate the direction and implication for the change in the data from *old_* to *new_ (arrow)* wrapped in a HTML column tag.
    - Example Usage:

```
Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");
Helpers.constructPayload("<tr>");
Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[1], dataSet[2], 'Indifferent'));
```

- Python 2.7.12 $*conf_Unwrapper(source_, target_)*
    - Pre-Con:
        - source_ :: The JSON response from **ConfluenceAPI::get(title)**
        - target_ :: The element to pull from the response
    - Post-Con:
        - Returns from three characters after *target_* to the next instance of the character *'<'*
    - Example Usage:

```
NumIss1 = Helpers.conf_Unwrapper(ConfluenceAPI.get('Flashline
Scorecard - ' + (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%m/%d/%Y')),
'NumIss1').encode("utf-8");

cur.execute("INSERT INTO
Flashline_FrontPage(ID,dataDescription,dataValue,dataDate)
VALUES(0,'Number of Issues Identified','" + NumIss1 + "'," +
(datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y%m%d') + ");");
```

II. Interactivity & Design Methodology

Interactivity

David Veits will discuss AWS Step Functions, Lambda, everyhing that starts the process up to the launch of the EC2 instance. Note that Jacob Adkins will write the documentation for the rest of the process as is. Meaning that I will assume, for the time being, the screenshots will still be handled in seperate buckets by a script that runs on EC2. Below is an image that shows the entire process flow. I believe this section could be composed as follows: an overall summary of the process could replace this paragraph, followed by the image, followed by more detailed portions.

Design Methodology

In this section I discuss coding, design choices, and the methodology for the various components of the scorecard. The goal of this section is to give some level of insight to the user that they may find helpful when attempting to automate new components of scorecards in the future.

1. S3
    a. Uploading an image

```
aws s3 cp "PATH_TO_YOUR_IMAGE" s3://$BUCKET_NAME --region
us-east-2
```

    b. Download an image

```
for obj in s3.Bucket(BUCKET_NAME).objects.all(): #Download all
files in S3 Bucket, BUCKET_NAME, to screenshot_path as filename
 filename = obj.key.rsplit('/')[-1];
    s3.Bucket(BUCKET_NAME).download_file(filename,
screenshot_path + filename)
```

2. Confluence
    a. Creating a new page

```
print("Creating Confluence page " + name + "...");
id_ = ConfluenceAPI.post("NAME OF CONFLUENCE PAGE")
id_ = Helpers.handle(id_)
print("Done. Page ID: " + id_);

#Assert page has been created
if not Helpers.valid(id_):
 print(id_)
    exit(2)
```

    b. Upload an image

```
print("Uploading image attachments...");
for currentFile in os.listdir(screenshot_path):
    ConfluenceAPI.dpost(id_, screenshot_path + currentFile);
 upload_count += 1;
print("Done.");
```

    c. Attach uploaded image to Confluence page (You must attach an image to the page before you can attach it to the page)

```
Helpers.constructPayload(Helpers.imageTag(CONFLUENCE_PAGE_ID,
IMAGE_NAME.png));
```

Remark: File type must be .png

    d. Add header text

```
Helpers.constructPayload("<h1
style=\\\"text-align:center;text-decoration:underline;\\\">Banner
4 Week KPI Review</h1>");
```

Remark: Most basic HTML is supported by Confluence. You make smaller headers by using *<h2>* or *<h3>* tags instead of *<h1>*.

You can also change the alignment by changing the *text-align* element to "right" or "left". Most CSS is supported as well. This page is a good reference for HTML beginners.

e. Create a table

```
Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">")
Helpers.constructPayload("<tr>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">COLUMN
CONTENT</td>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">COLUMN
CONTENT 2</td>")
Helpers.constructPayload("</tr>")
Helpers.constructPayload("</table>")
```

This will give you something similar to, but centered on the page:

COLUMN CONTENT	COLUMN CONTENT 2

f. Uploading to Confluence

- **Remark:** The *constructPayload* ultimately appends the passed string to a file in the current directory called *update.txt*. We then pass the file to the Confluence PUT call, and upload it's contents. Since we are appending, it is wise to delete *u pdate.txt* after a PUT call. Also note that a PUT call will replace the contents of the page, not append.

```
ConfluenceAPI.put(id_, "/home/ec2-user/Banner
Scorecard/update.txt" , "NAME_OF_CONFLUENCE_DOCUMENT"),
version)
os.remove("/home/ec2-user/Banner Scorecard/update.txt");
```

3. Scorecard Components

a. Front Page Data Table

- The data for this portion of the scorecard is to be kept in AWS RDS. In order to access it, you will need to query the database. After setting up the MySQL object *cur*, you can perform queries like this

```
cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=7)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
```

This fetches *Availability*. I've Discussed in detail how you can modify these queries in **IV::24**. **Remark:** For the APPLICATION_FrontPage tables in your RDS instance, you want to ensure that a call like this returns only one result due to the way it is handled:

```
dataSet[0] = Helpers.handle_SQL(str(cur.fetchall()));
```

I then use this list item and place it directly into a column in the Front Page Table:

```
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[0] + "%</td>")
```

The arrows are placed using the *resolve_Statisitcal_Inequality* function. See 1.6. Putting all of this together, we have this code to get us the header, and first row of the Front Page Data Table for Banner:

```
weeks = Helpers.getDates()
Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">")

Helpers.constructPayload("<tr>")
Helpers.constructPayload("<td>Week starting on</td>")
Helpers.constructPayload("<td></td>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
weeks[3] + "</td>") #automate date
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\"></td
>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
weeks[2] + "</td>") #automate date
Helpers.constructPayload("<td></td>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
weeks[1] + "</td>") #automate date
Helpers.constructPayload("<td></td>")
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
weeks[0] + "</td>") #automate date
Helpers.constructPayload("</tr>")

#App Performance Queries
dataSet = ['-1','-1','-1','-1', '-1']

cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=7)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[0] = Helpers.handle_SQL(str(cur.fetchall()))

cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=10)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[1] = Helpers.handle_SQL(str(cur.fetchall()));

cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=17)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[2] = Helpers.handle_SQL(str(cur.fetchall()));

cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=24)).strftime('%Y-%m-%d') + "' AND
```

```
dataDescription LIKE 'Availability';");
dataSet[3] = Helpers.handle_SQL(str(cur.fetchall()));

cur.execute("SELECT dataValue FROM Banner_FrontPage WHERE
dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=31)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[4] = Helpers.handle_SQL(str(cur.fetchall()));

Helpers.constructPayload("<tr>")
Helpers.constructPayload("<td>Availability<p
style=\\\"font-size:9px\\\">&#40;All Attempts &#45;
Failures&#41; &#47; All Attempts</p></td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequa
lity(dataSet[0], dataSet[1], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[0] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequa
lity(dataSet[1], dataSet[2], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[1] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequa
lity(dataSet[2], dataSet[3], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[2] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequa
lity(dataSet[3], dataSet[4], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[3] + "%</td>")
Helpers.constructPayload("</tr>")
```

b. Inserting a Screenshot

```
print("Adding Benchmark App Health to Banner Scorecard...");
Helpers.constructPayload("<h3
style=\\\"text-align:center;font-weight:bold;\\\">TITLE OF GRAPH
HERE</h3>");
Helpers.constructPayload(Helpers.imageTag(CONFLUENCE_PAGE_ID,
"YOUR_FILE_PATH_HERE.png"));
print("Done.");
```

c. Top Ten Slow URLs Table

- Since we only need to access this week's data for this component of the page, we can use an API call instead of RDS. the first thing we need to do is make and parse the API calls for our ten URLs:

```
dataContainer = [];
for i in range(0, 10):
dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.p
arseDataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_A
PPLICATION()), i)));
```

Now we can create and populate the table:

```
Helpers.constructPayload("<h3
style=\\\"text-align:center;text-decoration:underline;\\\">
Top 10 Slow Performing URLs</h3>");

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");

Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;font-weight
:bold;\\\">Operation</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">Count</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">Time</td>");
Helpers.constructPayload("</tr>");

for y in range(0,10):
 Helpers.constructPayload("<tr>");
    Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\\\">" +
str(dataContainer[y].pop(0)) + "</td>");
    Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">" + str(float(dataContainer[y].pop(0))) +
"</td>");
    Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">" + str(round(float(dataContainer[y].pop(0)),
2)) + "</td>");
    Helpers.constructPayload("</tr>");

Helpers.constructPayload("</table>");
```

d. Bucketlist
- The bucketlist has two steps, creating the bucketlist, and attaching the bucketlist.
    i. Creating the Bucketlist
    - The idea behind the bucketlist is looking at URL response time in time deltas of 0-2 seconds, 2-3 seconds, 3-5 seconds, 5-8 seconds, and 8+ seconds. What I do to get this data is the following:
        - Get the data out of DCRum for URLs 0-2s

```
query =
DCRumHelpers.execute_BucketList_Query(DCRumA
PI.GET_Bucket_02_Ban());
```

        - Put the URLs from the query results into a Google Sheet

```
url_02 = GoogleSheetsAPI.GS_Put('1',
str(len(query)), 'A', 'A', 'USER_ENTERED',
query, 'ROWS', 0);
```

- Put the operations from the query results into the next column over

```
GoogleSheetsAPI.GS_Put('1', str(len(query)),
'B', 'B', 'USER_ENTERED', query, 'ROWS', 1);
```

- Get a unique list of the URLs from the list of URLs we placed in column A

```
GoogleSheetsAPI.GS_Put_Unique('1', '1', 'C',
'C', 'USER_ENTERED', '=UNIQUE(A:A)', 'ROWS');
```

- Sum the OpTimes so that each unique list has the OpTime for all occurences of the URL in our query result

```
upper_02 = len(GoogleSheetsAPI.GS_Get('', '',
'C', 'C', 'USER_ENTERED', 'ROWS')); # get
number of unique URLs to use as a bound
GoogleSheetsAPI.GS_Put_Sumif(1, upper_02,
'E', 'E', '1', url_02, 'USER_ENTERED',
'ROWS');
```

- Repeat this process for all remaining time deltas using the next available row in column A. See how this is achieved below:

```
query =
DCRumHelpers.execute_BucketList_Query(DCRumA
PI.GET_Bucket_23_Ban());
url_23 = GoogleSheetsAPI.GS_Put(str(url_02 +
1), str(url_02 + 1 + len(query)), 'A', 'A',
'USER_ENTERED', query, 'ROWS', 0);
```

Remark:*url_02* is set to be the number of rows used in A by our last Google Sheets PUT request.

- See *Scorecard-Automation-Project/Deliverable/Banner/Bucketlist/Build_Bucket_List.py* to see how this process is carried out as the timedeltas progress. Basically, you just need to sum the every list up to that point. We put the sums of the OpTimes in the columns after the unique list of URLs. **Remark:** The unique list is in column C so we put the sum of the URLs' OpTimes for 0-2 seconds in column E, the sum of the URLs' OpTimes for 2-3 seconds in column F, and so on. Column D will be used for totals.
- Once we have all of the sums, we Sum all of the rows of operations for each URL and place the totals in column D.

```
GoogleSheetsAPI.GS_Put_Sum(1, upper_8, 'D',
'D', 'USER_ENTERED', 'ROWS');
```

- We use the time delta over the total to then give us the percentage of Operations for that time delta and place them in the next columns over. That is, The 0-2s% Ops go in column J, 2-3s% Ops go in Column K, etc.

```
GoogleSheetsAPI.GS_Put_Percent(1, upper_8,
'J', 'J', 'E', 'USER_ENTERED', 'ROWS');
```

- We also append the total number of operations that took over 8 seconds to the last column:

```
GoogleSheetsAPI.GS_Put_Sumif(1, upper_8, 'O',
'O', 1 + url_02 + url_23 + url_35, url_02 +
url_23 + url_35 + url_58 + url_8,
'USER_ENTERED', 'ROWS');
```

- To alleviate some degree of confusion, I put a table below explaining what data each column houses

A	B	C	D	E	F	G	H	I
Raw URLs from DCRum	Raw Operations from DCRum (Corresponding to the URLs in Row A)	Unique List of URLs Constructed from A	SUM(E,F,G,H,I)	Number of Operations 0-2s for Corresponding URL in C	Number of Operations 2-3s for Corresponding URL in C	Number of Operations 3-5s for Corresponding URL in C	Number of Operations 5-8s for Corresponding URL in C	Numbe Operat >8s f Correspo URL i

- Then we run a script that replaces all formula data with data values and sorts the data

- **Remark:** We need to create the Google Sheet Document and expand the number of rows we are allowed to use before we start this process. I accomplish this, and calling the last script I mentioned using *Scorecard-Automation-Project/Deliverable/APPLICATION/Bucketlist/driver.sh*

ii. Adding the Bucketlist to the Scorecard

- The Bucketlist examines 25 URLs. Specifically, the bottom 25 rows of the spreadsheet we just created. We need to get the unique URLs from set of data, the percentages, the total operations, and the number of operations that took more than eight seconds. In otherwords, we need the bottom 25 rows of columns C, D, and J through O.

```
unique_Lim = len(GoogleSheetsAPI.GS_Get('', '',
'C', 'C', 'USER_ENTERED', 'ROWS')); # get number
of unique URLs
bucketURLs = GoogleSheetsAPI.GS_Get(str(unique_Lim
- 24), str(unique_Lim), 'C', 'C', 'USER_ENTERED',
'ROWS');# select the bottom 25
bucketTotals =
GoogleSheetsAPI.GS_Get(str(unique_Lim - 24),
str(unique_Lim), 'D', 'D', 'USER_ENTERED',
'ROWS');
bucketValues =
GoogleSheetsAPI.GS_Get(str(unique_Lim - 24),
str(unique_Lim), 'J', 'O', 'USER_ENTERED',
'ROWS');
```

- Now we just need to create and populate the table:

```
Helpers.constructPayload("<h3
style=\\\"text-align:center;text-decoration:under
line;\\\">Top 25 Bucket List of URLs >8s with More
```

```
Than 10 Operations</h3>");

Helpers.constructPayload("<table
align=\\\"center\\\" width=\\\"967px\\\"
border=\\\"0\\\">");

Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;f
ont-weight:bold;\\\">Operation</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">Total</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">% of Ops 0-2s</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">% of Ops 2-3s</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">% of Ops 3-5s</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">% of Ops 5-8s</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">% of Ops &gt; 8s</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;font-weight:bold;\\\">Total &gt; 8s</td>");
Helpers.constructPayload("</tr>");

for x in range(0,25):
 Helpers.constructPayload("<tr>");
 Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\
\\">" + str(bucketURLs[x])[3:len(bucketURLs[x])-3]
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" +
str(bucketTotals[x])[3:len(bucketTotals[x])-3] +
"</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][0]), 2))
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][1]), 2))
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][2]), 2))
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][3]), 2))
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][4]), 2))
+ "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle
;\\\">" + str(round(float(bucketValues[x][5]), 2))
+ "</td>");
```

```
    Helpers.constructPayload("</tr>");

  Helpers.constructPayload("</table>");
```

e. Front Page Graph

- The front page graph shows correlation of Availability (total) %, Unique Users, and Application Performance, over time. We store one data entry per hour in DCRum over the course of a week, and insert it into RDS as part of the *data_collector.py* script:

```
query =
DCRumHelpers.handle(DCRumAPI.GET_Data_to_Export_Ban())
dataContainer =
DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query, 0));
i = 0;

while dataContainer[3] != '': # when the list is empty, stop
adding data to RDS
        cur.execute("INSERT INTO
Banner_FrontPageGraph(ID,begT,Avb,appPerf,CliCnt) VALUES( 0"
+ ",'" + dataContainer[0] + "'," + dataContainer[1] + "," +
dataContainer[2] + "," + dataContainer[3] + ");");
        i += 1;
        dataContainer =
DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query, i));
```

We then get the data out of RDS for the last four weeks to generate the graph in *Scorecard-Automation-Project/Deliverable/APPLICATION/Graphs/FrontPage.py*

```
aval_total = [];
cur.execute("SELECT avb FROM Banner_FrontPageGraph WHERE
begT > '" + (datetime.datetime.today() -
datetime.timedelta(hours=672)).strftime("%Y-%m-%d %H:%M") +
"' AND begT < '" + (datetime.datetime.today() -
datetime.timedelta(days=0)).strftime("%Y-%m-%d %H:%M") +"'
ORDER BY begT;");

for row in cur.fetchall():
    aval_total.append(row[0]);

uniq_user = [];
cur.execute("SELECT CliCnt FROM Banner_FrontPageGraph WHERE
begT > '" + (datetime.datetime.today() -
datetime.timedelta(hours=672)).strftime("%Y-%m-%d %H:%M") +
"' AND begT < '" + (datetime.datetime.today() -
datetime.timedelta(days=0)).strftime("%Y-%m-%d %H:%M") +"'
ORDER BY begT;");

for row in cur.fetchall():
    uniq_user.append(row[0]);

app_perf = [];
cur.execute("SELECT appPerf FROM Banner_FrontPageGraph WHERE
begT > '" + (datetime.datetime.today() -
datetime.timedelta(hours=672)).strftime("%Y-%m-%d %H:%M") +
"' AND begT < '" + (datetime.datetime.today() -
datetime.timedelta(days=0)).strftime("%Y-%m-%d %H:%M") +"'
ORDER BY begT;");

for row in cur.fetchall():
    app_perf.append(row[0]);
```

- We reverse the list data, set the x-axis to reflect the length of the list, and graph. Consult Python *matplotlib* module documentation and other internet resources for more information on how to edit these processes. **Remark:** Graphing three plots on two seperately labeled axes is too complex to explain in sufficient detail in the scope of this documentation.

4. Google Sheets
   a. Create new Google Sheets Document in Google Drive (After setting up credentials and authorization components)

```
file_metadata = {
 'name' : 'Banner : Bucket List - ' +
datetime.datetime.today().strftime('%m/%d/%Y'),
 'mimeType' : 'application/vnd.google-apps.spreadsheet'
}

file =
service.files().create(body=file_metadata,fields='id').execute()
```

5. General Design Patterns
   a. Removing an Element from the Scorecard

- If you remove an element from the Scorecard, be sure that you remove all of the HTML tags surrounding it as well. For example, to remove the Top Ten Slow URLs Table, we would have to remove all of the following lines of code:

```
dataContainer = [];
for i in range(0, 10):
 dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.
parseDataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_
Ban()), i)));

Helpers.constructPayload("<h3
style=\\\"text-align:center;text-decoration:underline;\\\">
Top 10 Slow Performing URLs</h3>");

Helpers.constructPayload("<table align=\\\"center\\\"
width=\\\"967px\\\" border=\\\"0\\\">");

Helpers.constructPayload("<tr>");
Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;font-weight
:bold;\\\">Operation</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">Count</td>");
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">Time</td>");
Helpers.constructPayload("</tr>");

for y in range(0,10):
 Helpers.constructPayload("<tr>");
 Helpers.constructPayload("<td
style=\\\"text-align:left;vertical-align:middle;\\\">" +
str(dataContainer[y].pop(0)) + "</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">" + str(float(dataContainer[y].pop(0))) +
"</td>");
 Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;font-weig
ht:bold;\\\">" + str(round(float(dataContainer[y].pop(0)),
2)) + "</td>");
 Helpers.constructPayload("</tr>");

Helpers.constructPayload("</table>");
```

If we forget to remove a closing tag like the last line in the above example, we will have invalid HTML, and we will get Status Code 400 when we try to push the page update to confluence.

III. Components & Setup

In this section I discuss process flow between the components, and how each component should be set up.

A. EC2 - Component Setup

- The EC2 Component of the Scorecard Automation Project is responsible for housing all of the active components of the process and executes all of the queries and API calls. It then compiles the scorecard and pushes it to the Confluence Server. Each application that is being built will have it's own EC2 instance. <u>**Hint: If you open files to edit them and they are missing line breaks, try using Notepad++ if you're on a Windows environment or use a remote linux enviornment and run</u> *:ret :wq* <u>using vim on the file.</u>
- **APPLICATION** denotes the need for you to apply the change to several files. The bolded part of the path reflects the part that will change. Unless other wise noted, you should apply it to all directories below the bold instance. For example, if I say *ConfluenceAPI.py* in *Scorecard-Automation-Project/Deliverable/**APPLICATION**/lib* I mean the files:
  - *Scorecard-Automation-Project/Deliverable/**Flashline**/lib/ConfluenceAPI.py*
  - *Scorecard-Automation-Project/Deliverable/**Banner**/lib/ConfluenceAPI.py*
  - *Scorecard-Automation-Project/Deliverable/**Learn**/lib/ConfluenceAPI.py*
  - *Scorecard-Automation-Project/Deliverable/**Solutions**/lib/ConfluenceAPI.py*

1. Use a text editor to enter the proper credentials for the Confluence Server in the file, *ConfluenceAPI.py* (lines 9-12) found in *Scorecard-Automation-Project/Deliverable/**APPLICATION**/lib* (Also ensure that the space you designate, exists):

```
#Example credentials entered
9:  url = 'http://18.220.102.127:8090'
10: user = 'admin'
11: pass_ = 'jakesucks'
12: SPACE_ID = 'spac'
```

2. Use a text editor to enter the proper credentials for the NewRelic Server in the file, *NewRelicAPI.py* (line 9) found in Scorecard-Automation-Project/Deliverable/**APPLICATION**/lib. Follow this guide to create a NewRelic query key.

```
9: queryKey = 'r2EtoM8pXEwu0BlaYrlfLG2Ao1kO0kla'
```

3. In order to access certain parts of the system you need API credentials. These are stored in files, and accessed by the system when they are needed. You need to create these files. The easiest way to do this will be to log into a linux terminal, and enter the command:

```
aws --config
```

   Follow the prompts and then navigate to *$HOME/.aws/* here you will find a *config* file. Now enter the command:

```
touch credentials
```

   This will create a file named credentials. Use vim or any other text editor you prefer to open *credentials*, and add the following contents:

```
[default]
aws_access_key_id = YOUR_ACCESS_KEY
aws_secret_access_key = YOUR_SECRET_ACCESS_KEY
```

   A guide on how to generate an AWS keyset can be found here. Download both the *credentials* and *config* files and replace the ones currently located in *Scorecard-Automation-Project/Deliverable/.aws*

4. Use a text editor to enter the proper credentials for the RDS Server in the file*, dataCollector.py (lines 11-14)* located at *Scorecard-Automation-Project/Deliverable/**APPLICATION***. An example is given below:

```
db = MySQLdb.connect(
 host="emonscorecards.cqn2vtdstij1.us-east-2.rds.amazonaws.com",
# your host
    user="scorecarduser",
# your username
    passwd="cheeseburger",
# your password
    db="scorecards"
# name of the database
);
```

5. Use a text editor to enter the proper credentials for the RDS Server and the S3 Bucket in the file *card_builder.py* (lines 29-39) located at *Scorecard-Automation-Project/Deliverable/**APPLICATION*** An example is given below:

```
BUCKET_NAME = "jakesource";

conn = boto.ec2.connect_to_region(
 "us-east-1",
    aws_access_key_id='AKIAJR45JK27X3F2OEGA',

aws_secret_access_key='kfv+j5ZoFy+hneHncIcGtEc8QE+nofL6eHfCcaVI'
);

db =
MySQLdb.connect(host="emonscorecards.cqn2vtdstij1.us-east-2.rds.
amazonaws.com",
 user="scorecarduser",
    passwd="cheeseburger",
    db="scorecards");
```

   Ensure that the AWS S3 credentials belong to the owner of the declared bucket, *BUCKET_NAME* which is also the account used to generate the *config* file in III - 3.1A (3)** and the account whose information resides in the *credentials* file at *Scorecard-Automation-Project/Deliverable/.aws*

6. Use a text editor to set the S3 Bucketname for the screenshots to be uploaded into in the file named flashline_screenshot_bootstrap.sh (line 4) located at *Scorecard-Automation-Project/Deliverable/**APPLICATION***. An example is given below:

```
BUCKET_NAME=jakesource
```

   Remark: Ensure there is no spaces in this line of code.

7. **Remark:** This project depends on this git repository. Specifically, the branch, *Automation_Screenshot Scripts*. Should the current NewRelic credentials used in those scripts become invalid, a commit to that branch of the repository will have to be made to update the log in credentials used for screenshots.

8. Branch all changes to the Git Server

B. RDS Component - Setup

- The RDS Component of the Scorecard Automation Project is responsible for housing all of the archived data used during the process for the purpose of creating graphs and/or tables. Each application requires two tables in the RDS database:

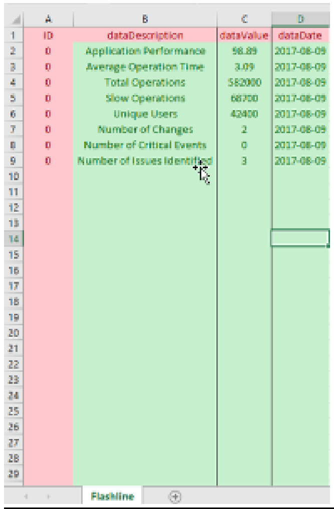
    - $AppName$_FrontPage
    - $AppName$_FrontPageGraph

  with the exception of Flashe line wich requires only the former.

- To begin, create an RDS instance with a database named, *scorecards*

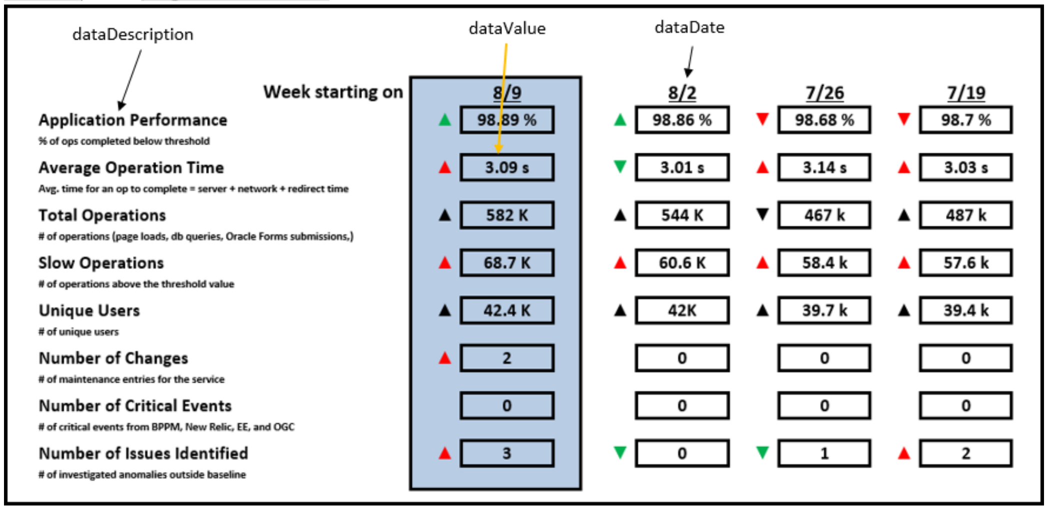
1. Use a MySQL Management Studio (Like MySQL Workbench) to open, and execute the necessary table construction queries located in the file *Scorecard-Automation-Project/Front Page Init.sql*

2. In order to initialize the automation process (See **V**), we'll need to have the past five weeks' data sets readily available. To do this we're going to have to enter them manually. This process can be expedited via Excel. See below.
    a. Open up the most recent scorecard available, and use the front page table data to fill in the columns in the *dataLoader.xlsx* file provided in *Scorecard-Automation-Project/dataLoader.xlsx* (See below)



	A	B	C	D
1	ID	dataDescription	dataValue	dataDate
2	0	Application Performance	98.89	2017-08-09
3	0	Average Operation Time	3.09	2017-08-09
4	0	Total Operations	582000	2017-08-09
5	0	Slow Operations	68700	2017-08-09
6	0	Unique Users	42400	2017-08-09
7	0	Number of Changes	2	2017-08-09
8	0	Number of Critical Events	0	2017-08-09
9	0	Number of Issues Identified	3	2017-08-09

Flashline


dataDescription
dataValue
dataDate

Week starting on	8/9	8/2	7/26	7/19
Application Performance % of ops completed below threshold	▲ 98.89 %	▲ 98.86 %	▼ 98.68 %	▼ 98.7 %
Average Operation Time Avg. time for an op to complete = server + network + redirect time	▲ 3.09 s	▼ 3.01 s	▲ 3.14 s	▲ 3.03 s
Total Operations # of operations (page loads, db queries, Oracle Forms submissions,)	▲ 582 K	▲ 544 K	▼ 467 k	▲ 487 k
Slow Operations # of operations above the threshold value	▲ 68.7 K	▲ 60.6 K	▲ 58.4 k	▲ 57.6 k
Unique Users # of unique users	▲ 42.4 K	▲ 42K	▲ 39.7 k	▲ 39.4 k
Number of Changes # of maintenance entries for the service	▲ 2	0	0	0
Number of Critical Events # of critical events from BPPM, New Relic, EE, and OGC	0	0	0	0
Number of Issues Identified # of investigated anomalies outside baseline	▲ 3	▼ 0	▼ 1	▲ 2

Add the number, 0, to the ID column for each row that is non-empty. Do not edit the column headers. Ensure that you are in the right sheet for the selected application. Ensure that your date format matches that in the example provided. Add only numerical characters to the *dataValue* column. 'K', 'k', and 'M' should be interprated to their byte value. Omit units of measurement like seconds. Insert all of the data present on this scorecard, then go back and get one last set of data. Remember, we need the last **5** weeks to initialize the automation process. This only gives us four. In this case, we would still need to insert the week of 7/12.

    b. Once you have entered all of the required information (Note Figure_2a.png is not complete as it shows only one week) export the excel file as $ANYFILENAME$.csv and proceed here.

    c. Click on the Load Form button and browse to *Scorecard-Automation-Project/SQL Insert JSONs/**APPLICATION**_FrontPage.json*, and click "Okay".

Now select "Choose a CSV/Excel File" and browse to the *.csv* you just saved. **Remark:** If this is for a new app and there is APPLICATION_FrontPage.json for the application, just load any of them and change the Schema.Table Name to match what you've created in RDS in step **III::B.1**:

   d. Scroll to the bottom of the page and click "CSV To SQL Insert", and copy and paste the results into your MySQL Management Studio. Execute all of the statements.

3. We'll also need the last four weeks of data for the Front Page Graph. This week's data will be added via script; however, you'll need to add the earlier weeks' data yourself.
   a. Open the most recent version of the *App Graph Data.xlsx* file and copy all of the data in the range *4 weeks ago from this Wednesday 0:00* to *3 weeks ago from this Tuesday 23:00*
   b. Paste your selection into a new Excel document starting at cell B1. Add a 0 to column A for every row containing data in Column B. Select all data in column B, and change the format to Custom *m/d/yyyy h:mm*, then click in the *Type:* bar and change it to the text, *yyyy-mm-dd hh:mm*. Export the document as a *.csv*. Head to the SQL Insert Converter again, and open the app's *Scorecard-Automation-Project/SQL Insert JSONs/APPLICATION_FrontPageGraph.json*, upload your .csv, convert, and execute.

C. Google Sheets Component - Setup

- I've created a Google Account for the sole purpose of housing Scorecard Bucketlists. It is located here. Contact WALTER BAINEY, David Veits, or Jacob Adkins for access.
- The Google Sheets component of the system is used to create the Bucketlist for Learn, Banner, and Solutions. In the past we used MS Excel; however, Sheets was used due to its web service nature (Web Service API Calls). You need to do two things to enable API calls from your EC2 instance:

1) Enable API Calls

- Follow **only Step 1** in this guide. **Remark:** Steps *1g* and 1*h* have been completed for you and *client_secret.json* currently resides in the package wherever it need be (That is *Scorecard-Automation-Project/Deliverable/**APPLICATION**/Bucketlist* and *Scorecard-Automation-Project/Deliverable/**APPLICATION**/Bucketlist/Drive*) if you're using the account mentioned above. In otherwords, if you are using the account mentioned above, skip steps 1g and 1h.
- Go to your API library and search for Google Sheets
- Click what should be the only result, and then click "ENABLE" at the top of the page. If done correctly, it will now show "DISABLE" as a clickable option.

2) Create User Credentials

- **Remark:** You can skip the entirety of Step 2 if you are using the account mentioned above
- Sign into a junk EC2 instance (not one you intend to use for scorecard creation), download the Scorecard-Automation-Project, copy the *Generate_Sheets_Credentials.py* to your $HOME directory, and copy your *client_secret.json* to your home directory as well.

```
git clone
https://jadkin31_stu:Jacksonv12@code.kent.edu/jadkin31_stu/
Scorecard-Automation-Project.git
mv
Scorecard-Automation-Project/Generate_Sheets_Credentials.py
.
cp
Scorecard-Automation-Project/Deliverable/Banner/Bucketlist/
client_secret.json .
```

- Run the following command from your home directory

```
python Generate_Sheets_Credentials.py
--noauth_local_webserver
```

use the link on your local machine to get the authorization code. This will generate *sheets.$HOME/.credentials/googleapis.com-python-quickstart.json*

- Repeat this process for the file Generate_Drive_Credentials.py:

```
mv
Scorecard-Automation-Project/Generate_Drive_Credentials.py .
python Generate_Drive_Credentials.py
--noauth_local_webserver
```

use the link on your local machine to get the authorization code. This will generate *sheets.$HOME/.credentials/drive-python-quickstart.json*

- Replace the *Scorecard-Automation-Project/Deliverable/.credentials* folder with the one on the instance at *$HOME/.credentials*
- Push all your changes to your branch on the Git Server

IV. Creating a New Scorecard

This part of the document explains how to create a new Scorecard using an existing one as a template. Pay close attention, and do nothing that isn't written. Follow my instructions exactly, and you will have great success. This section will contain some information found in **III :: 3.1.A**; however, I will also include instructions for setting up the other components mentioned in **III :: 3.1.C** that were not covered in great detail or were written to be initialized by a script. This guide assumes you are working in a linux environment, specifically AWSLinux v2017.03 using username, ec2-user, with access to code.kent.edu. Let us begin. Note you will need access to the repository this project is hosted under or the server will not be able to see any changes/additions you make.

A note about my documention convention: I write only executable shell code or python code below. Questions regarding a line's meaning can be expected to be answered by Googling the command and the word linux, shell, bash, python, depending on where you are stuck.

1. Run the following line of code on your EC2 instance:

```
git clone
https://jadkin31_stu:Jacksonv12@code.kent.edu/jadkin31_stu/Scorecard-
Automation-Project.git
```

2. Copy the shell directory from *Scorecard-Automation-Project/* and rename it to reflect the name of your new application in the directory, *Scorecard-Automation-Project/Deliverable*:

```
APP_NAME=Learn
cp -r Scorecard-Automation-Project/Shell
Scorecard-Automation-Project/Deliverable/$APP_NAME
```

3. Move into your new directory

```
cd $_
```

4. Open *APPLICATION_install.sh* and replace 'YOUR_APP_HERE' on lines 4 and 9. Ensure that you keep capitalization consistent throughout this tutorial. I would recommend the omission of space characters. The top line of the following code block is an exmaple of how you would use vim to accomplish this for the application, *Learn*.

```
:%s/YOUR_APP_HERE/Learn/g
4: #Application : YOUR_APP_HERE
9: app_name='YOUR_APP_HERE' #No spaces in entire line. Keep the '
marks.
```

5. Rename *APPLICATION_install.sh* to reflect the application name you entered in **IV::2** and **IV::4**

```
mv APPLICATION_install.sh "$APP_NAME"_install.sh
```

6. If your application does not already have a screenshot script in the git project housed here, you'll need to create one and push it to this project or edit the line that fetches the screenshot scripts package (*Scorecard-Automation-Project/Deliverable/$APP_NAME/APPLICATION_screenshots_bootstrap.sh:7*). If you push a new script to this package, ensure you add it to the branch entitled, *Automation_Screenshot_Scripts.* Ensure your script is written to be executed in a linux environment. If the script uses runtimes other than PhantomJS v2.1.1 and CasperJS1.1.4, you'll need to set their install scripts/commands to run during the app's *_install.sh* file (*Scorecard-Automation-Project/Deliverable/$APP_NAME/APPLICATION_install.sh:*14-27).

7. Browse to your AWS Control Pannel and add a bucket that will be used to hold the screenshots for this application. You add Versioning, Tags, and Logging, if you'd like; although there is really no need for them as the files you're going to be uploading likely won't have more versions or be staying around long. Give the owner all permissions, and the default options for other settings will suffice.

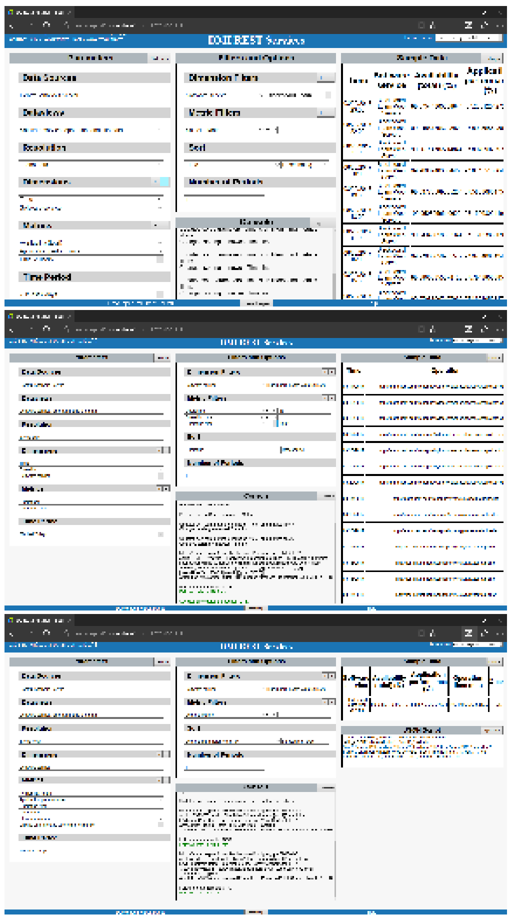
8. Open *APPLICATION_screenshot_bootstrap.sh* and replace add the S3 bucket's name designated to hold the screenshots for this application. Also add the execution command for your script on line 11 (remove the '#' at the beginning of the line), and upload all of your screenshots to S3 just after line 18 where the example is provided. You'll need to change the name of the file you're uploading to correspond to each image you need. Learn, for example has 8 screenshots so we will have 8 lines. In each line, we need to specify the file name on the local server, the S3 bucket name, and a filename to delete from the local host. See my example changes below (the right side reflects the change set).

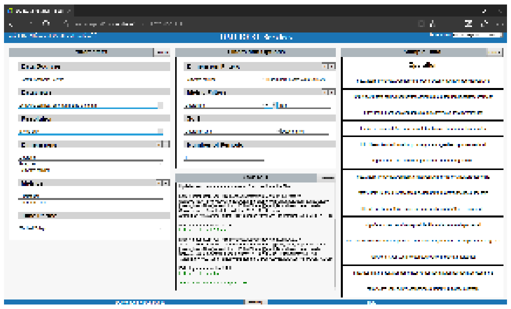
9. Rename *APPLICATION_screenshot_bootstrap.sh* to reflect the application name you entered in **IV::2** and **IV::4**

```
mv APPLICATION_screenshot_bootstrap.sh
"$APP_NAME"_screenshot_bootstrap.sh
```

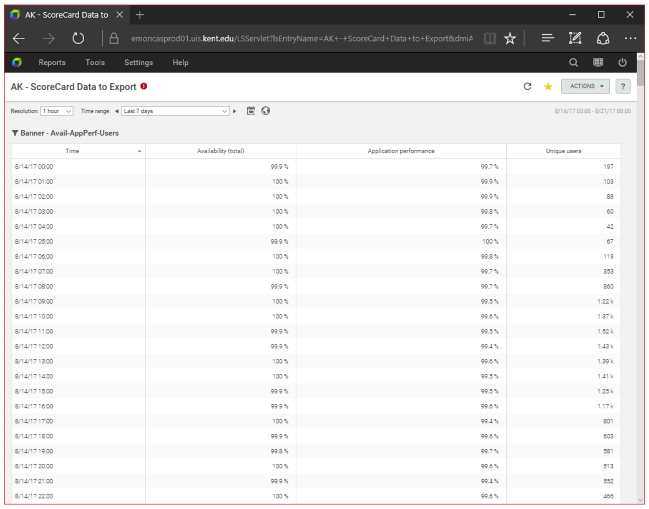
10. Enter the *lib* directory. We need to add functions to our libraries to support API calls for the new application. This directory contains six items. Detailed instructions on updating these files so that they work in general can be found in **III** (Changing credentials for 3rd party servers to gain API access).
    a. ConfluenceAPI
        - This file should not require any edits
    b. DCRumAPI
        - Use the DCRumAPI Helper to create the required queries for your scorecard. You'll see by examining the file that Banner was completed using eight queries. I will duplicate all of these for learn, and include screenshots of the Helper below for each one. I would reccomend sticking to the naming convention I have established; that is, keeping the function name the same as the others, and appending the first three letters of the app name to the end of the function name. Once you create your query, you'll need to copy it, and paste it into a new function in this file. I'd reccomend copying the old function, changing the function name, and then changing the URL. The rest of the function need stay the same. **Remark:** You'll need to escape all of the *'* characters. That is, put a \ infront of each one so that you have *\'* instead of *'* with the exception of the *'* at the beginning and end of the URL.

```
def New_Function_App():
        url =
'https://emoncasprod01.uis.kent.edu/rest/dmiquery/getDMIDat
a3?appId=CVENT&viewId=ClientView&dimensionIds=[\'begT\',\'p
Url\',\'appl\']&metricIds=[\'trans\',\'transTime\']&resolut
ion=d&dimFilters=[[\'appl\',\'Banner eProd SSB App
Servers\',false]]&metricFilters=[[\'trans\',\'>\',10,1],[\'
transTime\',\'>=\',8000,1]]&sort=[[\'pUrl\',DESC]]&topFilte
r=1000&timePeriod=7D&numberOfPeriods=1&dataSourceId=ALL_AGG
R'; #Your new query from DCRumAPI Helper

        r = requests.get(
                url,
                auth=(user_, pass_)
        );

        return (str(r.text));
```





Note that for the the banner bucket list intervals, it's just a matter of changing the metric filters to correspond to your query. It may help you to know that you can view the details of a query by going to a pre-made report



AK - ScoreCard Data to
emoncasprod01.uis.kent.edu/LSServlet?lsEntryName=AK+-+ScoreCard+Data+to+Export&dmiA
Reports
Tools
Settings
Help
ACTIONS
?

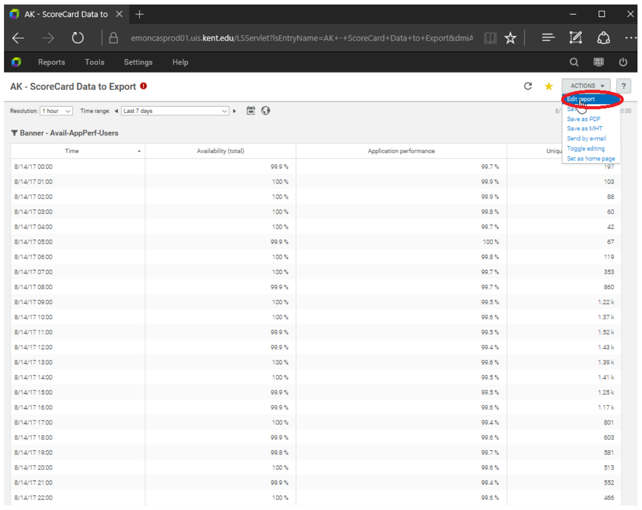
AK - ScoreCard Data to Export

Resolution: 1 hour Time range: Last 7 days 8/14/17 00:00 - 8/21/17 00:00

Banner - Avail-AppPerf-Users

Time	Availability (total)	Application performance	Unique users
8/14/17 00:00	99.9 %	99.7 %	197
8/14/17 01:00	100 %	99.9 %	103
8/14/17 02:00	100 %	99.9 %	88
8/14/17 03:00	100 %	99.8 %	60
8/14/17 04:00	100 %	99.7 %	42
8/14/17 05:00	99.9 %	100 %	67
8/14/17 06:00	100 %	99.8 %	119
8/14/17 07:00	100 %	99.7 %	353
8/14/17 08:00	99.9 %	99.7 %	860
8/14/17 09:00	100 %	99.5 %	1.22 k
8/14/17 10:00	100 %	99.6 %	1.37 k
8/14/17 11:00	99.9 %	99.5 %	1.52 k
8/14/17 12:00	99.9 %	99.4 %	1.43 k
8/14/17 13:00	100 %	99.6 %	1.39 k
8/14/17 14:00	100 %	99.5 %	1.41 k
8/14/17 15:00	99.9 %	99.5 %	1.25 k
8/14/17 16:00	99.9 %	99.6 %	1.17 k
8/14/17 17:00	100 %	99.4 %	801
8/14/17 18:00	99.9 %	99.6 %	603
8/14/17 19:00	99.8 %	99.7 %	581
8/14/17 20:00	100 %	99.6 %	513
8/14/17 21:00	99.9 %	99.4 %	552
8/14/17 22:00	100 %	99.6 %	466

Clicking edit



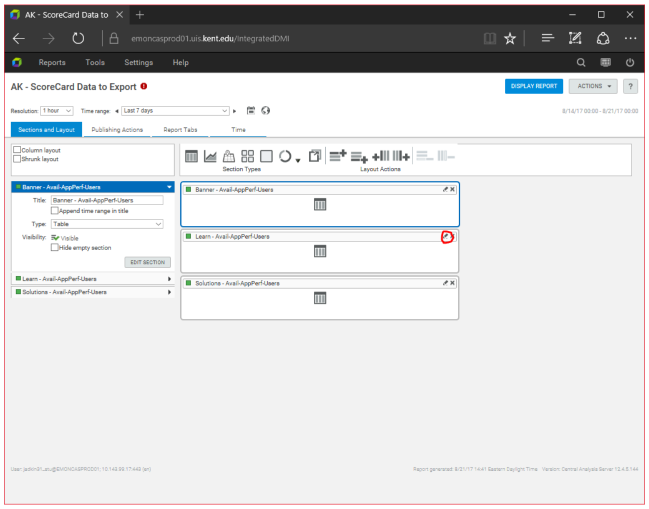
AK - ScoreCard Data to Export

Resolution: 1 hour Time range: Last 7 days

Banner - Avail-AppPerf-Users

Time	Availability (total)	Application performance	Unique users
8/14/17 00:00	99.9 %	99.7 %	197
8/14/17 01:00	100 %	99.9 %	103
8/14/17 02:00	100 %	99.9 %	88
8/14/17 03:00	100 %	99.8 %	60
8/14/17 04:00	100 %	99.7 %	42
8/14/17 05:00	99.9 %	100 %	67
8/14/17 06:00	100 %	99.8 %	119
8/14/17 07:00	100 %	99.7 %	353
8/14/17 08:00	99.9 %	99.7 %	860
8/14/17 09:00	100 %	99.5 %	1.22 k
8/14/17 10:00	100 %	99.6 %	1.37 k
8/14/17 11:00	99.9 %	99.5 %	1.52 k
8/14/17 12:00	99.9 %	99.4 %	1.43 k
8/14/17 13:00	100 %	99.6 %	1.39 k
8/14/17 14:00	100 %	99.5 %	1.41 k
8/14/17 15:00	99.9 %	99.5 %	1.25 k
8/14/17 16:00	99.9 %	99.6 %	1.17 k
8/14/17 17:00	100 %	99.4 %	801
8/14/17 18:00	99.9 %	99.6 %	603
8/14/17 19:00	99.8 %	99.7 %	581
8/14/17 20:00	100 %	99.6 %	513
8/14/17 21:00	99.9 %	99.4 %	552
8/14/17 22:00	100 %	99.6 %	466

And clicking on the pencil



AK - ScoreCard Data to Export

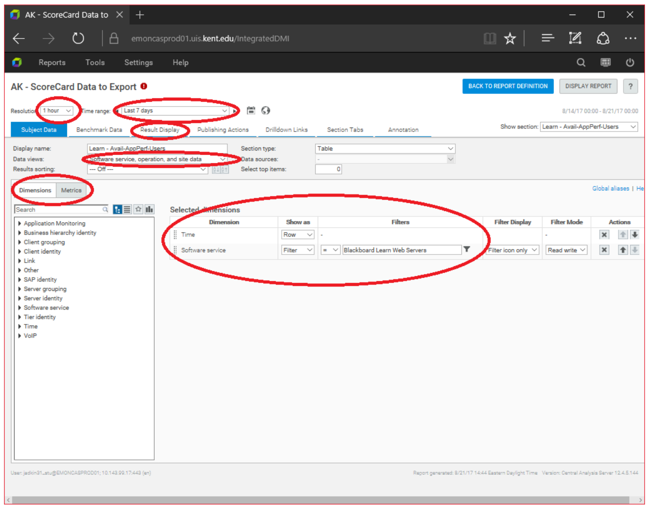
emoncasprod01.uis.kent.edu/IntegratedDMI

Resolution: 1 hour | Time range: Last 7 days

Title: Banner - Avail-AppPerf-Users
Type: Table
Visibility: Visible

Sections: Banner - Avail-AppPerf-Users; Learn - Avail-AppPerf-Users; Solutions - Avail-AppPerf-Users

This will give you access to some of the data you need to enter like Resolution, Time range, Dimensions, Metrics, Metric Filters, Dimension Fileters, and Data views. You will always need to select Central Data Server for the first option.



AK - ScoreCard Data to Export

emoncasprod01.uis.kent.edu/IntegratedDMI

Resolution: 1 hour | Time range: Last 7 days | 8/14/17 00:00 - 8/21/17 00:00

Show section: Learn - Avail-AppPerf-Users

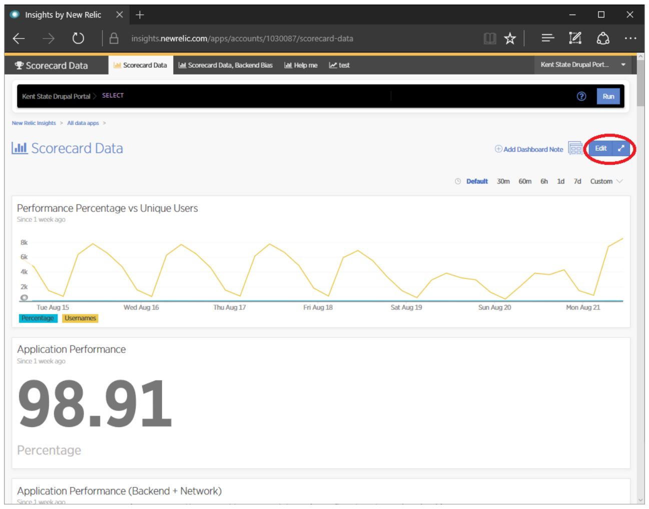
Display name: Learn - Avail-AppPerf-Users
Data views: Software service, operation, and site data
Results sorting: --- Off ---
Section type: Table
Data sources: -
Select top items: 0

Selected dimensions

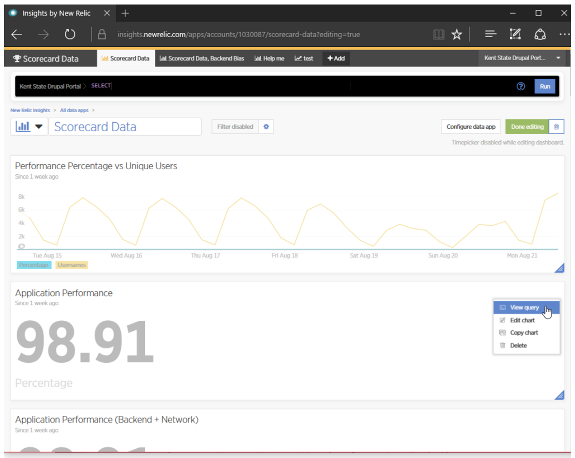
Dimension	Show as	Filters	Filter Display	Filter Mode
Time	Row	-		-
Software service	Filter	= Blackboard Learn Web Servers	Filter icon only	Read write

c. DCRumHelpers
   - This file should not require any edits.
d. GoogleSheetsAPI
   - Add your app name on line 23, and replace YOUR_APP_HERE on line 13
e. Helpers
   - Add your app name on line 6.
f. NewRelicAPI
   - If the application you're creating a scorecard for requires infromation from New Relic, create the NRQL query, or copy it from a report, and append it to the URL: https://insights-api.newrelic.com/v1/accounts/1030087/query?nrql= **Remark:** You will have to URL encode your NRQL query. Below is a tutorial on generating a NRQL query from a report.

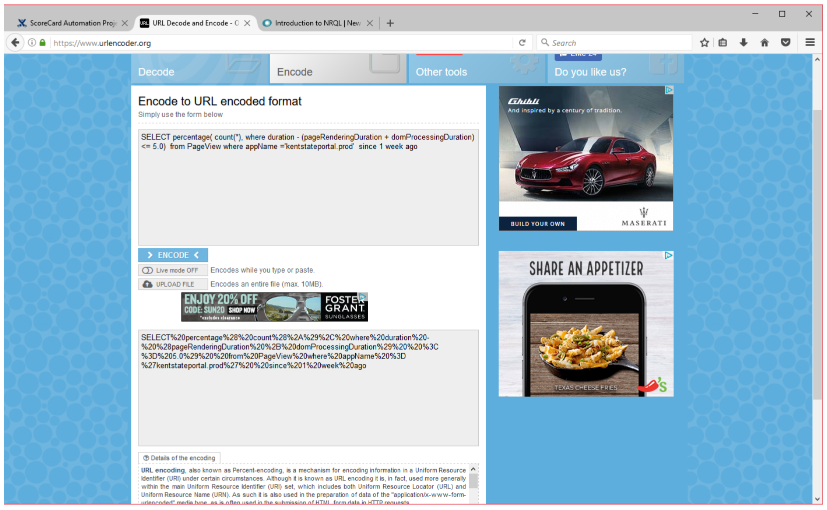
     Navigate to a report and click edit


Insights by New Relic
insights.newrelic.com/apps/accounts/1030087/scorecard-data
Scorecard Data
Scorecard Data, Backend Bias
Help me
test
Kent State Drupal Port...
Kent State Drupal Portal > SELECT
Run
Scorecard Data
Add Dashboard Note
Edit
Default 30m 60m 6h 1d 7d Custom
Performance Percentage vs Unique Users
Since 1 week ago
Tue Aug 15
Wed Aug 16
Thu Aug 17
Fri Aug 18
Sat Aug 19
Sun Aug 20
Mon Aug 21
Percentage
Usernames
Application Performance
Since 1 week ago
98.91
Percentage
Application Performance (Backend + Network)

Click in the top left-hand corner of the box you wish to view the query for, a drop down menu should appear. Click on "View query". A black box will appear at the top of the page (you might have to scroll up).


Insights by New Relic
insights.newrelic.com/apps/accounts/1030087/scorecard-data?editing=true
Scorecard Data
Scorecard Data, Backend Bias
Help me
test
+ Add
Kent State Drupal Portal > SELECT
Run
Scorecard Data
Filter disabled
Configure data app
Done editing
Timepicker disabled while editing dashboard
Performance Percentage vs Unique Users
Application Performance
98.91
Percentage
View query
Edit chart
Copy chart
Delete
Application Performance (Backend + Network)

Copy the query (not including the "Kent State Drupal Portal >" and paste it into the "Type or paste here" box on this site. Click encode, and paste the result into the URL in your new function as in the DCRumAPI example given above. See code below


ScoreCard Automation Proj
URL Decode and Encode - O
Introduction to NRQL | New
https://www.urlencoder.org
Decode
Encode
Other tools
Do you like us?
Encode to URL encoded format
Simply use the form below
ENCODE
Live mode OFF
Encodes while you type or paste.
UPLOAD FILE
Encodes an entire file (max. 10MB).
Details of the encoding

```
SELECT percentage( count(*), where duration - (pageRenderingDuration + domProcessingDuration)
<= 5.0)  from PageView where appName ='kentstateportal.prod'  since 1 week ago
```

```
SELECT%20percentage%28%20count%28%2A%29%2C%20where%20duration%20-
%20%28pageRenderingDuration%20%2B%20domProcessingDuration%29%20%20%3C
%3D%205.0%29%20%20from%20PageView%20where%20appName%20%3D
%27kentstateportal.prod%27%20%20since%201%20week%20ago
```

URL encoding, also known as Percent-encoding, is a mechanism for encoding information in a Uniform Resource Identifier (URI) under certain circumstances. Although it is known as URL encoding it is, in fact, used more generally within the main Uniform Resource Identifier (URI) set, which includes both Uniform Resource Locator (URL) and Uniform Resource Name (URN). As such it is also used in the preparation of data of the "application/x-www-form-urlencoded" media type, as is often used in the submission of HTML form data in HTTP requests.

```
def New_Func():
    url =
'https://insights-api.newrelic.com/v1/accounts/1030087/quer
y?nrql=SELECT%20percentage%28%20count%28%2A%29%2C%20where%2
0duration%20-%20%28pageRenderingDuration%20%2B%20domProcess
ingDuration%29%20%20%3C%3D%205.0%29%20%20from%20PageView%20
where%20appName%20%3D%27kentstateportal.prod%27%20%20since%
201%20week%20ago' #Note that the query is APPENDED to the
insights URL
    headers = {'Content-Type' : 'application/json',
'X-Query-Key' : queryKey}

    r = requests.get(
        url,
        headers=headers
    )

 return r.text
```

11. Enter the *Graphs* directory

```
cd ../Graphs
```

12. Open the *driver.sh* file and enter the app name you've been using throughout this guide on line 4

```
4: APP_NAME=Learn
```

Remark: Ensure you do not use spaces in this line of code.

13. Open the *bucketUpload.sh* file and enter the app name, and the bucket name you intend to use with this application on lines 4 and 5 respectively.

```
4: App_Name=Learn
5: BUCKET_NAME=emon-scorecard-learn
```

Remark: Ensure that you do not use spaces in these lines of code

14. Open *FrontPage.py* and enter the RDS credentials for the RDS server housing the data that will be used to generate the front page graph for this application (lines 7-10). Ideally, this is the RDS instance mentioned in **III :: 3.1.B**. You will also need to enter the app name as in the last step (line 14). See below.

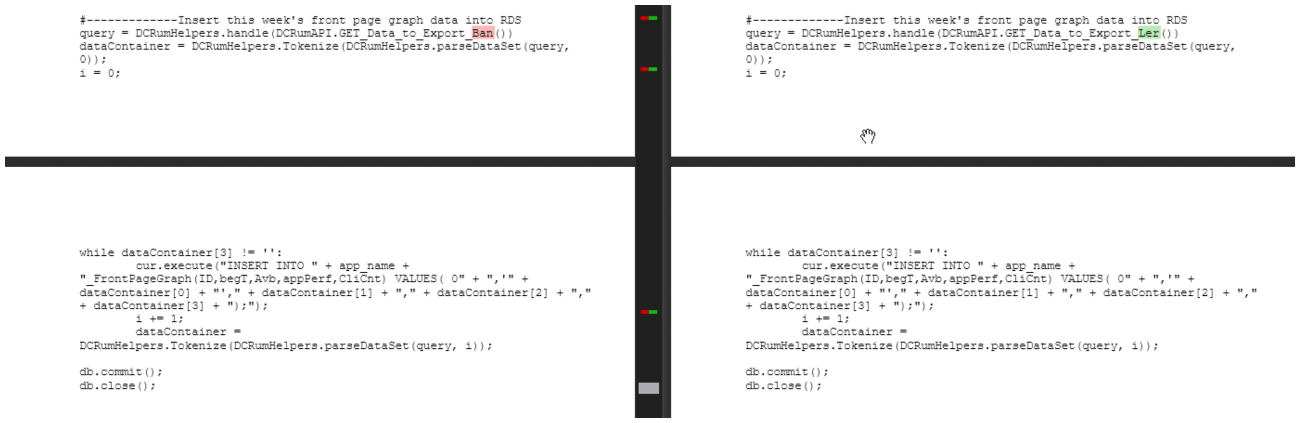
```
06: db = MySQLdb.connect(
07:
host="emonscorecards.cqn2vtdstij1.us-east-2.rds.amazonaws.com",     #
your host, usually localhost
08:                        user="scorecarduser",
# your username
09:                        passwd="cheeseburger",
# your password
10:                        db="scorecards");
# name of the database

12: cur = db.cursor();

14: app_name = 'Learn'; #keep the 's
```

15. Navigate back to the $App_Name directory, and open the *data_collector.py* file.

```
cd ../
```

16. Change the part of the file path on line 5 to reflect the new application's directory, and remove the TODO comment. Also add your application's name on line 19. You're also going to need to change the functions used in this file to reflect your new application. See Below.

Middle Section of Document Has No Changes so it is Omitted.



```
#-------------Insert this week's front page graph data into RDS
query = DCRumHelpers.handle(DCRumAPI.GET_Data_to_Export_Ban())
dataContainer = DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query,
0));
i = 0;
```

```
#-------------Insert this week's front page graph data into RDS
query = DCRumHelpers.handle(DCRumAPI.GET_Data_to_Export_Ler())
dataContainer = DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query,
0));
i = 0;
```

```
while dataContainer[3] != '':
        cur.execute("INSERT INTO " + app_name +
"_FrontPageGraph(ID,begT,Avb,appPerf,CliCnt) VALUES( 0" + ",'" +
dataContainer[0] + "'," + dataContainer[1] + "," + dataContainer[2] + ","
+ dataContainer[3] + ");");
        i += 1;
        dataContainer =
DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query, i));

db.commit();
db.close();
```

```
while dataContainer[3] != '':
        cur.execute("INSERT INTO " + app_name +
"_FrontPageGraph(ID,begT,Avb,appPerf,CliCnt) VALUES( 0" + ",'" +
dataContainer[0] + "'," + dataContainer[1] + "," + dataContainer[2] + ","
+ dataContainer[3] + ");");
        i += 1;
        dataContainer =
DCRumHelpers.Tokenize(DCRumHelpers.parseDataSet(query, i));

db.commit();
db.close();
```

You might try making the application name changes, and then running the following vim command:

```
:%s/Ban()/Ler()/g
```

17. If your application doesn't already have one, you'll need to create a table for the application in the RDS instance. Sign into a MySQL Management Studio (like MySQL Workbench). Below are two create table queries. You'll need to tweak, and execute both of them. The comments provided in the code should prove sufficient to guide you through the required changes.

```
use scorecards;

CREATE TABLE APPLICATION_FrontPageGraph( #Replace APPLICATION with the
name of the application (the one used in IV::2)
    ID                 INT                NOT NULL    AUTO_INCREMENT,
    begT            DATETIME        NOT NULL,
    avb                FLOAT            NOT NULL,
    appPerf            FLOAT            NOT NULL,
    CliCnt            FLOAT            NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE APPLICATION_FrontPage( #Replace APPLICATION with the name
of the application (the one used in IV::2)
    ID                 INT        NOT NULL    AUTO_INCREMENT,
    dataDescription TEXT    NOT NULL,
    dataValue        TEXT,
    dataDate        DATE    NOT NULL,
    PRIMARY KEY (ID)
```

18. Enter the app's Bucketlist Drive directory

```
cd Bucketlist/Drive
```

19. Open the *Create_APPLICATION_Bucketlist.py* and enter the application name on line 22.

20. Rename *Create_Application_Bucketlist.py* to reflect your new application. See below:

```
mv Create_APPLICATION_Bucketlist.py Create_"$APP_NAME"_Bucketlist.py
```

21. Navigate up a directory and open *Build_APPLICATION_Bucketlist.py*. Again, in addition to adding the name of your application to line 4 (as in step 16), you'll have to make multiple function changes in this document as well. If you followed my naming convention, this can be accomplished with a single vim command. See below. See **II** for how this file works, and what you may be able to accomplish with the *Go ogleSheetsAPI*

```
4: sys.path.insert(0, '/home/ec2-user/Learn Scorecard/lib')
```

```
Vim Command:
:%s/Ban()/Ler()/g
```

This command basically converts the query lines:

```
query =
DCRumHelpers.execute_BucketList_Query(DCRumAPI.GET_Bucket_02_Ban());
--> query =
DCRumHelpers.execute_BucketList_Query(DCRumAPI.GET_Bucket_02_Ler());
```

22. Rename *Build_APPLICATION_Bucketlist.py* to reflect your new application

```
mv Build_APPLICATION_Bucketlist.py Build_"$APP_NAME"_Bucketlist.py
```

23. Open *driver.sh* and enter your new app name on line 4 without using any spaces in the line of code.

24. All that is left at this point is to edit *card_builder.py*. You'll likely find **II** to be of more use for this then giving specific line by line examples. If you're creating another DCRum Scorecard, you may with to line by line with another Scorecard, for example Banner, and change function names out as needed. At this point you're ready to follow **V**. Below I offer an explanation, as best I can, to the changes I made to Banner's *card_builder.py* to create the Learn Scorecard.
    - Replace 'YOUR_APP_HERE' with your application name on lines 11 and 29
    - Replace 'YOUR_BUCKET_HERE' with your application's S3 bucket name on line 25
    - Update the *SCREENSHOT_UPLOAD_TARGET* variable on line 26. Set it equal to the number of screenshots you uploaded to S3 (number of *aws cp...* lines you added in **IV::8**)
    - Locate the comment on line 141. This marks where you are to put the rest of the Front Page Data Table. See below. The code below represents a single row of the table which is comprised of two parts: getting the information out of RDS with SQL, and putting into the table with Helper function calls:

```
#App Performance Queries
dataSet = ['-1','-1','-1','-1', '-1'] # a data structure to house
the row's information

#Get the data for last week's availability
cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=7)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[0] = Helpers.handle_SQL(str(cur.fetchall())) # store it
in the data structure

#Get availability for 2 week's ago availability
cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[1] = Helpers.handle_SQL(str(cur.fetchall())); # store it
in the data structure

cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=21)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[2] = Helpers.handle_SQL(str(cur.fetchall()));

cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=28)).strftime('%Y-%m-%d') + "' AND
```

```
dataDescription LIKE 'Availability';");
dataSet[3] = Helpers.handle_SQL(str(cur.fetchall()));

cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=35)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Availability';");
dataSet[4] = Helpers.handle_SQL(str(cur.fetchall()));

#Insert data into table
Helpers.constructPayload("<tr>")
Helpers.constructPayload("<td>Availability<p
style=\\\"font-size:9px\\\">&#40;All Attempts &#45; Failures&#41;
&#47; All Attempts</p></td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[0], dataSet[1], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[0] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[1], dataSet[2], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[1] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[2], dataSet[3], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[2] + "%</td>")
Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[3], dataSet[4], 'Good'))
Helpers.constructPayload("<td
style=\\\"text-align:center;vertical-align:middle;\\\">" +
dataSet[3] + "%</td>")
Helpers.constructPayload("</tr>")
```

Remark: To get other information we need only change SQL call. For example, if we needed Unique Users instead of Availability, we would query the RDS as follows:

```
cur.execute("SELECT dataValue FROM " + app_name + "_FrontPage
WHERE dataDate LIKE '" + (datetime.datetime.today() -
datetime.timedelta(days=7)).strftime('%Y-%m-%d') + "' AND
dataDescription LIKE 'Unique Users';");
```

Also note how we change *days=* argument in the query to change what week's data we are retrieving. Consult the completed *card_builder.py* scripts in *Scorecard-Automation-Project/Deliverable/Learn, Scorecard-Automation-Project/Deliverable/Banner, Scorecard-Automation-Project/Deliverable/Solutions,* or *Scorecard-Automation-Project/Deliverable/Flashline* for more information. Remark, if your application's Front Page Data Table format/information matches that of Learn's, a copy and paste of Learn's Front Page Data Table rows would suffice.

- Locate the commented line:

```
#Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-Apphealth-GraphOnly-" + today + ".png"));
```

Examine the line. It is a nested function call. You need to change the second argument of the inner function, *"SSB-DCA-Apphealth-GraphOnly-" + today + ".png"*, to reflect your image for title added to the card directly above this function (which you can also change if you'd like". You'll likely just need to change the hard coded string, *"SSB-DCA-Apphealth-GraphOnly-".* Remark that *today* is a variable containing today's date in the format mm-dd-yyyy.

- Repeat this until you have put all of your screenshots on the page. Space is give for four screenshots. If you need to add more, this is the block of code that titles, and places an image:

```
print("Adding Benchmark App Health to " + app_name + "
Scorecard...");
Helpers.constructPayload("<h3
style=\\\"text-align:center;font-weight:bold;\\\">Benchmark
Application Health</h3>");
Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-Apphealth-GraphOnly-" + today + ".png"));
print("Done.");
```

The commented lines you need to change after completing the first one are:

```
#Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-OpTime-GraphOnly-" + today + ".png"));
#Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-Availability-GraphOnly-" + today + ".png"));
#Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-Usage-GraphOnly-" + today + ".png"));
```

- Locate the commented line:

```
#dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.parse
DataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_Ban()),
i)));
```

You need to change the API call made to reflect your new application and uncomment the line of code. Note this line of code is populating the data structure that will be used to create the Top Ten Slow URL List

```
dataContainer.append(DCRumHelpers.Tokenize_3(DCRumHelpers.parseD
ataSet(DCRumHelpers.handle(DCRumAPI.GET_Top_Ten_Slow_Ler()),
i)));
```

- Locate the commented line:

```
#Helpers.constructPayload(Helpers.imageTag(id_,
"SSB-DCA-Transaction-GraphOnly-" + today + ".png"));
```

As before, you need to change the name of the image to reflect your new application's Transaction screenshot (If you have one) and uncomment the line. If not make sure you read **II::5**

25. Push all your changes to your branch on the Git Server

Let's discuss pre-conditions for what you've just created. You'll note that in **III**, you verify connections to API servers. This is something you'll need to do for your new application as well. I'd reccomend following the guide for any other application to do this save Flashline, as Flashline doesn't use all the components that others do. You'll also need to make sure sufficient data exists in the RDS instance (5 weeks). If any of these conditions fail, your application will not produce a scorecard.

V. First Run Conditions

Remark: The following steps need to be taken for each application

1. Since we typically pull this week's Number of Changes, Number of Issues Identified, and Number of Critical Events, from *last* week's scorecard, we need comment those lines out in your application's *data_collector.py* before the first run.

```
#--------------Get last week's trailers
#NumChanges0 = Helpers.conf_Unwrapper(ConfluenceAPI.get('Banner
Scorecard - ' + (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%m/%d/%Y')),
'NumChanges0').encode("utf-8");

#CritEvents0 = Helpers.conf_Unwrapper(ConfluenceAPI.get('Banner
Scorecard - ' + (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%m/%d/%Y')),
'CritEvents0').encode("utf-8");

#NumIss0 = Helpers.conf_Unwrapper(ConfluenceAPI.get('Banner Scorecard
- ' + (datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%m/%d/%Y')),
'NumIss0').encode("utf-8");

#---------------Insert last week's trailers
#cur.execute("INSERT INTO
Banner_FrontPage(ID,dataDescription,dataValue,dataDate)
VALUES(0,'Number of Changes','" + NumChanges0 + "'," +
(datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y%m%d') + ");");

#cur.execute("INSERT INTO
Banner_FrontPage(ID,dataDescription,dataValue,dataDate)
VALUES(0,'Number of Critical Events','" + CritEvents0 + "'," +
(datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y%m%d') + ");");

#cur.execute("INSERT INTO
Banner_FrontPage(ID,dataDescription,dataValue,dataDate)
VALUES(0,'Number of Issues Identified','" + NumIss0 + "'," +
(datetime.datetime.today() -
datetime.timedelta(days=14)).strftime('%Y%m%d') + ");");
```

2. Complete **III**
3. Create a scorecard.
4. Uncomment the lines we commented in step one.
5. Replace the "DAVE"s in the Front Page table with the appropriate values, and enter Log information.

VI. Troubleshooting, Known Issues, and Error Messages

PUT Status Code: 400

- If your Confluence PUT request (currently is last line printed by *card_builder.py*) returns status code 400, one of two things is likely wrong
    1. There is already a page in the space on the Confluence Server with the page name you're attempting to use
        - Log into Confluence and make sure you're not trying to update to name that already exists
    2. You have an error in your HTML
        - Check over your code and ensure that all open tags are closed, all quote and tick marks are properly escaped,

etc.
- Use *""" python code """* to coment out multiple lines of Pyhon code and see how much of Scorecard is written properly. After a few tests, you should be able to identify the line giving you the issue, and fix the HTML.
- Make sure your *update.txt* is being deleted at the end of execution

ValueError

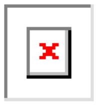
- If you get a response resembling:

```
Traceback (most recent call last):
  File "/home/ec2-user/Banner Scorecard/card_builder.py", line
446, in <module>
    main(sys.argv[1:])
  File "/home/ec2-user/Banner Scorecard/card_builder.py", line
130, in main

Helpers.constructPayload(Helpers.resolve_Statistical_Inequality(
dataSet[0], dataSet[1], 'Good'))
  File "/home/ec2-user/Banner Scorecard/lib/Helpers.py", line
165, in resolve_Statistical_Inequality
    if float(new_) > float(old_):
ValueError: could not convert string to float:
```

  It's likely that you are missing a data entry in your RDS instance. Examine the line number given in the error (in this case it is 130) and check it in the source code. In this case we see that *dataSet[0], dataSet[1]*, or both is/are causing the issue. So we look at the assignment statement and use a MySQL Mangement Studio to check and make sure that, that data exists in the correct RDS table. If it does not, add it. This should fix the error.
- This occurs because if the data is absent, the query returns *NULL* or an empty string which cannot be converted to a float value for comparison.

Invalid Images

- If all or any of the images on the Confluence page are appearing like this

  Ensure that you have attached the image before using it in the document, and that the file names are the same for attaching and placing the image.

Next

- Hi

VII. RDS Database Schema

1) APPLICATION_FrontPage

ID	dataDescription	dataValue	dataDate

2) APPLICATION_FrontPageGraph

ID	begT	avb	appPerf	CliCnt

VIII. Developer Account Dependencies

This section lists all of the accounts that are dependent on my login. When my account is deleted, you'll need to go through **III** with another account.

- DCRum
- Confluence
- NewRelic
- S3
- code.kent.edu (**Remark:** This contains the git project)