A Prototypical Full-Stack Digital Twin Implementation for Optimizing Airport Security Screening Processes

Raymond **David**
Sean **Park**
Hyoju **Kang**
Shaun **Ho**
Jinny **Kim**

Carnegie Mellon University

Abstract

The Transportation Safety Administration (TSA) screening process at national airports has received extensive attention in the Operations Management literature, with various attempts being made to apply queue and process flow management theory toward the optimization of passenger flow through security checkpoints. A literature review conducted during our previous work uncovered at least one work that posited the feasibility of conceptualizing the airport screening process within a Digital Twin framework. In that previous work, we added to the discourse by formally identifying the relevant Operations Management theories and incorporating them into a fully-elucidated Digital Twin framework, thus confirming the feasibility of developing a Digital Twin framework for this specific problem. In this work, we further that proof-of-concept by developing a prototypical full-stack Digital Twin implementation with a fully functioning front-end and back-end centered on a question-answering (QA) system. In doing so, we build on the observations and lessons learned from our initial trial implementation pertaining to ease of use in human-computer interaction.

I. Introduction

The efficiency of the TSA security screening process is pivotal to the reliability of nationwide air travel. Comprising several critical stages, the TSA process is designed to ensure the safety and security of passengers as they prepare to board their flights. This report examines the key components of TSA screening, which include the verification of identification and travel documents, the scanning of personal items, and physical pat-downs when necessary, and the digital twin solution we propose to enhance the TSA process.

TSA checkpoints play a pivotal role in ensuring airport security and passenger flow, yet they can become points of frustration affecting both airlines and passengers. Frequently, the time required to process travelers varies at each checkpoint. This discrepancy can prevent TSA agents from performing their duties continuously, as bottlenecks at subsequent checkpoints cause delays that ripple back through the line. Furthermore, when staff are not positioned optimally—either overstaffed at low-activity points or understaffed at high-traffic areas—the risk of delays and missed flights escalates. This can have a ripple effect, causing not just immediate inconvenience but also financial repercussions for airlines due to rebooking costs, compensation, and reduced

customer satisfaction. With air travel figures on the rise each year, the imperative for TSA to refine its processes and improve queue management is evident. Enhancing these systems could lead to better throughput, fewer delays, and a smoother travel experience for passengers.

We propose the integration of a Digital Twin into TSA operations as a promising solution to these challenges. This innovative approach, which simulates and analyzes real-world airport security processes, can equip TSA officers with actionable insights for strategic decision-making. By virtually modeling passenger flow and staff allocation, the Digital Twins framework can help in predicting bottlenecks and optimizing resource allocation, thus minimizing the chances of flight misses and delays. This technology offers a forward-looking strategy that, while not a cure-all, especially for issues related to passenger behavior, represents a significant step towards modernizing and streamlining airport security operations.

II. Model Overview

Creating an ultra-realistic digital rendition of the TSA process is crucial to validate our project as a Digital Twin. This model will act as a comprehensive digital counterpart, enabling seamless testing, integration, and simulations without requiring significant corresponding testing in real-world TSA operations, staying true to the fundamental characteristics of a Digital Twin (Arnautova, 2023). Enhancing the applicability and relevance of our Digital Twin involves incorporating familiar concepts from traditional process analysis, allowing us to harness the advantages from both approaches.

Among the aspects of process analysis, our focus gravitates towards assessing capacity and utilizing queuing theory to determine waiting times. Operations management introduces three distinct methodologies for capacity planning: Lag Strategy, Lead Strategy, and Match Strategy. The *Lag Strategy* aims to maintain adequate resources to meet the present demand (Simplilearn, 2023). In contrast, the *Lead Strategy* extends beyond current demand to accommodate estimates, incorporating additional capacity beyond the system's current needs (Simplilearn, 2023). The *Match Strategy* integrates elements of both previous approaches, considering current demand, demand projections, and market conditions (Simplilearn, 2023). Our project aims to assist the TSA in implementing the Match Strategy . It's essential to note that, in the TSA context, demand originates from incoming passengers into the system, unlike the typical "units" leaving the system in a conventional business setting.

Functionally, our model will allow TSA personnel to access baseline performance metrics and offer them insights into its current state and future trajectory. It will also uncover existing and potential bottlenecks while showcasing how modifications can impact both present and future performance.

III. Methodology

We deployed a systematic four-phase approach. Initially, we constructed an Excel spreadsheet to meticulously assess the feasibility and functionality of our project, striving to create an intuitive and precise framework. Following this, we transitioned our Excel model to Python, enriching it with increased flexibility, adaptability, and reproducibility. The subsequent phase involved incorporating the fundamental logic of our model into a QA System, integrating it seamlessly within our user-friendly front-end application. Lastly, we validated the system's outputs using *SimQuick* simulations, to generate diagnostics.

We evaluate the success of our project based on the following criteria:

(1) Enhanced planning for travelers, enabled by more accurate wait-time predictions.
(2) Improved allocation and utilization of additional personnel during peak travel periods.
(3) Augmentation of passenger satisfaction due to a more streamlined and efficient process.
(4) More timely flight departures, and reduced costs arising from boarding delays.

IV. Challenges from Previous Work

We encountered several challenges in improving our optimization module over the initial prototype in our previous submission.

First, the introduction of nonlinearity into our objective function requires us to ensure the convexity of the queuing theory function. Once proven, our focus shifted to confirming that none of the constraints violated the conventional criteria of a convex program, a notably intricate task due to the significance of a convex objective function for meaningful minimization within our optimization module. Absence of convex sets would render our module non-functional.

Another pertinent challenge emerged concerning the distinction between global and local minima. While the convex nature of our objective function ensures that local minima is the global maxima, careful consideration was essential to navigate potential entrapment at stationary points during specific iterations of the Gradient Descent Algorithm. Intelligent manipulation of the starting values for decision variables was necessary to avert confinement at local minima.

Furthermore, a substantial hurdle arose concerning the scale of feasible solutions within our model. With 40 binary decision variables, the number of potential values escalates to 2^{40} (or 1,099,511,627,776). Resolving a problem of this magnitude demanded considerable computational power, although manageable on a regular personal computer. Solver encountered over 4000 subproblem iterations before arriving at an optimal solution, highlighting the criticality of preventing mid-process errors, such as Zero Division Errors, to prevent crashes, necessitating additional domain-related constraints.

Moreover, the incorporation of multiple parameters into the queuing theory introduced additional complexities, demanding adjustments in nearly all formulas from Mini 1. These modifications,

however, enabled us to unify our model for both supply-constrained and demand-constrained scenarios, eliminating the need for two separate models as previously done in Mini 1.

V. Upgraded Model

We managed the above challenges by taking the following steps: First, we opted for a new spreadsheet due to the extensive anticipated modifications. Returning to first principles, we re-evaluated the quality of our training data and decided to re-import our data and obtain more samples to avoid the "Garbage-In-Garbage-Out" problem. To do this we utilized Python to scrub the TSA throughput dataset for the Pittsburgh Airport, encompassing data from the past year. This new, clean dataset formed the basis of our final prototype, residing in the 'TSA Final Project Model' under the 'New Data' tab.

We then defined the specifications of our "Data Pulling" tool which would form the interface between user inputs and the Digital Twin's analysis. We defined the inputs as follows: Expected number of passengers at Pittsburgh Airport, day, time of day, the maximum number of available TSA officers for a shift, and TSA Employee ID numbers (generated from synthetic employee database within the 'EmployeeData' tab).

Our subsequent task was to delve into the model's technicalities, incorporating new and more granular assumptions and adjustments over and above the model we introduced in previous work:

- Customer arrivals follow an exponential distribution.
- A single TSA employee can manage a lane (e.g., ID Verification Lane, Baggage Lane, AIT Lane, Pat-down Lane).
- Each employee must be assigned to a stage, impacting queueing theory's server allocation and wait times.
- Every stage requires at least one employee's service.
- Activity time variability follows a normal distribution (CVa=1, CVp =1).

By utilizing these updated parameters our model was well-equipped to return the envisaged outputs, namely: Waiting times for each stage, total service time for passengers, and work-in-process inventory insights within the TSA.[1] We specify further details pertaining to model mechanics in the following page. For details on the full implementation of the model, proceed to Section VI.

[1] As an added caveat, we added an optimization module that converted decision variables into binary format, allowing only assigned employees to influence capacity changes. We optimized the spreadsheet using Excel's Built-in Optimization Solver (GRG Nonlinear), elucidating the Nonlinear formulation in *Appendix H*.

Demand-constrained Process

In terms of addressing a demand-constrained process, our model does so perfectly, outputting the exact wait time estimates for each stage. We can also see the capacity and utilization across every stage. All of this information is available under 'Model Insights for Demand-Constrained' as seen in *Appendix A*. This model predicts accurately all the hours where the Pittsburgh airport is less busy, typically from 7:00pm - 2am.

Supply-constrained Process

In terms of a supply-constrained process, our model will revert to our optimization module to ensure that the process, at the minimum, will have a capacity higher than the rate of arrival. If the current capacity is not enough, Solver will not be able to find an optimal solution. However, the 'Model Insights for Supply-Constrained' section of our model will be able to output the minimum extra capacity we need in order to handle the current situation. Moreover, this alternative insights section will let us know if an alternative checkpoint is needed (*Refer to Appendix B*).

To enhance the granularity of our analysis, we returned to SimQuick to replicate our current TSA setup, specifically emulating operations on a Tuesday night at 7 pm. Through 200 simulation iterations, we derived comparable utilization metrics (Fraction Time Working) and waiting time data (Mean Cycle Time).

While it's feasible to calculate waiting times in an extremely supply-constrained scenario, it necessitates assuming a triangular distribution for waiting times, similar to our approach in Mini 1. Yet, aligning our waiting time estimates with real TSA procedures urges us to steer clear of such assumptions. Our current methodology reflects standard practices in process analysis, validated during discussions with Professor Suresh Chand, a process analysis expert, affirming the precision of our model.

Limitations

In terms of limitations, our advanced model has effectively resolved the constraints observed in our base model, presenting a highly specialized and adaptable framework. For instance, a simple change in mathematical symbols in our model allows us to determine the minimum employee count required to transition the process into a demand-constrained one, providing accurate waiting time estimates.

However, certain limitations persist. Firstly, operating the model still demands a fundamental grasp of Excel functionalities and the model's overarching concept. While we've simplified the process, basic familiarity with Excel and model comprehension remains necessary.

Secondly, the complexity of version control poses a challenge. Accidental deletion of cells or rows by TSA staff can lead to system crashes. Furthermore, updates to core assumptions and model inputs are managed locally, making comprehensive version control challenging.

Another constraint lies in the model's specificity. While our model accurately replicates TSA processes, extending this analysis to other scenarios demands the creation of entirely new models. Consequently, the model's extreme specificity can be considered a limitation. However, our forthcoming project stage aims to utilize Language Models (LLMs) and its generative capabilities to address this issue. Leveraging lessons from our TSA capacity studies, LLMs such as ChatGPT will replicate analogous procedures for diverse systems with minimal effort, paving the way for broader applicability.

VI. Full-Scale Implementation of QA System

The advanced excel model laid an excellent foundation for the development of our model. In order to create an amenable human-computer interaction environment which allows users to effectively deploy our model, we opted to develop a QA system. This is because the conversational nature of a QA system makes it easier for users to understand and interact with the model. It presents solutions and results in a clearer, more approachable manner, allowing users from any background, regardless of their technical expertise, to engage with it effortlessly.

Base Model

In our system's backend, two key functions form the main functionality: the wait time calculation function and the optimal staff allocation function (see Appendix C). To closely mirror our advanced model, we created a Python function for just the waiting time calculation using GPT4. We fed our Excel model into GPT4, detailing its workings and limitations, to derive the Python function.[2]

The backend also employs prompt engineering to finalize the creation of the QA system. When specific trigger words in a question are detected, the system references the corresponding function associated with that prompt. If the query doesn't match any predefined functions, it is then passed to GPT4 for generating a response. Further details on prompt engineering and its applications are elaborated on in subsequent sections.

In the remainder of this section ,we provide specifics of the QA system’s mechanics.

[2] Our base model operates on two primary assumptions. Firstly, it presumes that each checkpoint is manned by one person. The activity times in the base model are set as follows: ID Inspection takes 15 seconds, Carry-On processing 40 seconds, AIT (Advanced Imaging Technology) 15 seconds, and Pat-Down 10 seconds. Additionally, both CVa (Coefficient of Variation for arrival times) and CVp (Coefficient of Variation for processing times) are set at 1.

Calculating Wait Time

This function requires specific parameters, such as Activity Time for all checkpoints and the Arrival Rate, measured in passengers per hour. By default, the system assumes pre-set values for activity times unless specified otherwise.

When a user poses a question, the program actively scans for trigger words that could initiate the waiting time calculation. In this context, phrases like “waiting time” and “passengers” or “people” are crucial. For instance, when asked, “What is the waiting time given 100 passengers?”, the program detects the keywords and calculates the waiting time for 100 passengers accordingly. The calculated value is then presented through a formatted response in our update_output function.

However, the process is more nuanced than a straightforward calculation. Initially, the system assesses potential bottlenecks - situations where a checkpoint's activity time is insufficient, leading to delays. In technical terms, a bottleneck is identified when the utilization factor exceeds 1, signaling a backlog. If a bottleneck is detected, the system strategizes to find the minimum number of additional resources required to alleviate the issue. For example, if a bottleneck occurs at the Carry-On checkpoint with 100 passengers, the system incrementally assigns more workers to this station until the bottleneck is resolved. Only after addressing these bottlenecks does the system recalculate and provide the updated waiting time, considering the added workforce.

Optimal Staff Allocation

The next key feature of our system is the optimal staff allocation function, which builds upon the parameters used for waiting time calculation. Additionally, it considers the number of workers who are currently inactive or not engaged in the process.

In this scenario, prompt engineering plays a vital role. It identifies phrases like “sitting around” and “allocate staff” to activate the staff optimization function. For example, a user might ask, “We have 4 staff sitting around; how should we allocate them to reduce wait time for 100 passengers?” Similar to wait time calculation, the system computes the optimal staff allocation and outputs the result through a structured response in the update_output function.

To determine the best staff allocation, the backend undertakes a two-step process. First, it pinpoints the bottleneck in the base model based on the number of passengers mentioned. The initial allocation of available staff aims to eliminate this bottleneck. Subsequently, the approach branches into three logical steps:

1. If the bottleneck is resolved and all available staff have been allocated, the system outputs the updated waiting time with these new allocations.
2. If the bottleneck is resolved but surplus staff remains, the algorithm iterates through various staff distribution combinations to find the one that minimizes waiting time. Both the new waiting time and staff allocation are then presented to the user.
3. If the available staff is insufficient to address all bottlenecks, the system informs the user of the need for additional workers to resolve the issue.

While there are differences in calculation between the optimization processes of our advanced Excel model and this code, the QA system closely replicates the Excel model's calculations and has consistently produced comparable results in our trial cases.

Database

The base data on which the above calculations rely is the set of historical TSA data for Pittsburgh International Airport, sourced directly from the TSA website. This valuable dataset includes detailed information on the total number of passengers at specific entrances, dates, and times (see Appendix D). We seamlessly incorporated this dataset into our backend using a pandas dataframe. In the user interface, individuals can specify various parameters such as the Airport, date, entrance, and time. Upon entering these details, the program retrieves the passenger count from the dataset and computes the waiting times using the default activity times. For users desiring customization, we've provided a drop-down option in the front end to adjust the activity times as needed (see Appendix E). Once the waiting time is calculated, it is displayed in a structured and user-friendly format, akin to the output generated by the wait time calculation feature. This integration not only enhances user interaction but also adds a layer of real-time relevance and accuracy to the wait time predictions.

Integrating historical TSA data for Pittsburgh International Airport is crucial in enabling TSA agents to plan proactively by analyzing passenger volume trends and seasonality. This data, especially consistent year-to-year patterns during holidays, helps predict peak passenger flows. Agents can use these insights for strategic resources and staff allocation, optimizing checkpoint processes, and adjusting activity times to manage surges in passenger numbers. Such preparation not only enhances operational efficiency but also improves passenger experience by potentially reducing wait times. This approach shifts TSA operations from reactive to proactive, ensuring a more responsive and effective airport security system.

GPT4

For queries that don't activate the specific functions outlined earlier, we defer to GPT4. Leveraging the OpenAI API, we access a range of models tailored to address general inquiries. Our primary aim in integrating ChatGPT is to enhance user-friendliness and expand the scope of our front-end capabilities.

In the context of TSA agents, this integration proves particularly useful. They can utilize ChatGPT to obtain a wide array of information, from daily weather forecasts to more intricate aspects of their work. This includes up-to-date travel guidelines and procedures, detailed clarifications on security protocols, essential emergency response information, and effective resource management strategies. This versatility makes this QA system a valuable tool for TSA agents, facilitating access to a broad spectrum of information and advice to support their daily operations.

Front End Development

Our front end was designed and deployed using Dash, a versatile Python framework for building web applications (see Appendix F). This choice aligns seamlessly with our Python-based backend, allowing for a cohesive and interactive user experience. Dash's flexibility is a key asset in our design, enabling us to create a user interface that is both engaging and intuitive.

In addition to the functionalities previously outlined, we've incorporated a 'Frequently Asked Questions' (FAQ) section, conveniently positioned next to the title for easy access (see Appendix G). This feature is designed to support the user experience by providing essential information about the QA system and its capabilities. It offers a range of sample questions to guide users in interacting with the model, particularly for queries specific to our model. Furthermore, the FAQ section includes explanations of more technical aspects, such as bottlenecks and Little's Law, ensuring that even users unfamiliar with these concepts can understand and effectively use our QA system. This addition enhances the overall utility and user-friendliness of our application.

VII. Applicability of Model to TSA Operations

In this section, we describe how TSA Agents can interpret model outputs.

We can interpret utilization of each stage as the amount of time an employee remains working for a given time period, in our case, an hour. Essentially, this is an important piece of information as the TSA can attempt to lower utilization to reduce waiting times, but at the same time too low of utilization means that idle time is on the rise. Hence, this aids the TSA in finding the right balance and optimal utilization benchmarks, depending on the ultimate objective of the TSA. Optimized Max Capacity shows how much passengers we can take in using our present number of employees, after assigning them to their optimal locations. Total Time outputs how much it takes for one passenger to go through the entire system from start to finish. Total Passengers determines how much passengers are within the TSA, on average. Optimal Staff Allocation shows the optimal number of employees to be assigned in every stage. Waiting Time in Queue represents the time it takes a passenger to wait before getting a service in each of the respective stages. A snapshot of our output insights can be seen in Appendix A, or from our "TSA Final Project Model" Excel file. This output is based on a Tuesday night at 7pm in the Pittsburgh Airport.

These parameters are important in crafting an effective staff allocation and capacity strategy. From our hourly demand rate of 190 customers, the current setup with 10 employees yields a maximum capacity of 327 passengers per hour. This signifies that if the incoming passenger count exceeds 327, the system could encounter waiting time issues. Conversely, reverse engineering this analysis reveals that with 10 employees, the system's maximum output is 327 passengers per hour. Therefore, during historically high throughput periods, more than 10 employees would be required, enabling the TSA to anticipate peak hours.

Our current output extends our analysis to determine the actual service level of passengers. By establishing a Target Wait Time (TWT), we can effectively utilize our model's information to ascertain the probability of customers being served within the specified TWT (Cachon).

Moreover, currently, we haven't assigned costs to our generated outputs due to the lack of precise data on employee wages to compute service costs or definitive metrics for calculating waiting costs. However, with access to such information, these outputs could be adjusted to strike a balance between service costs and waiting expenses. Another advanced statistical approach involves utilizing the Erlang loss formula to compute the probability of all our servers/stages being occupied, particularly crucial in vital processes like the TSA (Cachon). All these extensive studies can be done purely from the outputs of our model.

Average activity time of employees plays a crucial role in waiting time, since higher variability in processing times increases overall waiting time estimates. Therefore, implementing new Standard Operating Procedures (SOPs) and employee training becomes crucial to reduce variability in processing times and overall average service duration. In a service environment, customer cooperation significantly impacts addressing service variability. For the TSA, ensuring customer familiarity with procedures reduces disruptions and enhances process efficiency.

To diminish arrival variability, promoting TSA PreCheck participation aids in spreading demand and reducing arrival fluctuations, akin to appointment scheduling concepts. Sharing wait time data effectively encourages travelers to stagger their arrivals across wider timeframes, as evidenced by Xie and Youash's findings (2011) in a hospital setting, where disseminating wait time information reduced occurrences of prolonged wait times.

Buffers also play a pivotal role in affecting capacity, as there's a limited space available for queues to form. Increasing buffer capacity has shown to diminish wait times and facilitate smoother operations. Insufficient buffers, especially before and after bottleneck stages, could obstruct upstream resources, preventing them from servicing customers, even after they've passed through a specific stage.

VIII. Insights Gained

Learnings and Challenges

As mentioned previously, our original objective was to train a personalized ChatGPT with our advanced model and incorporate that into our QA system using the OpenAI API. However, after developing the frontend and the personalized model, we faced limitations in integrating our GPT with the frontend due to the beta status of the personalization feature. As a workaround, we built functions into the backend to mimic the personalized GPT with prompt engineering. The development of our front-end presented a complex challenge, primarily focusing on three key areas:

1. **User Interface Design:** Crafting a visually appealing and intuitive interface was crucial. This involved strategically placing buttons and text fields to ensure ease of navigation.

2. **Interactivity:** Our application demanded a highly interactive front-end. This required the integration of dynamic elements, such as drop-down elements and model changes.

3. **Data Integration:** Seamlessly connecting the front-end with our backend was essential for real-time data processing. Challenges included efficiently displaying and updating outputs, maintaining accuracy, and ensuring it could handle complex data without lag.

Addressing these aspects was vital to creating a user-friendly and effective front-end for our application.

Limitations

The program currently faces certain limitations, mainly regarding its prompt engineering and response flexibility. The system's reliance on specific keywords means it might overlook differently-phrased prompts. Additionally, the program's hard-coded responses, while operationally sound, limit the QA system conversational fluidity, a key aspect of the user experience. Moreover, the program lacks memory, causing it to revert to the base model for each new query. This could necessitate multiple steps for users to reach their desired answers.

To overcome these challenges, our plan includes integrating a trained, personalized GPT4 model into the backend. This enhancement will significantly improve the program's ability to recognize various phrasings and nuances in user queries. The personalized GPT model will also bring more natural, conversational responses, enriching user interactions. Furthermore, this model will maintain a history of previous interactions on the server, allowing for the use of context to create a more seamless and continuous user experience, eliminating the need for repeated updates.

IX. Future Applications

The TSA Security screening approach, notable for its multi-level checks and intervening buffers, shows potential for adoption in various contexts where security and efficient movement are priorities. This screening method has been proven to be effective in settings such as concerts and theme parks, enhancing safety and customer experience. However, its application in more complex environments, like healthcare, demands a nuanced and detailed adaptation of the model.

In healthcare, specifically, hospital visits mirror the TSA security check's structure but with significantly more complexity. A hospital visit typically involves a series of steps: check-in at reception, doctor consultation, and receiving treatment or a prescription, each separated by considerable wait times. However, the unique nature of each patient's health condition, requiring distinct treatment paths, adds layers of complexity, differentiating it from the more uniform process at TSA checkpoints. Moreover, the consequences of errors in healthcare are far more serious than at TSA checkpoints, necessitating a more specialized approach.
.
To further enhance this model's applicability, incorporating QA systems presents a notable improvement. In environments like concerts, theme parks, and fast-food restaurants, these QA systems will offer real-time wait time updates and identify bottlenecks, suggesting reallocation of staff to reduce congestion. In healthcare, they will aid in managing patient flow by advising on staff distribution based on current waiting periods. This integration not only optimizes operations but also significantly improves customer and patient experiences, demonstrating our model's adaptability and effectiveness in a range of environments.

X. Appendix

[Appendix A]: Mini 2 Advanced Model Insights A (Demand-Constrained)

Model Insights for Demand-Constrained

Idle Employees 0

Stage	Utilization	Capacity
ID	49%	384
Carry-on	48%	393
AIT	58%	327
Pat-Down	16%	1160

Max Capacity 327 *passengers/hour*

Total Time 66.28 *Seconds*
1.10 *Minutes*

Total Pax 3 *Passengers within TSA at any point in time*

Stage	Optimal Staff Allocation
ID	2
Carry-on	4
AIT	1
Pat-Down	3

Stage	Waiting Time In Queue
ID	9.181701031
Carry-on	8.573684804
AIT	15.26819795
Pat-Down	0.60789193

[Appendix B]: Mini 2 Advanced Model Insights B (Supply-Constrained)

Model Insights for Supply-Constrained
If solver can't find a solution

Stage	Extra Capacity Needed	Min Activity Time Needed	
ID	0	#DIV/0!	Additional Worker Minimum Activity Time
Carry-on	0	#DIV/0!	Additional Worker Minimum Activity Time
AIT	0	#DIV/0!	Additional Worker Minimum Activity Time
Pat-Down	0	#DIV/0!	Additional Worker Minimum Activity Time

Note: 'Extra Capacity Needed' column will default to 0 and 'Min. Activity Time Needed' column will show '#DIV/0!' if solver is able to optimize
Note: If solver cant optimize, this is the relevant Model Insights analysis to use

[Appendix C]: Waiting Time Function Generated by PaLM2

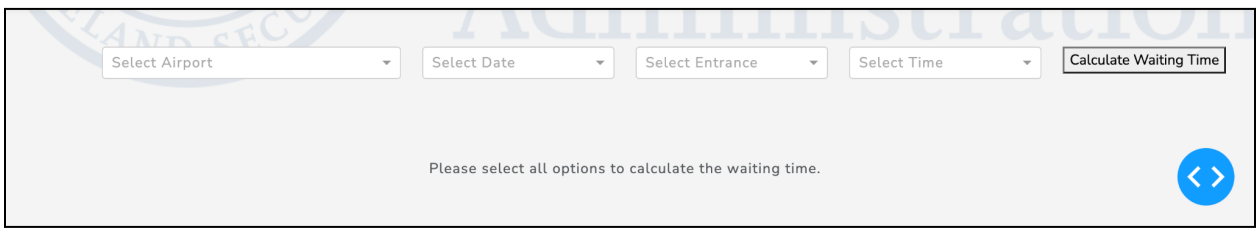
```python
# Function to calculate waiting time at each checkpoint
14 usages
def calculate_waiting_time(arrival_rate, CVa, CVp, activity_time):
    if arrival_rate == 0:
        return None, "Error: Arrival rate cannot be zero."
    A = (1 / arrival_rate) * 60
    utilization = activity_time / A
    if utilization >= 1:
        return None, "Error: Utilization is 100% or more, system is overloaded."
    utilization_factor = utilization / (1 - utilization)
    waiting_time = activity_time * utilization_factor * ((CVa ** 2 + CVp ** 2) / 2)
    return waiting_time, f"Estimated waiting time: {waiting_time:.2f} seconds"
```

```python
def allocate_staff(staff_count, people_count, updated_times):
    if staff_count == 0:
        return "No additional staff available for allocation."

    allocated_staff = {checkpoint: 1 for checkpoint in updated_times}

    # Identify initial bottlenecks
    bottleneck_checkpoints = []
    for checkpoint in updated_times:
        time, _ = calculate_waiting_time(people_count / 60, DEFAULT_CVa, DEFAULT_CVp, updated_times[checkpoint])
        if time is None:
            bottleneck_checkpoints.append(checkpoint)

    while staff_count > 0:
        for checkpoint in bottleneck_checkpoints:
            if staff_count > 0:
                allocated_staff[checkpoint] += 1
                staff_count -= 1
                new_time, _ = calculate_waiting_time(people_count / 60, DEFAULT_CVa, DEFAULT_CVp, updated_times[checkpoint] / allocated_staff[checkpoint])
                if new_time is not None:
                    bottleneck_checkpoints.remove(checkpoint)
```

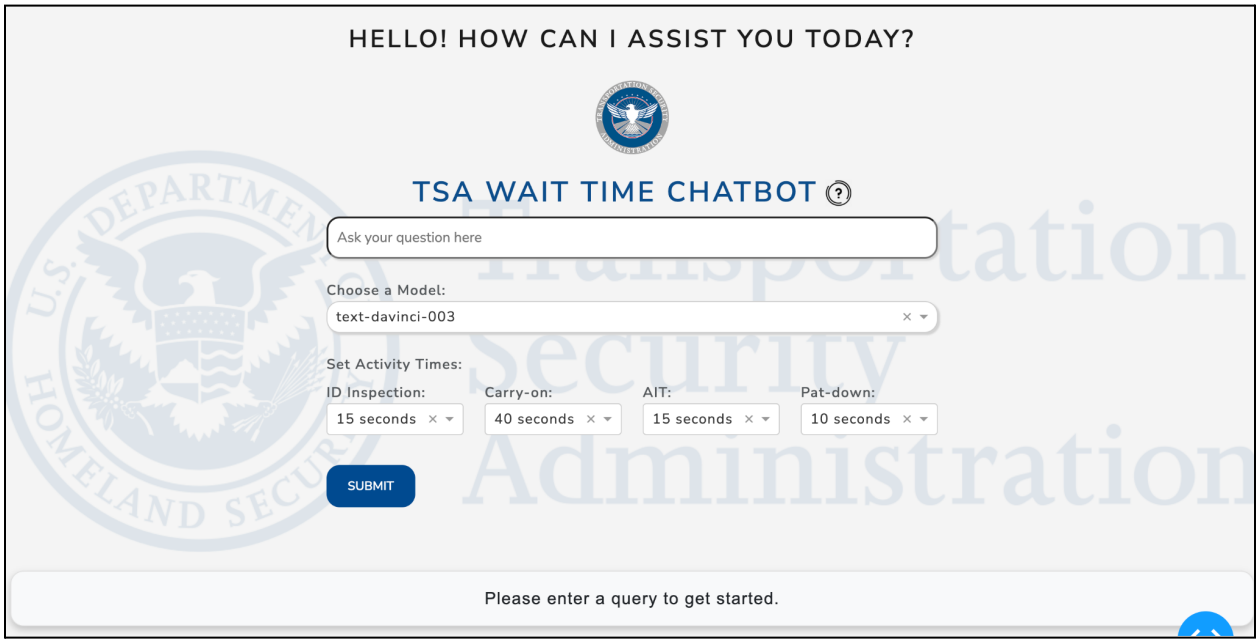
[Appendix D]: Official TSA Throughput Data

A	B	C	D	E	F	G
Date	Hour of Day	Airport	City	State	Entrance	Total Passengers
1/1/23	00:00	PIT	Pittsburgh	PA	Main Checkpoint	10
1/1/23	23:00	PIT	Pittsburgh	PA	Main Checkpoint	11
1/1/23	22:00	PIT	Pittsburgh	PA	Main Checkpoint	21
1/1/23	21:00	PIT	Pittsburgh	PA	Main Checkpoint	76
1/1/23	20:00	PIT	Pittsburgh	PA	Main Checkpoint	96
1/1/23	19:00	PIT	Pittsburgh	PA	Main Checkpoint	186
1/1/23	18:00	PIT	Pittsburgh	PA	Main Checkpoint	310
1/1/23	16:00	PIT	Pittsburgh	PA	Main Checkpoint	874
1/1/23	15:00	PIT	Pittsburgh	PA	Main Checkpoint	781
1/1/23	14:00	PIT	Pittsburgh	PA	Main Checkpoint	718
1/1/23	13:00	PIT	Pittsburgh	PA	Main Checkpoint	644
1/1/23	12:00	PIT	Pittsburgh	PA	Main Checkpoint	543
1/1/23	11:00	PIT	Pittsburgh	PA	Main Checkpoint	427
1/1/23	17:00	PIT	Pittsburgh	PA	Main Checkpoint	606
1/1/23	09:00	PIT	Pittsburgh	PA	Main Checkpoint	689
1/1/23	10:00	PIT	Pittsburgh	PA	Main Checkpoint	820
1/1/23	01:00	PIT	Pittsburgh	PA	Main Checkpoint	2
1/1/23	02:00	PIT	Pittsburgh	PA	Main Checkpoint	11
1/1/23	03:00	PIT	Pittsburgh	PA	Main Checkpoint	501
1/1/23	04:00	PIT	Pittsburgh	PA	Main Checkpoint	875
1/1/23	05:00	PIT	Pittsburgh	PA	Alternate Checkpoint	400
1/1/23	04:00	PIT	Pittsburgh	PA	Alternate Checkpoint	181
1/1/23	05:00	PIT	Pittsburgh	PA	Main Checkpoint	698
1/1/23	06:00	PIT	Pittsburgh	PA	Alternate Checkpoint	152
1/1/23	06:00	PIT	Pittsburgh	PA	Main Checkpoint	663
1/1/23	07:00	PIT	Pittsburgh	PA	Main Checkpoint	318
1/1/23	08:00	PIT	Pittsburgh	PA	Main Checkpoint	624
1/2/23	16:00	PIT	Pittsburgh	PA	Main Checkpoint	1058
1/2/23	13:00	PIT	Pittsburgh	PA	Main Checkpoint	1023

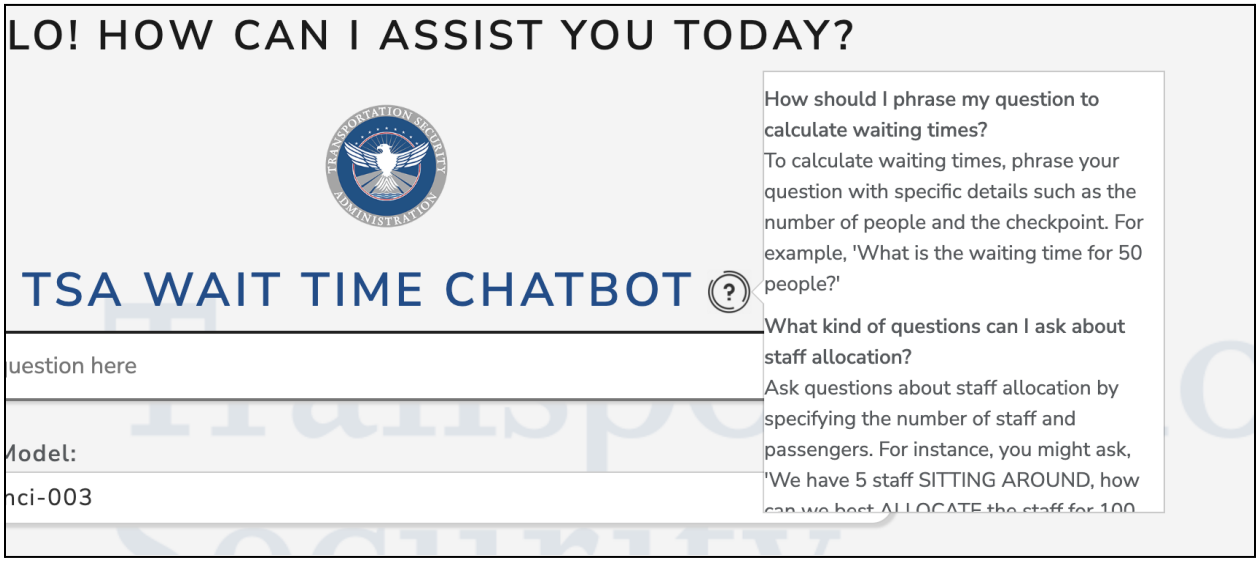
[Appendix E]: QA System Data Pulling Tool



[Appendix F]: QA System Interface



[Appendix G]: QA System FAQs



[Appendix H]: Multiobjective Nonlinear Integer Optimization Function

Integer Nonlinear Optimization

Decision Variables:

$$X_{ij}: \begin{cases} \mathbf{1}\ \textit{if employee i is assigned to stage j} \\ \mathbf{0}\ \textit{otherwise} \end{cases}$$

Objective Function:

$$\textit{Minimize} \sum_{j=1}^{N} Tq_j = A_i * \left(\frac{U_j}{1-U_j}\right) * \left(\frac{CVA_i^2 + CVP_i^2}{2}\right), \text{where } Tq_j = \text{Average Waiting time for Stage j}$$

Parameters:

$A_i = Activity\ time\ for\ employee\ i$
$B_t = Total\ Number\ of\ employees\ at\ time\ period\ t$
$C_t = Arrival\ data\ for\ customers\ at\ time\ period\ t$
$U_j = Utilization\ of\ stage\ j$
$CVA_i = Arrival\ Variability\ for\ stage\ i$
$CVP_i = Service\ Variability\ for\ employee\ i$

Constraints:

$\sum_j X_i \leq 1\ Employee\ Workload\ Constraint$
$\sum_i X_j \geq 1\ Stage\ Allocation\ Constraint$
$\sum_{i,j} X_{ij} \leq B_t\ Total\ Employee\ Constraint$
$\sum \left(\frac{1}{A_i * X_{ij}} \times 3600\right) \geq C_t\ Utilization\ Constraint$
$X_{ij}\ binary$

[Appendix J]: Individual Team Contribution

Name	Presentation	Report
Raymond David	- Created the advanced excel model version - "Implementation on Advanced Model and Parameters (Mini2)" slides - Nonlinear Integer Optimization Formulation - Excel Model Demonstration	- Implementation and and methodologies (Section) - Challenges With Building the Fully-fledged Excel Model (Section) - Upgraded Model in Mini 2 & Results (Section) - Insights Section
Shaun Ho	- Review and fix any inaccuracies or language errors in the presentation.	- Abstract - All formatting - All referencing (in-text and bibliography) - All copywriting and titling - Academic journals research
Sean Park	- TSA Wait-Time QA system - QA system Full-Stack Development (Frontend + Backend) + ReadMe	- Introduction to the TSA security check process (Section) - Current Challenges faced at TSA airport checkpoints (Section)

		- QA system (Section)
Hyoju Kang	- Structuring PPT slides and putting contents - "Considerations for Enhancing Performance and User Experience" and "Challenges with PaLM2"	- Index - Introduction to the TSA security check process (Section) - Current Challenges faced at TSA airport checkpoints (Section)
Jinny Kim	- Reformatted and restructured all the presentation - Did the all the introductory slides and the Mini 1 model discussions	- Aided in reformatting the overall structure of the report - Reformatted the Appendix - Helped write the Implementation and and methodologies

****We affirm that each of us is satisfied with the contributions of all group members toward the completion of the presentation and report. ****

Supplementary Materials:

Design & Run-time parameters formulas:
Design Parameters:
Stage Capacity (single resource) = 1 / activity time
Stage Capacity (multiple resource) = sum(individual resource capacity)
Process Capacity = min{stage capacity}
Designed Cycle Time = 1 / process capacity
Run Time:
Flow Rate = min{process capacity, demand rate}
Actual Cycle Time = 1 / flow rate
Stage Utilization = Flow Rate / Stage Capacity
Process Utilization = Flow Rate / Process Capacity
Idle Time = Actual Cycle Time - Activity TIme
Direct Labor Cost = Total Wages / Flow Rate

(Adapted from or Attributed to Stevenson, 2020)

Waiting time calculations:

Average Waiting Time = Average Service Time * Utilization Factor * ((Arrival Variability + Service Variability) / 2) (Stevenson, 2020)

Average Total Time = Time in Queue + Service (Stevenson, 2020)

References

Arnautova, Y. (2023, March 16). Digital Twins Technology, its benefits & Challenges to Information Security. GlobalLogic. https://www.globallogic.com/insights/blogs/if-you-build-products-you-should-be-using-digital-twins/

Simplilearn. (2023, June 28). What is capacity planning? definition, methodologies, benefits. Simplilearn.com. https://www.simplilearn.com/capacity-planning-article

Cachon, G. (n.d.). Matching Supply with Demand. In Matching Supply with Demand.
Evaluating Process Capacity. (n.d.). Chapter 3. Retrieved October 10, 2023, from http://www2.nkfust.edu.tw/~smguo/teaching/slides/OM_Process_Capacity_2015.pdf

FOIA Electronic Reading Room. (n.d.). Transportation Security Administration. Retrieved October 10, 2023, from https://www.tsa.gov/foia/readingroom

Frequently Asked Questions. (n.d.). Transportation Security Administration. Retrieved October 10, 2023, from https://www.tsa.gov/travel/frequently-asked-questions

Park, Y., & Ahn, S. B. (2006). Optimal assignment for check-in counters based on passenger arrival behaviour at an airport. *Transportation Planning and Technology*, *26*(5). Taylor & Francis. https://www.tandfonline.com/doi/abs/10.1080/03081060310001635887

Pekoske, D. P. (2022, May 26). *The State of the Transportation Security Administration*. Transportation Security Administration. Retrieved October 10, 2023, from https://www.tsa.gov/news/press/testimony/2022/05/26/state-transportation-security-administration

Simquick. (n.d.). SimQuick – Process Simulation with Excel. Retrieved October 10, 2023, from https://simquick.net/

Stevenson, W. J. (2020). *Operations Management*. McGraw-Hill Education.
Travel - Security Screening. (n.d.). Transportation Security Administration. Retrieved October 10, 2023, from https://www.tsa.gov/travel/security-screening

Xie, B., & Youash, S. (n.d.). The effects of publishing emergency department wait time on patient utilization patterns in a community with two emergency department sites: a retrospective, quasi-experiment design. *International Journal of Emergency Medicine*, *4*(29). NIH. https://www.google.com/url?q=https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3123545/&sa=D&source=docs&ust=1696972467819363&usg=AOvVaw1XLEiupTOxzMV3zkhwKgZ8

Report Content Includes Self-Cited Works From The Following Items:

David, R., Park, S., Kang, H., Ho, S. (2023). Investigating the Applicability of Digital Twin Technology in the Management and Capacity Analysis of Airport Security Screening Processes.

David, R., Park, S., Kang, H., Ho, S. (2023). Toward Augmenting the Development and Implementation of Digital Twins with Google's PaLM Suite.

David, R., Park, S., Kang, H., Ho, S. (2023). Final Project Check-in.