ACN TECHNOLOGIES LLC

LEADERS IN ARBITRAGE IDENTIFICATION, NOTIFICATION & EXECUTION

Introduction

- ❖ Current Fintech Landscape
- ❖ Democratizing of the Financial Industry
- ❖ Individualization of Account and Asset Management
- ❖ Personal Arbitrage Opportunity Execution

Executive Summary

01 **Problem**

The inability to quickly identify Arbitrage Opportunities and Execute on them.

02 **Solution**

Arbitrage Notification & Execution Application
Code Name: ANEA

03 **Competitor**

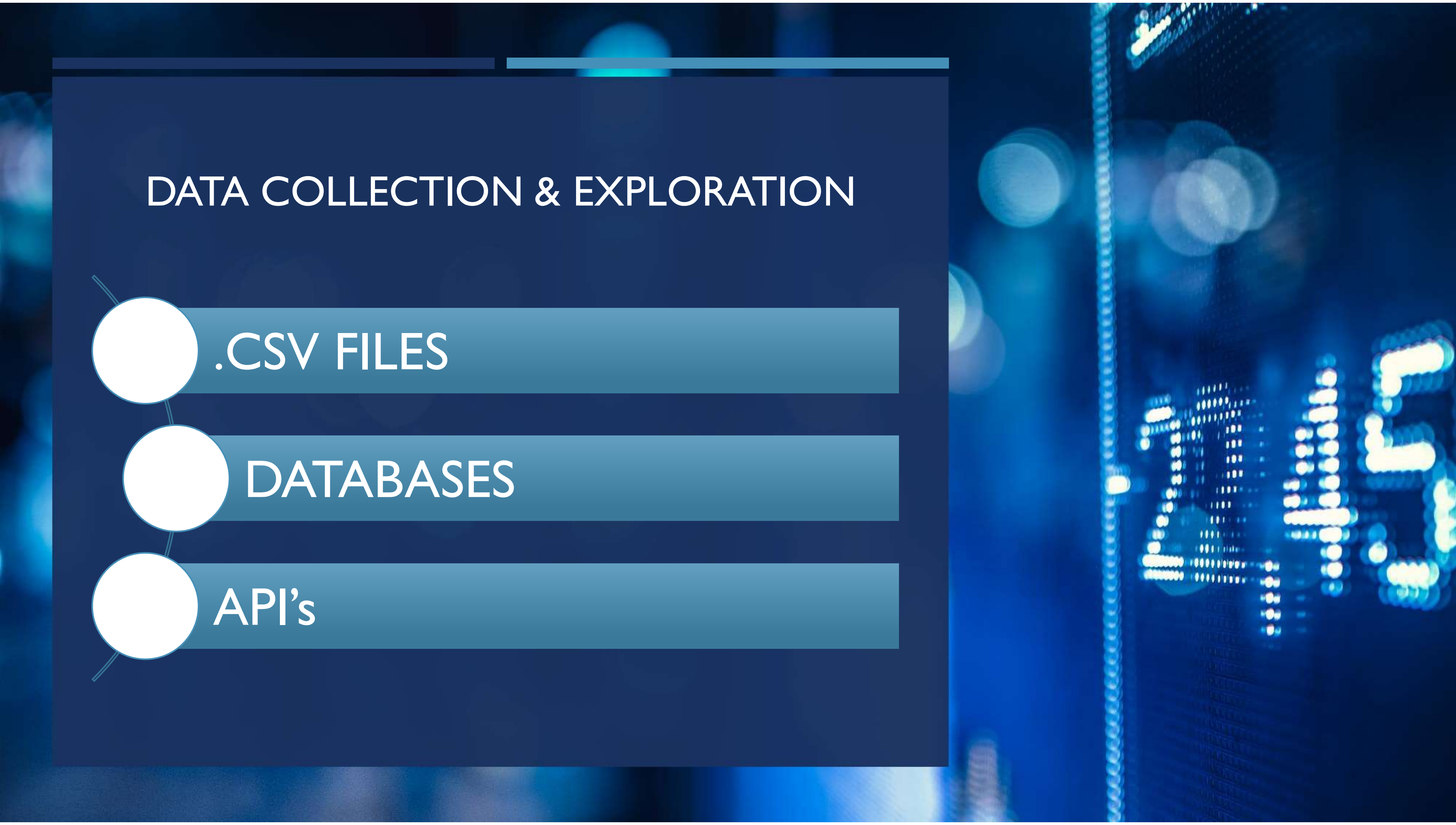
NONE: While there are Arbitrage platforms, there are very few if any user-friendly applications that automate the process.

04 **Key to success**

- Efficiency
- Portability
- Simplicity

DATA COLLECTION & EXPLORATION

- .CSV FILES
- DATABASES
- API's



Application Construction

Connectivity & Notifications

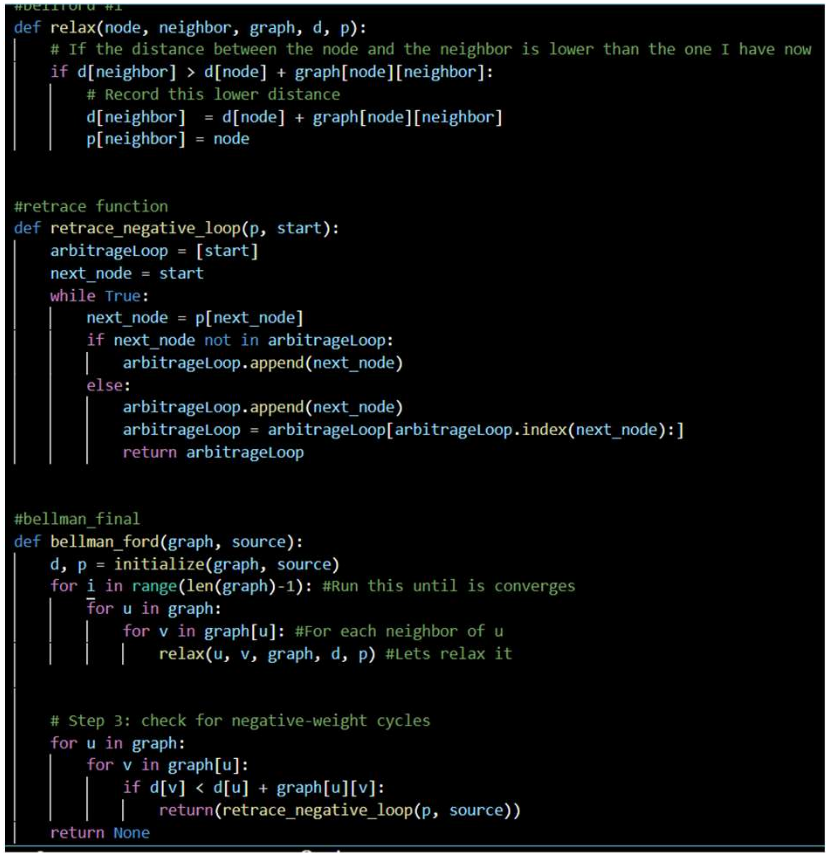
User Friendly Graphical User Interface

APPROACH

INSIGHTS & COMPLICATIONS

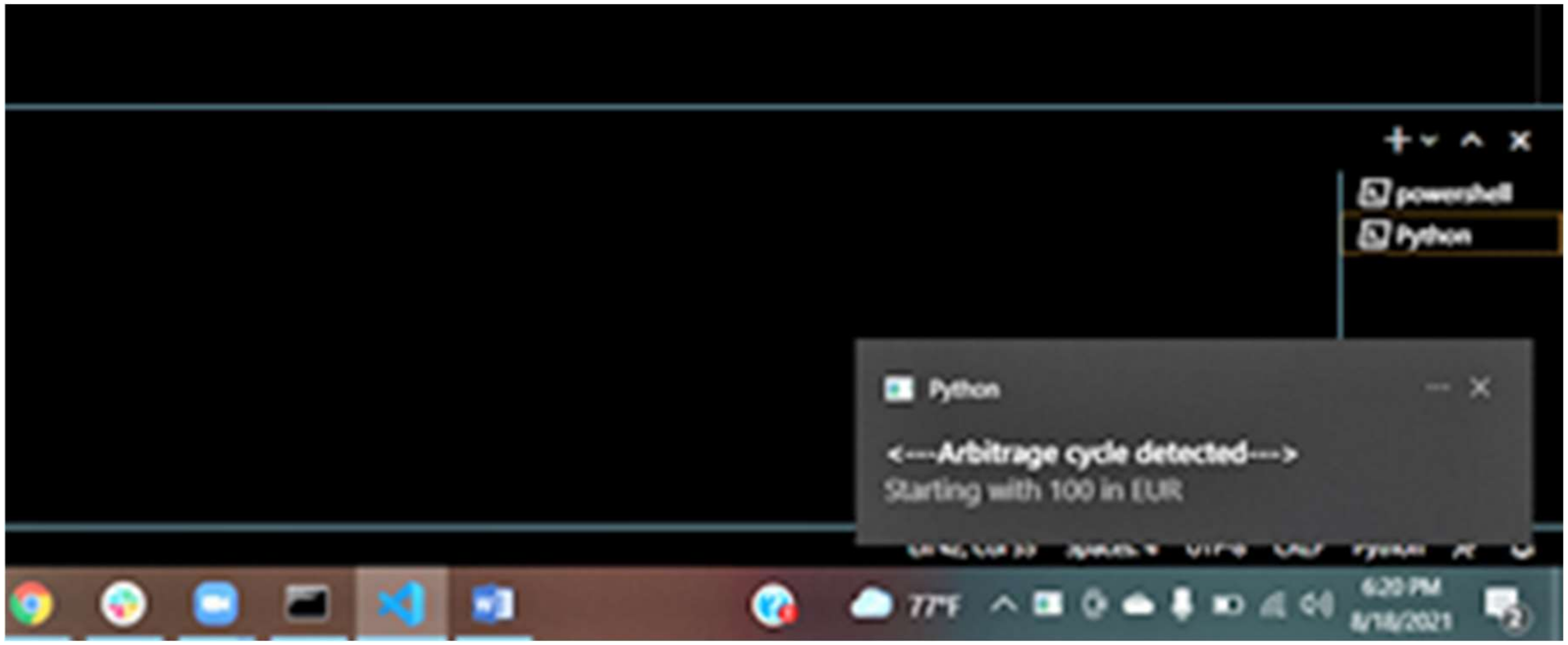
- ❖ Application development is both challenging and rewarding.
- ❖ Finding a suitable API
- ❖ Incorporating the Bellman-Ford Algorithm into our Code
- ❖ Finding a Python Library that facilitated Desktop Notifications
- ❖ Finding a Python Library and service that supported text notifications.
- ❖ Finding a Python GUI Library
- ❖ Integrating all of the above into a working application.

DEMONSTRATION



```
#bellford #1
def relax(node, neighbor, graph, d, p):
    # If the distance between the node and the neighbor is lower than the one I have now
    if d[neighbor] > d[node] + graph[node][neighbor]:
        # Record this lower distance
        d[neighbor]  = d[node] + graph[node][neighbor]
        p[neighbor] = node


#retrace function
def retrace_negative_loop(p, start):
    arbitrageLoop = [start]
    next_node = start
    while True:
        next_node = p[next_node]
        if next_node not in arbitrageLoop:
            arbitrageLoop.append(next_node)
        else:
            arbitrageLoop.append(next_node)
            arbitrageLoop = arbitrageLoop[arbitrageLoop.index(next_node):]
            return arbitrageLoop


#bellman_final
def bellman_ford(graph, source):
    d, p = initialize(graph, source)
    for i in range(len(graph)-1): #Run this until is converges
        for u in graph:
            for v in graph[u]: #For each neighbor of u
                relax(u, v, graph, d, p) #Lets relax it


    # Step 3: check for negative-weight cycles
    for u in graph:
        for v in graph[u]:
            if d[v] < d[u] + graph[u][v]:
                return(retrace_negative_loop(p, source))
    return None
```

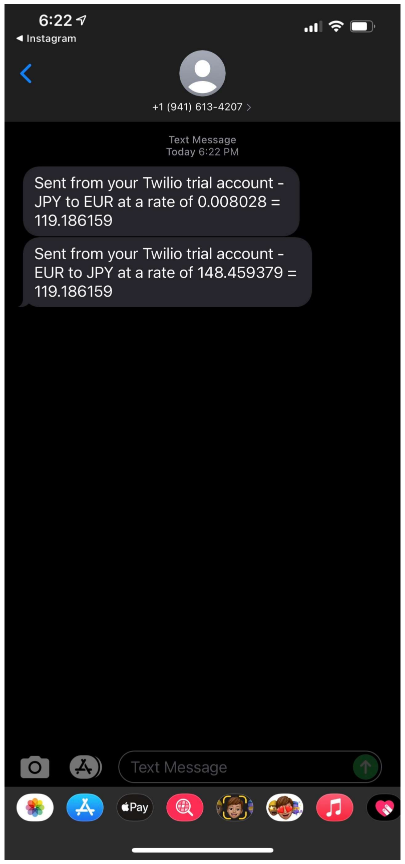
DEMONSTRATION II



Python

<---Arbitrage cycle detected--->
Starting with 100 in EUR

DEMONSTRATION III

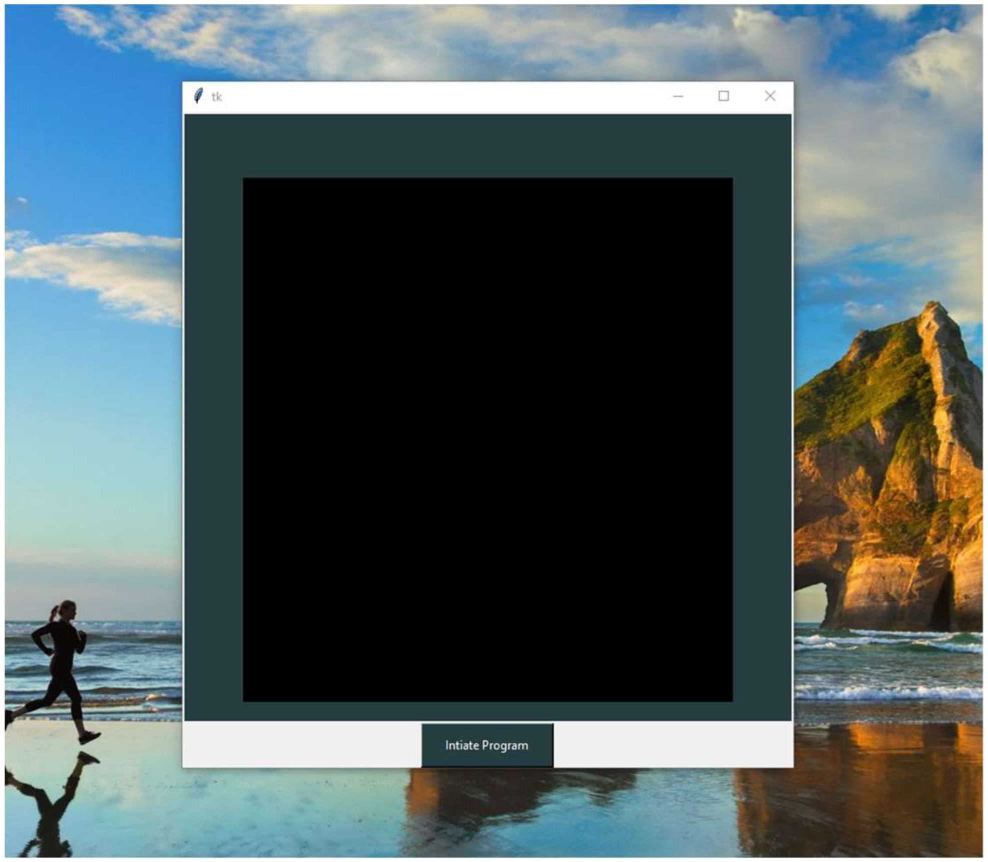

6:22
◀ Instagram
+1 (941) 613-4207

Text Message
Today 6:22 PM

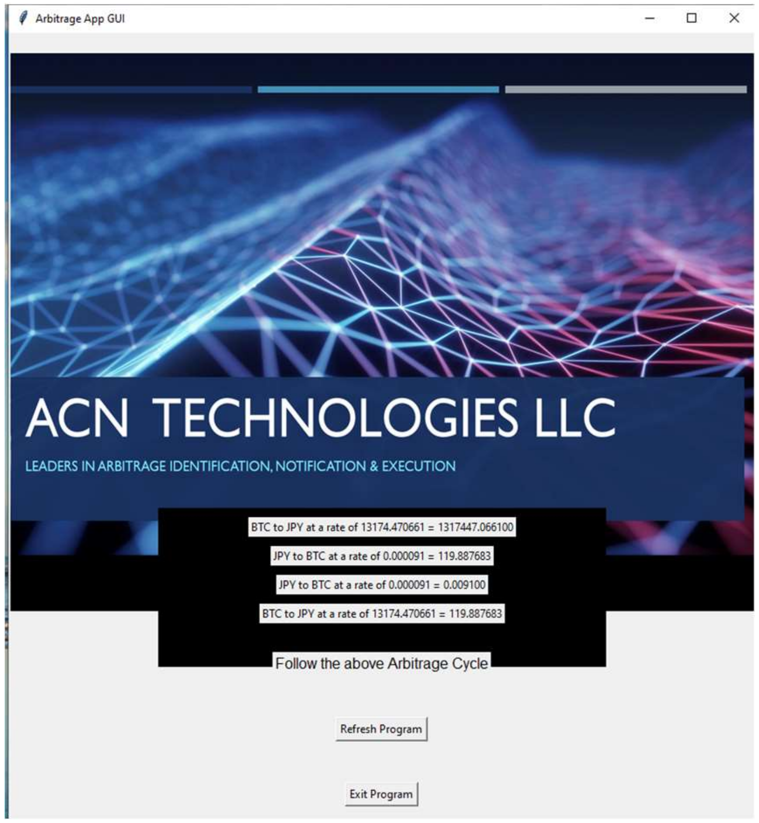
Sent from your Twilio trial account - JPY to EUR at a rate of 0.008028 = 119.186159

Sent from your Twilio trial account - EUR to JPY at a rate of 148.459379 = 119.186159

DEMONSTRATION IV



DEMONSTRATION V
UPDATED GUI


Arbitrage App GUI
ACN TECHNOLOGIES LLC
LEADERS IN ARBITRAGE IDENTIFICATION, NOTIFICATION & EXECUTION

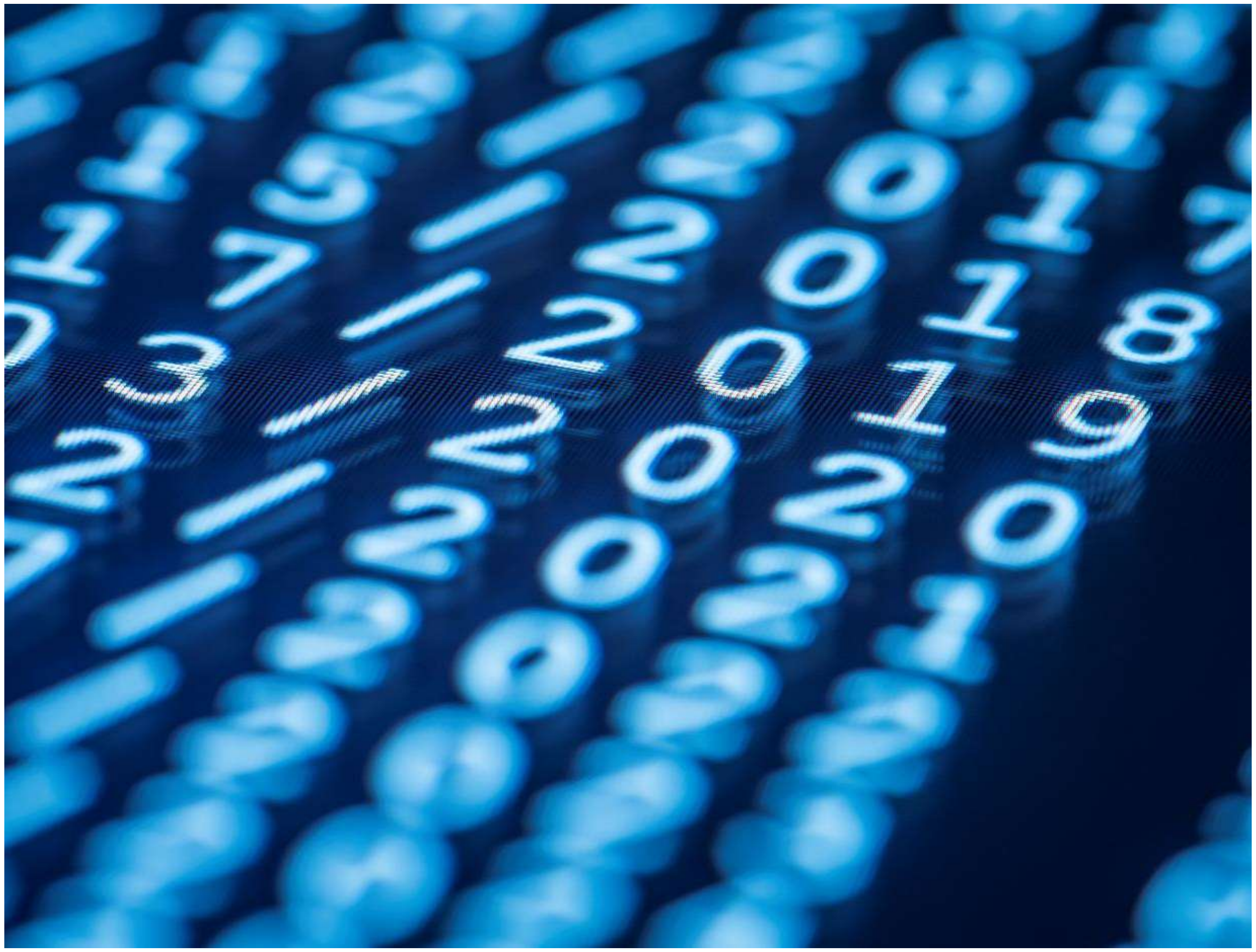
```
BTC to JPY at a rate of 13174.470661 = 1317447.066100
JPY to BTC at a rate of 0.000091 = 119.887683
JPY to BTC at a rate of 0.000091 = 0.009100
BTC to JPY at a rate of 13174.470661 = 119.887683

Follow the above Arbitrage Cycle
```

CONCLUSIONS

- The application was able to detect arbitrage opportunities within less than five seconds of initialization
- Project Goal was Achieved!!!!!!

NEXT STEPS

- ❖Further build out and expand the User Interface
- ❖Increase the currency choices and customization capabilities
- ❖Develop an Institutional Iteration of the application



THANK YOU

ACN TECHNOLOGIES TEAM:

PROJECT MANAGER:

BABAJIDE ADEMOLA

LEAD SOFTWARE ENGINEER:

NATHAN NELSON

LEAD UI ENGINEER:

JOEL CARBALLO