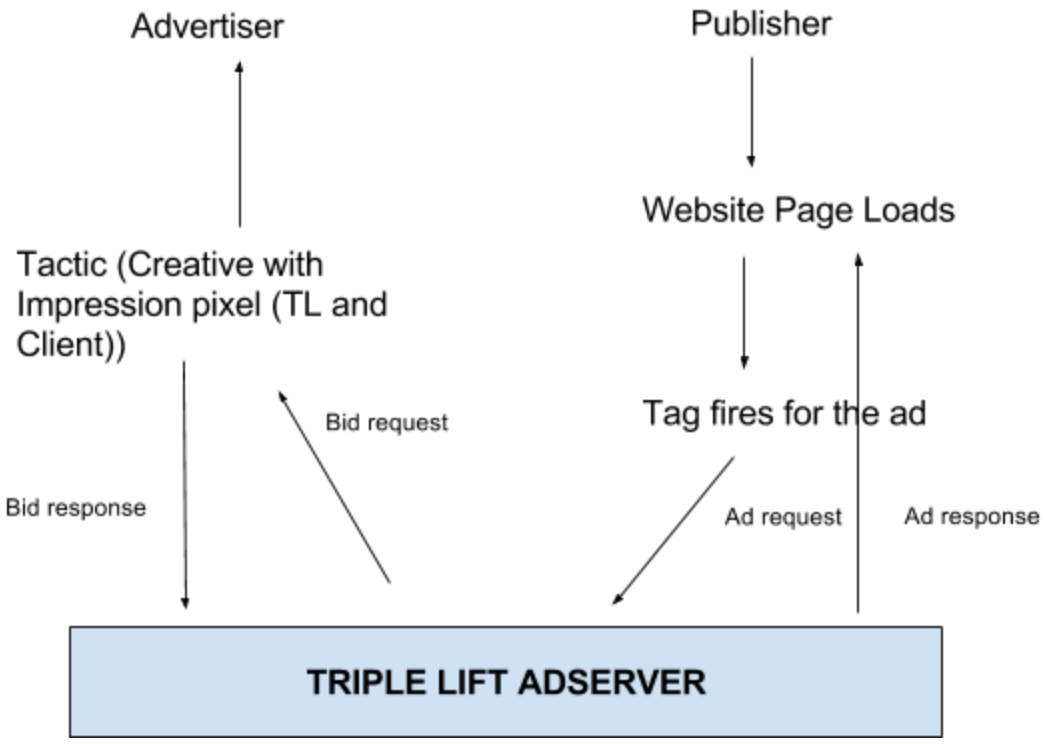
Solution Challenge

Prompt: Illustrate the flows(on paper, google slides,etc) to depict the mechanisms and to visualize the problem.

Flow-Chart (Adserver working ecosystem schema)



Flowchart detailed explanation:

(NOTE: From my past experience in the RTB ecosystem and the TripleLift Adserver working description mentioned in the problem, I've come to the following understanding points which solves the problem. I tool the elements which are used to solve the current problem.

In general there are other components too in the flowchart which complete the RTB working schema, which I didn't mentioned.)

1. When a publisher page loads which has TripleLift code (THE JS) it fires and sends a request to fetch an ad.

2. The call is made to the TripleLift Adserver (**Ad request**)
3. The TripleLift Adserver makes a call to Advertiser to fill the ad slot by a creative (**Bid Request**)
4. The Advertiser UI, searches for the best tactic (which meets all the targeting criteria) and looks for the creative in the respective tactic (While setting up the tactic, the client places third-party impression pixel with the creative) (**Trafficking**)
5. Once the creative is eligible to serve, it enters into the auction and the response (**Bid response**).
6. Once the creative wins the auction, **Ad response** is sent to the publisher page to render the Creative on the page.
7. Once the Ad is rendered/displayed, the Adserver drops 1x1 impression pixels, to record the impression in the TL analytics.
8. When the TL pixel is dropped, it also drops the Third-party impression pixel hosted by the Advertiser, so that the Third party can record the impression from their funnels.

This is how the counting of impressions happens in the TripleLift and the Third-party Adserver.

DISCREPANCY Problem Solving

Prompt: Analyze this data to diagnose the source of the data misalignment and issue a correspondence to the communicate this out to relevant parties (so who are the internal or external parties? what do you tell them?).

Problem Summary:

- Client is comparing the Triple Lift reporting data to the Third- party reporting data.
- There is a discrepancy of 4.45% (Triplelift showing higher number compared to the client data). Why?

Reports (Provided):

- Log level report from TripleLift (with detailed dimensions)
- Client log report (Doesn't have much dimensions, but solves the problem).

Troubleshooting steps and My findings:

- Downloaded the report (from dropbox link) and Opened it in Excel.
- Analysed the impression reported from the reports and stated them in below table.

Report Analysis	**TripleLift**	**Third party**
Total number of impressions served	9917	9475
Difference	9917 - 9475 = 442	
Discrepancy percentage calculation	(442/9917) x 100 = 4.45%	

Confirmation: So, there is a 4.45% discrepancy is present

- Analysed the shared Third party report dimensions (**Domain_Name, Browser_Name, OS_Name**) - (Comment: Reported in the form of Name (not ID). e.g., diply.com, chrome, Android)
- These 3 dimensions are also present in the TripleLift report, however, in TL report, the IDs are retrieved than the respective names - (Comment: Reported in the form of ID. e.g., (36773, 3, 3)

- However, I noticed that there are 3 sheets in the dropbox in the IDs respective names are mentioned (TL_Domain_mapping, TL_OS_mapping, TL_Browser_mapping) (examples mentioned the above point comment).
- With the help of the sheet, I've mapped the IDs to get the names as Output - Comment
- Mapping of the columns in excel formula is used as =VLOOKUP(V2,TL_OS_MAPPING!A$2:B$12,2,FALSE)
- Changed the domain value of N/A's and Null to UNKNOWN in TripleLift and Third-party report.
- Concatenated the 3 columns with "|" delimiter.
- Compared the data with the Third Party report and filtered the Missed and Matched data by VLOOKUP
- Screenshots of Mapping - Link

Report: Link

Output:

- Now, the reports of Third party and TripleLift were in same format dimensions and the values in the cells are in same format too (NAME).
- Report (Which has Mapped columns - Orange colour).

(Note: In the Last column, Thirdparty exist means the impression combination is matched with the TripleLift impression
Missed means the combination is not present in the Third Party report, however, present in the TripleLift report).

Mapping:

- After Mapping the both the reports, **442 impressions were unmapped** (Not Matched).
- Which means the 9475 impressions recorded in the Third party report are also recorded in the TL reporting (**Matched**)
- Not matched **442 impressions are recorded** from the **DOMAIN nam**e: **politicalwire.com**
- Therefore, the suspect domain is politicalwire.com in TripleLift.

Communication:

Internal (to TripleLift): To check if TL serve is properly dropping the Third party 1x1 impression pixel

External (to client): To double-check if they've selected from the DOMAIN name: **politicalwire.com** while pulling the reports or, is its reportable site or non-reportable site in their reporting UI (In some scenarios this situation may happen. In this problem, I've taken the scenario of the domain is reportable in Third-party to solve the problem further in the next question).

Prompt: Run simulations using this forced mechanism and exploring pixel drops to confirm your suspicions from part 1. Provide a clear recommendation to the relevant stakeholders / parties.

Further Troubleshooting steps:

- I've checked the **politicalwire.com and http://makersalley.com/search/?tripleliftTest=true&tl_tactic_id=343664,** and shared my observation in the below table by analysing the **/qa and the Network console (Inspect element)**.
- Debugged the console and checked the calls (Impression pixel drop and its firing).

About Third-party Impression Pixel	**Political Wire (Domain)**	**Makers Sally (Domain)**	**Observation**
Third-party pixel dropped by TripleLift.	Yes	Yes	Yes (Perfectly dropped) and impression count is recorded in TripleLift reporting.
Did the dropped pixel fire?	No	Yes	TripleLift recorded and Third-party din't record. (Website and Pixel http://, https:// compatible)

Observation:

- The pixel dropped is "http://", however, political wire domain is "https://"
- Because of the Mixed SSL (http://) impression didn't fire, so impression count is not recorded to the Third-party reporting.
- Screenshot - Link

To Client:

- Educate the client about the SSL and non-SSL behaviour of the Political wire Domain and Third-party impression pixel.

- The discrepancy is intended (no issues from TripleLift reporting).
- As, the site didn’t rendered the third-party pixel to fire, suggest the client to reach out to the Third-party pixel provider to revise the pixel and make it compatible, so that it can report the impressions from the **Politicalwire.com Domain**.

D. Prompt: Write a script that will programmatically check whether all impressions pixels are valid.

(Note: To be honest, I hardly know about the languages (PHP, C, JAVA etc) because I'm not from computer science background! However, I understand the workflow of the code and to solve the problem, I took a help of a programming friend in code (who helped me with syntax and function)!

Rest Ideation, flow of the code, result analysis was done by me.)

Workflow:

- Download the Tactic Sheet from dropbox.
- Convert the file to .CSV
- Identify which code language to use.
- Identify Impression pixel structure in the cells and what to clean the data.
- List down the required language packages.
- List down the functions that needed to solve the problem.
- Predict and print the results format.
- Generate the output file.

Software Requirements:

- Install Anaconda Continuum (Jupiter notes Application will be available).
- Launch the Jupiter notes application.

Working:

If the below screenshots are not clear, please refer the link for reference (file:///Users/nithinchowhan/Downloads/Final+Assignment+Solution.html)

1. Required Packages:

Required Packages

```
In [8]: import requests
        import pandas as pd
        import numpy as np
        import os
        import re
        import validators
        os.chdir("C:\\Users\\ganreddy\\Documents\\R Scripts\\Nithin\\")
```

2. Ingest the CSV file:

Ingest the CSV file

In [20]:
```
Data = pd.read_csv('C:\\Users\\ganreddy\\Documents\\R Scripts\\Nithin\\tactic.csv',sep=',');
ReqData=Data[["tactic_id","impression_pixel_json"]]
ReqData.head()
```

Out[20]:

	tactic_id	impression_pixel_json
0	333304	["https:\/\/ad.doubleclick.net\/ddm\/adN7676...
1	337773	["https:\/\/ad.doubleclick.net\/ddm\/trackimp\...
2	333514	[]
3	334448	[]
4	336385	["https:\/\/ad.atdmt.com\/i\/img;p=11142202536...

3. Clean the data:

Clean the data for analysis

In [21]:
```
# Removing first and last 2 charater
ReqData['impression_pixel_json'] = ReqData['impression_pixel_json'].str[2:-2]

#Removing all the black slashs
ReqData['impression_pixel_json']=ReqData['impression_pixel_json'].str.replace("\\", "")

#Removing all the four forward slashs
ReqData['impression_pixel_json']=ReqData['impression_pixel_json'].str.replace("////", "//")

#selecting data with http format and removing others
ReqData = ReqData[ReqData['impression_pixel_json'].str.contains("http",na=False)]

ReqData.head()
```

4. Function to find the Status code, OK and Failed:

Function to find Status codes

In [26]:
```
j=0;
def StatusCodes( str ):

    global j
    j=j+1
    try:
        r = requests.head(str)
        result=r.status_code
        print(result,j)

                # prints the int of the status code. Find more at httpstatusrappers.com :)
    except requests.ConnectionError:
        result="failed to connect"
        print(result,j)


    return result
```

Function to create OK and Failed

In [27]:
```
def codes(agrs):
    if(str(agrs)[:1]=="2" or str(agrs)[:1]=="3" ):
        result = "OK"
    else:
        result = "Failed"

    return result
```

5. Saving the results:

Results to be created ans saved to excel

In [34]:
```
#Grabing Status codes
ReqData['StatusCodes'] = ReqData['impression_pixel_json'].map(StatusCodes)
ReqData.head()
```

```
200 191
200 192
200 193
302 194
302 195
302 196
302 197
302 198
200 199
200 200
200 201
failed to connect 202
200 203
200 204
302 205
200 206
302 207
200 208
302 209
failed to connect 210
```

In [35]:
```
#Checking whether OK or FAILED with the Status codes
ReqData['Status'] = ReqData['StatusCodes'].map(codes)
ReqData.head()
```

```
C:\Users\ganreddy\AppData\Local\Continuum\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-
copy
```

Out[35]:

	tactic_id	impression_pixel_json	StatusCodes	Status
0	333304	https://ad.doubleclick.net/ddm/ad/N7676.791086...	200	OK
1	337773	https://ad.doubleclick.net/ddm/trackimp/N2724....	200	OK
4	336385	https://ad.atdmt.com/i/img;p=11142202536676;a=...	200	OK
6	328277	https://voken.eyereturn.com/pix?1132530	302	OK
7	339051	https://t.mookie1.com/t/v1/imp?migAgencyId=616...	302	OK

6. Print the summary:

Print out the summary results

In [36]:
```
#Print the status of the OK Failed and Timeout
#OK :17042   #Failed :704       #No URLs Present :7161
ReqData['Status'].value_counts()
```

In [41]:
```
#Failed cases are recorderd
FailedReqData = ReqData[ReqData['Status'].str.contains("Failed",na=False)]
FailedReqData.to_csv('C:\\Users\\ganreddy\\Documents\\R Scripts\\Nithin\\tactic_result_Failed.csv', sep=',')
```

In [42]:
```
#complete Data of URLS present
ReqData.to_csv('C:\\Users\\ganreddy\\Documents\\R Scripts\\Nithin\\ReqData.csv', sep=',')
```

Report: Link

Template for Unit Test Case document

Tested By	Nithin
Tested On	6/18/2017

Test case Id	Test case Description	Expected Result	Actual Result
1	Reading the CSV rows	All CSV rows appear	PASS
2	Collect only two columns Tactic ID and Impression URL	Columns that belong to the chosen state appear	PASS
3	Cleaning Data set A. Removing first and last two characters [" "] B. Removing \ characters C. Removing NA / NAN /NULLs	Impression are cleaned	PASS
4	Validating the URL format Before passing to function [Status Codes] to fire	Validation Successful	PASS
5	Firing the validated URLs	Fired the URLs	PASS
6	Collecting Status Codes	Status Codes are updated	PASS
7	Function (codes) to derive status whether OK or FAILED	Status created successfully OK for 2xx or 3xx. Others Failed	PASS
8	Creating statics of OK and FAILED count	Count created successfully	PASS
9	Collecting FAILD TacticID and URLs	Collected successfully	PASS
